# Ma´ster en Visi´on Artificial

### Curso 2023-2024



**APUNTES DE FUNDAMENTOS MATEMA´TICOS: I PARTE - ESTAD´ISTICA**

### Autores:

Iv´an Ram´ırez, Victoria Ruiz, Emanuele Schiavi

©2023 Iv´an Ram´ırez D´ıaz, Victoria Ruiz Parrado, Emanuele Schiavi. Al- gunos derechos reservados. Este documento se distribuye bajo la licencia “Atribuci´on-CompartirIgual 4.0 Internacional” de Creative Commons,

disponible en <https://creativecommons.org/licenses/by-sa/4.0/deed.es>.

## ´Indice

1. [**I Clase**](#_bookmark0) **3**
   1. [Introducci´on y Material](#_bookmark1) 3
   2. [Fundamentos de Estad´ıstica Bayesiana.](#_bookmark2) 3
      1. [Probabilidad Condicionada](#_bookmark3) 5
      2. [Independencia e Independencia Condicional](#_bookmark5) 7
   3. [Variables Aleatorias](#_bookmark6) 11
      1. [Variables Multimensionales](#_bookmark11) 26
   4. [Distribuciones de Probabilidad.](#_bookmark12) 29
      1. [Distribuciones Discretas](#_bookmark13) 29
      2. [Distribuciones Continuas](#_bookmark19) 38
2. [**II Clase**](#_bookmark23) **47**
   1. [Estimadores](#_bookmark24) 47
      1. [Estimaci´on Puntual](#_bookmark25) 47
      2. [Estimador de M´axima Verosimilud](#_bookmark26) 51
      3. [Estad´ısticos Bayesianos](#_bookmark27) 54
   2. [Aprendizaje Autom´atico.](#_bookmark28) 56
      1. [Capacidad, Sobre Entrenamiento y Bajo Entrenamiento](#_bookmark29) 56 [2.2.2. Sesgo -Varianza](#_bookmark31) 58
   3. [Ejercicios Propuestos](#_bookmark33) 59

## I Clase

#### Introduccio´n y Material

En las pr´oximas dos clases se van a explicar los conceptos b´asicos de la teor´ıa de la probabilidad, la teor´ıa de la informaci´on y aprendizaje profundo.

La teor´ıa de la probabilidad nos proporciona un medio de cuantificar la incer- tidumbre. En aplicaciones de Inteligencia Artifical (IA) se usa la probabilidad principalmente en dos contextos. En el primero, si se conoce exactamente c´omo deber´ıa funcionar el sistema pero no tengo recursos suficientes para construirlo, se puede usar la probabilidad para aproximar el modelo real y/o mejorar su eficiencia. En segundo lugar, una vez construido el sistema, se puede usar la probabilidad y estad´ıstica para analizar te´oricamente el com- portamiento de los sistemas propuestos.

Por otro lado, mientras que la probabilidad nos permite hacer afirmaciones y razonamientos en presencia de incertidumbre, en teor´ıa de la informaci´on se puede cuantificar la cantidad de incertidumbre que hay en una distribuci´on de probabilidad.

Por u´ltimo, el aprendizaje autom´atico usa las herramientas anteriores para construir algoritmos que realicen una tarea concreta a partir de los datos de nuestro problema. Es decir, el ordenador “aprende” de nuestros datos. Den- tro del aprendizaje autom´atico se encuentra el *Deep Learning* o aprendizaje profundo, que ser´a uno de los temas que se vea en esta y otras asignaturas del M´aster.

#### Fundamentos de Estad´ıstica Bayesiana.

Comenzamos explicando las nociones b´asicas de la teor´ıa de la probabilidad en general y de los fundamentos de estad´ıstica Bayesiana en particular.

Dentro de la teor´ıa de la probabilidad se distingue la *probabilidad frecuentista*

de la *probabilidad Bayesiana*.

La probabilidad frecuentista se basa en la frecuencia relativa de los sucesos tras una larga serie de repeticiones de un experimento. Segu´n esta interpre-

taci´on, la probabilidad de un determinado suceso ser´a la frecuencia relativa de este despu´es de que el experimento se haya repetido un nu´mero grande de veces. Por ejemplo, la probabilidad de obtener un seis al lanzar un dado se considera que es 1*/*6, porque si el dado fuera lanzando un gran nu´mero de veces en circunstancias similares, esa ser´ıa aproximadamente la frecuencia relativa del suceso consistente en obtener un seis. No obstante muchas veces no se puede repetir el experimento un gran nu´mero de veces (por ejemplo en ensayos cl´ınicos), ni siempre se tienen condiciones similares.

Por otra parte, en la probabilidad Bayesiana, debida a Bayes, se utiliza infor- maci´on *a priori*. Es decir, se utiliza el conocimiento previo que se tiene antes de realizar el experimento. Sin embargo, esta informaci´on puede cambiar en funci´on de qui´en o c´omo se realiza el experimento. Si no se tiene informaci´on se supondr´a que todos los eventos tienen la misma probabilidad de ocurrir (es decir, tienen distribuci´on uniforme como se ver´a m´as adelante). Por tanto, uno de los problemas de esta interpretaci´on es que la informaci´on *a priori* no est´e un´ıvocamente determinada.

Antes de desarrollar m´as la estad´ıstica Bayesiana, presentaremos los con- ceptos b´asicos de la teor´ıa de la probabilidad, aplicables a ambas ramas, y finalmente nos centraremos en la probabilidad Bayesiana.

**Experimento, espacio muestral, muestra y suceso** Supongamos que tenemos un dado regular, no trucado, con sus 6 caras marcadas con 1,2,3,4,5 y 6, respectivamente. Consideremos los siguientes experimentos:

1. Dejar caer el dado desde una altura de 1 metro y medir el tiempo que tarda en llegar al suelo.
2. Lanzar el dado al aire y, una vez en reposo, observar la puntuaci´on que aparece en su cara superior.

¿Qu´e diferencia hay entre los dos experimentos? En ocasiones, los resultados de un experimento constituyen una magnitud que puede obtenerse, mediante una ley matem´atica precisa, como funci´on de otras magnitudes previamente fijadas. En este caso, se suele decir que estamos ante un fen´omeno o expe- rimento *determinista*. Es decir, la repetici´on del experimento en las mismas condiciones produce el mismo resultado. En el caso del experimento 1, el

dado tardar´a el mismo tiempo en llegar al suelo cada vez que lo dejemos caer desde la misma altura.

En otros casos, la repetici´on del experimento en id´enticas condiciones no permite asegurar que el resultado vaya a ser el mismo. Decimos que un ex- perimento de este tipo es *aleatorio*; es decir, su resultado depende del azar. Formalmente, un *experimento aleatorio* es un proceso cuyo donde:

Se conocen todos sus posibles resultados

No puede predecirse cu´al de ellos se producir´a en una experiencia con- creta.

Realizado en condiciones an´alogas, puede dar lugar a resultados distin- tos en cada experiencia en particular.

El conjunto de resultados posibles de dicho experimento es lo que se conoce como *espacio muestral*, *S*. Los resultados concretos de realizar el experi- mento se denominan *muestras*. En el caso del dado, su espacio muestral ser´ıa *S* = 1*,* 2*,* 3*,* 4*,* 5*,* 6 . Sin embargo, no todos los experimentos aleatorios tienen resultados num´ericos; por ejemplo, al tirar una moneda el espacio muestral ser´ıa *S* = *{cara, cruz}*.

*{ }*

Por otra parte, se denomina *suceso* asociado al experimento *E* a cualquier subconjunto, *A S*, del espacio muestral. En el ejemplo del lanzamiento del dado, *obtener un 2* y *obtener un nu´mero impar* son dos sucesos diferentes (*A* y *B* respectivamente). Escribiremos los sucesos as´ı: *A* = *{*2*}, B* = *{*1*,* 3*,* 5*}*.

*⊆*

##### Probabilidad Condicionada

Hasta ahora hemos calculado las probabilidades de algunos sucesos, supo- niendo que no cont´abamos con m´as informaci´on que la que proporciona la descripci´on del experimento aleatorio. Sin embargo, en muchas situaciones, au´n sin conocer el resultado del experimento, s´ı que disponemos de informa- ci´on adicional, y dicha informaci´on debe incorporarse al c´alculo de la proba- bilidad de los sucesos. Por ello, se va a introducir el concepto de probabilidad condicionada, y se establecen resultados, entre los que destaca el teorema de Bayes, que permiten incorporar informaci´on al c´alculo de probabilidades.

El conocimiento de que un determinado suceso ha ocurrido, o no, puede in- fluir en el c´alculo de las probabilidades de otros sucesos. Supongamos que extraemos, sin mirar, una carta de una baraja y debemos adivinarla. ¿La probabilidad de acertar es la misma si nos informan de que la carta extra´ıda es un oro que si no tenemos ninguna informaci´on? Parece claro que no.

**Ejemplo 1.1.** *Consideremos el experimento consistente en lanzar un dado y observar la puntuaci´on obtenida. Sean los sucesos A: obtener un nu´mero primo y B: obtener el nu´mero 1. Las probabilidades de cada uno de ellos viene dada por*

3 1

*P* (*A*) = =

6 2

1

*P* (*B*) =

6

*Supongamos ahora, que el experimento se ha realizado y que sabemos que ha salido un nu´mero par (suceso C). Con esta nueva informaci´on, las probabili- dades de los sucesos A y B se ven modificadas, ya que al saberse que se han obtenido un nu´mero par, los resultados posibles se reducen a tres (* 2*,* 4*,* 6 *), de entre los cuales s´olo uno es primo (* 2 *), por lo que la probabilidad del suceso A una vez que sabemos que ha ocurrido el suceso C es P* (*A C*) = 1*/*3*. Adem´as, como entre los actuales resultados posibles (los de C) no esta el 1 (suceso B), resulta que P* (*B C*) = 0*. Podemos escribir estas probabilidades de la siguiente manera:*

*|*

*|*

*{ }*

*{ }*

*P* (*A|C*) = 1 = 1*/*6 = *P* (*A ∩ C*)

3

1*/*2

*P* (*C*)

*P* (*B|C*) = 0 = 0 = *P* (*B ∩ C*)

*Es decir, la informaci´on proporcionada por el suceso C ha modificado las probabilidades relacionadas con los sucesos A y B.*

1*/*2

*P* (*C*)

Es decir, nos interesa la *probabilidad condicional*. Denotamos la probabilidad condicional de *Y* = *y* dado *X* = *x* como *P* (*Y* = *y X* = *x*). Esto se puede calcular como:

*|*

*P* (*Y* = *y X* = *x*) = *P* (*Y* = *y ∩ X* = *x*)

*|*

*P* (*X* = *x*)

(1)

La probabilidad condicional s´olo se define cuando *P* (*X* = *x*) *>* 0, ya que en caso contrario no es posible calcularla.

**Ejemplo 1.2.** *De una caja que contiene 4 bolas rojas y 3 blancas se extraen, sucesivamente, 2 bolas sin reemplazamiento. Queremos determinar la proba- bilidad de que la primera bola sea roja y la segunda blanca.*

*Sean los sucesos A: la primera bola es roja y B: la segunda bola es blanca.*

*Queremos calcular P* (*A ∩ B*)*. Como P* (*A*) = 4 *y P* (*B|A*) = 3 *, se tiene que*

7

6

*P* (*B|A*) = *P* (*A ∩ B*) *⇔ P* (*A∩B*) = *P* (*B|A*)*·P* (*A*) *↔ P* (*A∩B*) = 3 *·* 4 = 12 = 2

*P* (*A*)

6

7

42

7

Cualquier distribuci´on de probabilidad conjunta sobra varias variables puede descomponerse en la distribuci´on condicional sobre una de las variables. Es decir, se cumple la *regla de la cadena de la probabilidad*. Por ejemplo,

*n*

IT

*P* (*X*1 = *x*1*, · · · , Xn* = *xn*) = *P* (*X*1 = *x*1)*· P* (*Xi* = *xi|X*1 = *x*1*, · · · Xi−*1 = *xi−*1)

*i*=2

Esto se deduce de la ecuacion [1](#_bookmark4). En efecto:

*P* (*a, b, c*) = *P* (*a|b, c*)*P* (*b, c*) (2)

*P* (*b, c*) = *P* (*b|c*)*P* (*c*) (3)

*P* (*a, b, c*) = *P* (*a|b, c*)*P* (*b|c*)*P* (*c*) (4)

Como veremos, las variables condicionadas nos servir´an para plantear nues- tros problemas de Machine Learning y Deep Learning.

##### Independencia e Independencia Condicional

Dentro del tratamiento conjunto de variables aleatorias, existe un concepto que es de m´axima importancia: el concepto de *independencia*. Se dice que un conjunto de variables aleatorias es *independiente* si y s´olo si la funci´on de distribuci´on conjunta es producto de las funciones de distribuci´on marginales. Es decir,

*∀x*1 *∈ X*1*, · · · , xn ∈ Xn,*

*P* (*X*1 = *x*1*, · · · , Xn* = *xn*) = *P* (*X*1 = *x*1) *· . . . · P* (*Xn* = *xn*)

Obs´ervese que el concepto de independencia estad´ıstica que se acaba de de- finir no implica que no haya relaci´on alguna entre las variables, sino que el conocimiento de una no modifica la distribuci´on de probabilidad de las otras. Por ejemplo, sea el experimento de sacar una carta de una baraja espan˜ola y observar el palo y nu´mero de la carta que han salido. As´ı, sea la variable *X* el palo (numerando 1 a copas, 2 a oros, 3 a espadas y 4 a bastos), y la variable *Y* el nu´mero (sota es 8, caballo es 9 y rey 10). La funci´on de masa de ambas variables conjuntamente es *p*(*x, y*) = 1*/*40 *x* 1*,* 2*,* 3*,* 4*, y*

*∀ ∈ ∀ ∈*

1*, ,* 10. Las funciones marginales son *p*(*x*) = 1*/*4*, x* 1*,* 2*,* 3*,* 4 y *p*(*y*) = 1*/*10*, y* 1*, ,* 10. Evidentemente, la funci´on conjunta es producto de las marginales, luego, se puede decir que ambas variables son estad´ısticamen- te independientes. Es decir, dada una carta saber cu´al es el palo no modifica la probabilidad de cu´al ser´a su nu´mero.

*∀ ∈ · · ·*

*· · · ∀ ∈*

La definici´on anterior se puede extendir de manera que un conjunto de va- riables *X*1*, · · · , Xn* son condicionalmente independientes de otro conjunto *Y*1*, · · · , Ym*, si la distribuci´on de probabilidad condicionada sobre *X*1*, · · · , Xn* factoriza de la siguiente manera con los valores de *Y*1*, · · · , Ym*:

*∀x*1 *∈ X*1*, · · · , xn ∈ Xn, y*1 *∈ Y*1*, · · · yn ∈ Ym, P* (*X*1 = *x*1*, · · · , Xn* = *xn|Y*1*, · · · , Ym*) =

*P* (*X*1 = *x*1*|Y*1*, · · · , Ym*) *· . . . · P* (*Xn* = *xn|Y*1*, · · · , Ym*)

**Independencia y Probablidad Condicionada:** Volviendo a la defini- ci´on de probabilidad condicionada, se dice que dos sucesos *A* y *B*, asociados a un experimento aleatorio, son independientes si el hecho de que ocurra uno de ellos no influye en la probabilidad de que ocurra el otro; es decir, si

*P* (*A|B*) = *P* (*A*) o lo que es igual *P* (*B|A*) = *P* (*B*)

En otro caso, se dice que los sucesos son dependientes. La independencia de dos sucesos se puede caracteriar por la siguiente propiedad:

Los sucesos *A* y *B* son independientes si y s´olo si

*P* (*A ∩ B*) = *P* (*A*) *· P* (*B*)

**Ejemplo 1.3.** *De una caja que contiene 4 bolas rojas y 3 blancas se extraen sucesivamente 2 bolas con reemplazamiento. Queremos determinar la proba- bilidad de que primera bola sea roja y la segunda blanca.*

*Sean los sucesos A: la primera bola es roja y B: la segunda bola es blanca.*

*Queremos calcular P* (*A ∩ B*)*. Como P* (*A*) = 4 *y P* (*B|A*) = 3 *, se tiene que*

7

7

3 4 12

*P* (*A ∩ B*) = *P* (*B|A*) *· P* (*A*) *↔ P* (*A ∩ B*) = 7 *·* 7 = 49

*Es inmediato constatar que se trata de sucesos independientes. En este ejem- plo, la segunda extracci´on se efectu´a en las mismas condiciones que la prime- ra. Es decir, el resultado de la primera extracci´on no afecta al resultado de la segunda. N´otese que si las extracciones hubieran sido sin reemplazamiento, los sucesos A y B no ser´ıan independientes.*

**La regla de Bayes** Por u´ltimo, a menudo nos encontramos en una situa- ci´on donde conocemos *P* (*Y X*), pero necesitamos saber *P* (*X Y* ). Si tambi´en conocemos P(X), podemos calcular la probabilidad gracias a la **Regla de Bayes**:

*| |*

*P* (*X Y* ) = *P* (*X*)*P* (*Y |X*)

*|*

*P* (*Y* )

donde P(Y) se puede calcular como

*P* (*Y* ) = *P* (*Y |x*)*P* (*x*)

*x*

Esta regla se puede extender al caso en el que tenemos los sucesos *A*1*, A*2*, · · · Ak*,

de manera que todos tienen probabilidad positiva (*P* (*Ai*) *>* 0 *i* 1*,* 2*, , k* ). Sea *B* cualquier suceso con *P* (*B*) *>* 0. Entonces aplicando la definici´on de probabilidad condicionada, se tiene el **Teorema de Bayes**:

*∀ ∈ { · · · }*

*P* (*A |B*) = *P* (*Aj ∩ B*) = L *P* (*Aj*) *· P* (*B|Aj*)

*j*

*P* (*B*)

*k*

(*P* (*A* ) *· P* (*B|A* ))

*i*

*i*

*i*=1

No´tese que el teorema de Bayes nos proporciona una regla simple para calcu- lar las probabilidades condicionadas de cada suceso *Ai*, una vez que se sabe que el suceso *B* ha ocurrido.

**Ejemplo 1.4.** *Para producir un determinado art´ıculo se utilizan tres m´aqui- nas diferentes, M*1*, M*2*, M*3*. Del total de producci´on, el 20 % corresponde a la m´aquina M*1*; el 30 % a M*2*; y el 50 % a M*3*. Supongamos, adem´as, que el 1 % de los art´ıculos producidos por la m´aquina M*1 *es defectuoso, que el 2 % de los art´ıculos producidos por M*2 *es defectuoso y que lo mismo pasa con el 3 % de los art´ıculos producidos por M*3*. Finalmente, supongamos que en el transcurso de un control de calidad, se elige al azar un art´ıculo y resulta ser defectuoso. Queremos determinar la probabilidad de que hay sido producido por la m´aquina M*3*.*

*En primer lugar, denotamos Ai* = *{ el suceso el art´ıculo a sido producido por la m´aquina Mi}, con i* = 1*,* 2*,* 3*. Sea B el suceso { el art´ıculo selec- cionado es defectuoso , debemos determinar P* (*A*3 *B*)*. Es decir, tenemos que determinar la probabilidad de que el art´ıculo haya sido producido en la m´aquina M*3*, sabiendo que es defectuoso. Por el enunciado sabemos que P* (*A*1) = 0*.*2; *P* (*A*2) = 0*.*3; *P* (*A*3) = 0*.*5 *son las probabilidades de que el art´ıculo seleccionado haya sido producido en cada una de las m´aquinas. Tam- bi´en nos dicen que P* (*B A*1) = 0*.*01; *P* (*B A*2) = 0*.*02; *P* (*B A*3) = 0*.*03 *son las probabilidades e que el art´ıculo producido en cada m´aquina sea defectuoso. De este modo se tiene que*

*} |*

*| | |*

*P* (*B*) = *P* (*A*1) *· P* (*B|A*1) + *P* (*A*2) *· P* (*B|A*2) + *P* (*A*3) *· P* (*B|A*3) =

= 0*.*2 *·* 0*.*01 + 0*.*3 *·* 0*.*02 + 0*.*5 *·* 0*.*03 = 0*.*023

*Ahora, por el teorema de Bayes, tenemos que*

*P* (*A |B*) = *P* (*A*3) *· P* (*B|A*3) = 0*.*5 *·* 0*.*03 = 0*.*65217

3

*P* (*B*)

0*.*023

En el ejemplo anterior, a una probabilidad como *P* (*A*3) se la suele llamar *probabilidad a priori* de que el art´ıculo seleccionado haya sido producido por la m´aquina *M*3, ya que *P* (*A*3) es la probabilidad de este suceso antes de que el art´ıculo sea seleccionado, y antes de que se sepa si el art´ıculo es o no de- fectuoso.

Por otra parte, a una probabilidad como *P* (*A*3 *B*) se le llama *probabilidad a posteriori* de que el art´ıculo seleccionado haya sido producido por la m´aqui- na *M*3, ya que es la probabilidad del suceso *A*3 despu´es de conocer que el art´ıculo es defectuoso.

*|*

El teorema de Bayes, junto con las funciones multidimensionales son, de los conceptos aqu´ı, presentados, los que m´as utilizaremos en lo que queda del curso o a la hora de plantear un modelo de Machine Learning (ML). Intui- tivamente, dispondremos de varias muestras que ser´an variables aleatorias y querremos inferir (obtener) su distribuci´on real. Gracias a la regla de bayes, a partir de los datos que conocemos podremos plantear un modelo que nos d´e la mejor aproximaci´on a la funci´on. Es decir, buscaremos la distribuci´on de funciones (v.a) que, con mucha probabilidad, haya generado la salida Y con las entradas X.

#### Variables Aleatorias

Una *variable aleatoria (v.a) X* es una funci´on que asigna un valor *X*(*s*) a cada resultado del experimento *s S*. As´ı, por ejemplo, si al resultado del lanzamiento de una moneda le asignamos el valor 1 si es cara y -1 si es cruz, tendremos una variable aleatoria cuyos posibles valores es el conjunto *{*1,-1*}*.

*∈*

**Ejemplo 1.5.** *Sea el experimento consistente en lanzar dos dados, el espacio muestral S est´a formado por todos los pares* (*i, j*)*, con i, j* = 1*,* 2*,* 3*,* 4*,* 5*,* 6*; en total, 36 pares. Se considera la variable aleatoria X: suma de las puntua- ciones observadas en cada lanzamiento definida en S y con valores en los nu´meros reales: X*(*i, j*) = *i* + *j.*

**Ejemplo 1.6.** *Considerar el experimento consistente en elegir una persona adulta de una determinada poblaci´on para medir su altura, X, en cent´ımetros. La variable aleatoria X podr´a tomar cualquier valor, digamos entre 150 y 220 cm.*

**Ejemplo 1.7.** *En una empresa que fabrica l´amparas se selecciona al azar una y se mide su tiempo de vida X. Esta variable aleatoria puede tomar cualquier valor del intervalo* [0*,* +*∞*)*.*

Las variables aleatorias son de gran importancia pues trasladan la probabi- lidad de un espacio no num´erico a uno num´erico, con todas las ventajas que ello conlleva, entre otras, las caracterizaciones y propiedades que se ver´an a continuaci´on. Las variables aleatorias puede ser discretas o continuas. Ambas variables se definen formalmente m´as adelante.

Habitualmente, se nota a las variables aleatorias con letra mayu´scula y a los valores muestrales de dicha variable con letras minu´sculas. As´ı, por ejemplo, se habla de la variable *X* y de las muestras *xi*.

**Funciones de probabilidad y Funciones de distribucio´n** Para asig- nar a cada valor una probabilidad de que ocurra, se utilizar´an las *funciones de probabilidad*, que se conocen como *funciones de masa* o *funciones de den- sidad*, en funci´on de si la variables es discreta o continua. A continuaci´on se detallar´an m´as estas funciones.

Si se ha definido una probabilidad, P, sobre el espacio muestra, *S*, asociado a un experimento aleatorio, podemos determinar la distribuci´on de probabi- lidad de los posibles valores que toma cualquier variable aleatoria *X* definida sobre *S*. Sea *A* cualquier subconjunto de la recta real.

**Ejemplo 1.8.** *Consideramos el experimento consistente en lanzar tres mo- nedas equilibradas (o una moneda tres veces). Hay 8 resultados posibles*

*{C, C, C}, {C, C, X}, {C, X, C}, {X, C, C},*

*{C, X, X}, {X, C, X}, {X, X, C}, {X, X, X}*

*todos ellos con la misma probabilidad. Por tanto todos tienen probabilidad*

1*/*8*.*

*Sea X la variable aleatoria nu´mero de caras que se han obtenido, entonces se tiene que*

1 1 1 3

*P* (*X* = 1) = *P{CXX, XCX, XXC}* = 8 + 8 + 8 = 8

*An´alogamente se calcular´ıa la funci´on de masa como:*

*p*(*x*) = 

****

****

|  |  |  |
| --- | --- | --- |
| 1*/*8 | *si* | *x* = 0 |
| 3*/*8 | *si* | *x* = 1 |
| 3*/*8 | *si* | *x* = 2 |
| 1*/*8 | *si* | *x* = 3 |

*Adem´as, si se quiere observar la probabilidad del suceso A: obtener al menos dos caras, basta observar que*

3 1 1

*P* (*A*) = *P* (*X ≥* 2) = *P* (*X* = 2) + *P* (*X* = 3) = 8 + 8 = 2

En general, para asignar probabilidades a subconjuntos de la recta real, basta asignar probabilidades a los intervalos de la forma ( *, x*], para todo nu´mero real x. Es decir, para asignar a cada suceso la probabilidad de que la variable aleatoria, *X*, tome un valor igual o inferior a uno dado, *x*, se utilizar´a una *funci´on de distribuci´on F* (*x*):

*−∞*

*F* (*x*) = *P* ((*−∞, x*]) = *P* (*X ≤ x*)

Se trata de una funci´on que acumula la probabilidad de todos los valroes de la variable aleatoria que son menores o iguales que el valor actual *x*. Las propiedades de esta funci´on son las siguientes:

Su valor est´a comprendido entre 0 y 1, es decir, 0 *≤ F* (*x*) *≤* 1 Es no decreciente

En los l´ımites, se cumple l´ım

*x→*+*∞*

*F* (*x*) = 1 y l´ım

*x→−∞*

*F* (*x*) = 0

**Ejemplo 1.9.** *La funci´on de distribuci´on del ejemplo anterior se obtiene de la siguiente manera: como los u´nicos valores de la variable que tienen la pro- babilidad estrictamente positiva son 0,1,2 y 3, dividimos la recta real en los siguientes intervalos:*

(*−∞,* 0) [0*,* 1) [1*,* 2) [2*,* 3) [3*,* +*∞*)

*y acumulamos la probabilidad en cada uno de ellos; por ejemplo, si x* [1*,* 2)*, resulta que*

*∈*

1 3 1

*F* (*x*) = *P* (*X ≤ x*) = *P* (*X* = 0) + *P* (*X* = 1) = 8 + 8 = 2

*De forma an´aloga se procede con los dem´as intervalos, con lo que la funci´on de distribuci´on de la variable aleatoria X es*

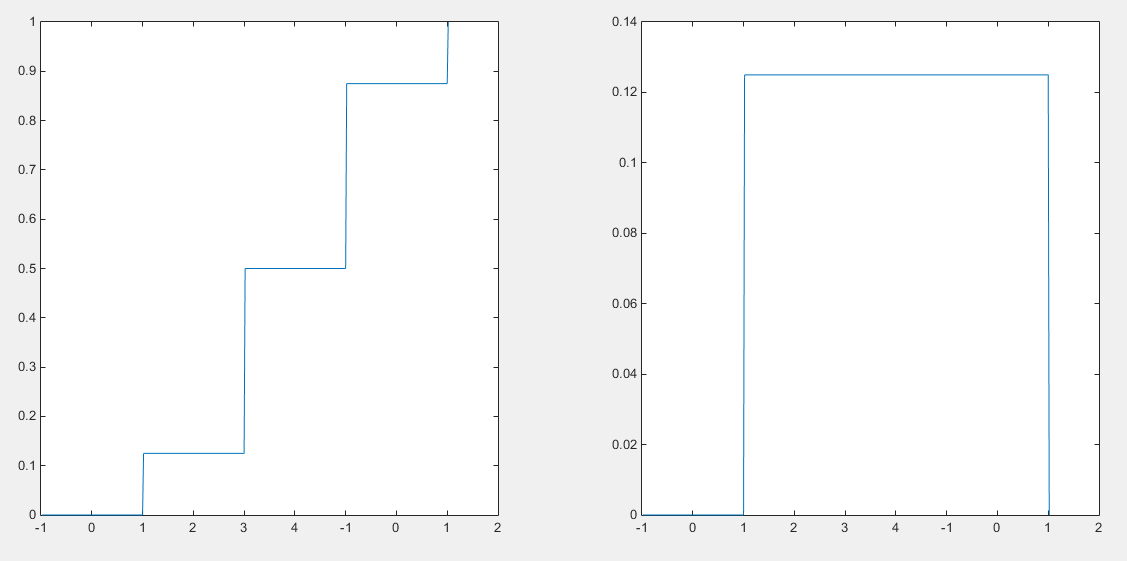


Figura 1: Funci´on de masa del ejercicio 1.5 (derecha) y funci´on de distribuci´on (izquierda)

*F* (*x*) =

1. *si x <* 0 1*/*8 *si* 0 *≤ x <* 1 4*/*8 *si* 1 *≤ x <* 2 7*/*8 *si* 2 *≤ x <* 3
2. *si* 3 *≤ x*



****

****

*Y el resultado ser´an las gr´aficas de la figura* [*1*](#_bookmark8)*:*

**Variables Discretas** Una *variable aleatoria discreta* es aquella que tiene un nu´mero finito (o infinito numerable) de valores. N´otese que estos valores no son necesariamente enteros, pueden incluso no ser num´ericos. Hasta aho- ra, en todos los ejemplos mostrados las variables eran de tipo discreto.

Si el experimento tiene un nu´mero finito de resultados posibles; es decir, si el espacio muestral es de la forma *S* = *{s*1*, s*2*, · · · , sn}*, para construir una probabilidad basta con asignar probabilidades *p*1*, p*2*, , pn*, con *pi* = *P* (*si*). Esta funci´on *P* se conoce como *funci´on de masa de probabilidad* y debe cumplir:

*· · ·*

El dominio de la funci´on de masa debe ser el conjunto de todos los posibles estados de la variable aleatoria *X*.

*x X,* 0 *p*(*x*) 1. Es decir, los valores de estas funciones han de estar entre 0 y 1 ya que son probabilidades.

*∀ ∈ ≤ ≤*

L*x∈X p*(*x*) = 1. Es decir, su suma ha de ser 1.

De este modo, la probabilidad de cualquier suceso *A ⊂ S* se calcular´a como:

*P* (*A*) = *pi*

*si∈A*

**Ejemplo 1.10.** *Consideremos nuevamente el experimento consistente en lanzar un dado y observar el resultado. Como S* = 1*,* 2*,* 3*,* 4*,* 5*,* 6 *, pode- mos asignar probabilidades de la siguiente manera:*

*{ }*

*P* (1) = *P* (2) = 1 *P* (3) = *P* (4) = *P* (5) = *P* (6) = 1

4

8

*En este caso, las probabilidades de los sucesos A: obtener un nu´mero mu´ltiplo de 3 y B: obtener nu´mero primo son:*

*P* (*A*) = *P* (3) + *P* (6) = 1 *P* (*B*) = *P* (1) + *P* (2) + *P* (5) = 3

4

4

Por otra parte, la funci´on de distribuci´on *F* (*x*) de este tipo de variable se calcula como la suma de la funci´on de probabilidad para valores iguales o inferiores al valor *x*, y resulta por lo tanto una funci´on escalonada:

*F* (*x*) = *P* (*x* = *xi*); *− ∞ < x <* +*∞*

*xi≤x*

**Ejemplo 1.11.** *De nuevo en el ejemplo del dado, sea X es la variable aleato- ria nu´mero del dado y sea el suceso A: obtener un nu´mero menor o igual que*

*3, sabiendo que la funci´on de masa es p*(*x*) = 1 *∀x ∈ {*1*,* 2*,* 3*,* 4*,* 5*,* 6*}, para*

6

*calcular la funci´on de distribuci´on dividimos la recta real en los intervalos:*

(*−∞,* 1) [1*,* 2) [2*,* 3) [3*,* 4) [4*,* 5) [5*,* 6) [6*,* +*∞*)

*y acumulamos la probabilidad en cada uno de ellos; por ejemplo, si x* [3*,* 4)*, resulta que*

*∈*

1 1 1 1

*F* (*x*) = *P* (*X ≤ x*) = *P* (*X* = 1) + *P* (*X* = 2) + *P* (*X* = 3) = 6 + 6 + 6 = 2

*De forma an´aloga se procede con los dem´as intervalos, con lo que la funci´on de distribuci´on de la variable aleatoria X es*

*F* (*x*) =

1. *si x <* 1 1*/*6 *si* 1 *≤ x <* 2 2*/*6 *si* 2 *≤ x <* 3 3*/*6 *si* 3 *≤ x <* 5 4*/*6 *si* 4 *≤ x <* 6 5*/*6 *si* 5 *≤ x <* 6
2. *si* 6 *≤ x*

****

****

****

*De este modo, la probabilidad del suceso A ser´a:*

*F* (3) = *P* (*X* = *x*) = *P* (*X* = 1) + *P* (*X* = 2) + *P* (*X* = 3) = 1

2

*x≤*3

**Variables Continuas** El concepto de variable aleatoria discreta permite modelizar un gran nu´mero de situaciones. Sin embargo, existen otras muchas en las que el modelo discreto no es el adecuado.

Por ejemplo, suponemos que para ir a su trabajo una persona debe coger todos los d´ıas el tren, y en la franja horaria en la que llega a la estaci´on los trenes pasan con una frecuencia de 10 minutos. Suponiendo que dicha persona llega de forma aleatoria, el tiempo de espera en la estaci´on es una variable aleatoria continua *X*, ya que el tiempo es una variable continua.

Formalmente, las *variables continuas* son aquellas que toman cualquier valor de los nu´meros reales. En este caso, para asignar a cada posible valor su probabilidad de que ocurra se utiliza la *funci´on de densidad de probabilidad*. Formalmente, se dice que una variable es continua cuando existe una funci´on, denominada *funci´on de densidad, f(x)*, tal que su integral para un conjunto *I* determina la probabilidad de ocurrencia de dicho conjunto:

*P* (*x ∈ I*) = *f* (*x*)*dx*

*I*

Ana´logamente, esta funci´on cumple las propiedades:

El dominio *I* de *p* debe ser el conjunto de todos los posibles valores de

*X*.

*∀x ∈ X, f* (*x*) *≥* 0.

J

*f* (*x*)*dx* = 1. Es decir, la integral de la funci´on de densidad en todo el espacio ha de ser la unidad.

**Ejemplo 1.12.** *La funci´on*

*f* (*x*) =

1(4 *− x*) 0 *< x <* 4

0 *en el resto*

8

*es la funci´on de densidad de alguna variable aleatoria X, puesto que es una funci´on no negativa y*

+*∞* 1 4

*f* (*x*)*dx* =

*−∞* 8 0

(4 *− x*)*dx* = 1

La funci´on de distribuci´on en este caso se calcula como la integral de la fun- ci´on de densidad en el intervalo ( *, x*], y de aqu´ı que su derivada constituye la funci´on de densidad asociada a la variable *x*. La funci´on de distribuci´on en este caso es continua en todos los puntos, y es derivable en todos excepto a lo sumo en una cantidad numerable de ellos:

*−∞*

*F* (*x*) = *P* (*x ∈* [*−, ∞, x*])) =

*F ′*(*x*) = *f* (*x*)

*f* (*y*)*dy* ; *−∞ ≤ x ≤* +*∞*

*−∞*

*x*

**Ejemplo 1.13.** *La funci´on de distribuci´on de la variable aleatoria del ejem- plo anterior se clacula de la siguiente manera:*

*F* (*x*) =

**** 1

0 *x <* 0

*x*

1

(4 *− t*)*dt* = (8*x − x*2) *si* 0 *≤ x <* 4

**** 0

8

16

1 *si x ≥* 4

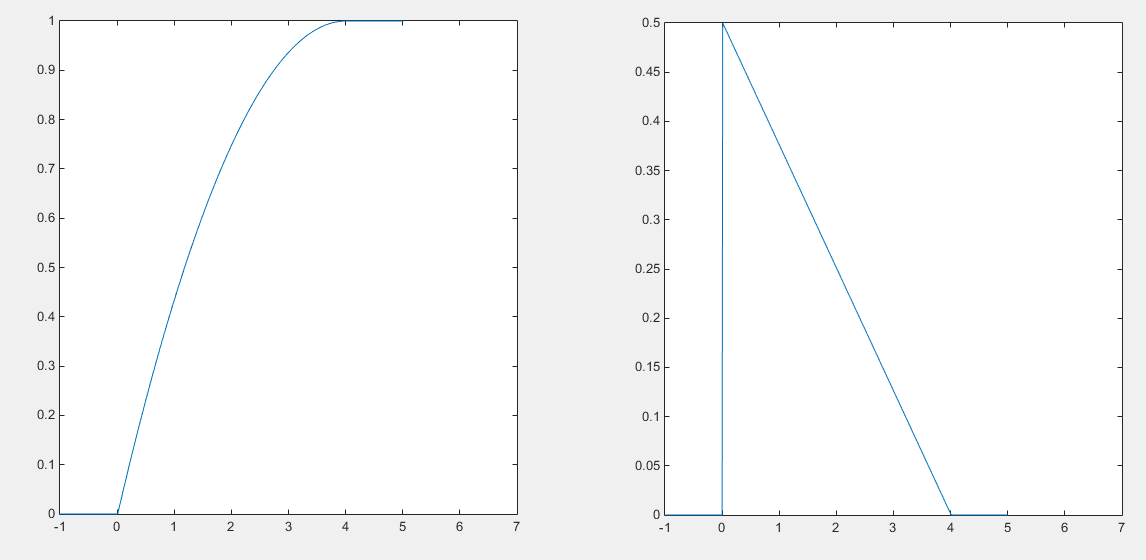


Figura 2: Funci´on de densidad (derecha) y funci´on de distribuci´on (izquierda) del ejem 1.11

*N´otese en la figura* [*2*](#_bookmark9) *que ahora las funciones no son escalonadas, si no que crecen de manera suave.*

*Por otra parte, generar valores de las variables aleatorias continuas no es tan trivial y requieren de m´etodos (como el de la funci´on inversa) que no se ver´an en este curso.*

**Esperanza de un v.a** A continuaci´on se presentan unas de las carac- ter´ısticas m´as importantes de las variables aleatorias son sus medidas de centralizaci´on (media) y sus medidas de dispersi´on (varianza):

La *esperanza* o valor esperado o media de una variable aleatoria *X*, denotada por *µ* o *E*[*X*], es una medida central del comportamiento de la variable que representa su centro de masas. Se puede interpretar como el *valor medio que tomar´ıa la variable en una larga serie de observaciones*. Este valor se calcula de manera distinta dependiendo de si se trata de una variable discreta o continua:

*E*[*X*] = *µ* =

****

****

*xip*(*xi*) siendo *Xi* variable discreta

*xi*

*xf* (*x*)*dx* siendo *Xi* variable continua

R

La esperanza por su definici´on es un operador lineal, es decir

*n*

*E*

*i*=1

*ciXi*

*n*

=

*i*=1

*ciE*[*Xi*]

**Ejemplo 1.14.** *Sea X una variable aleatoria discreta cuyos valores y sus probabilidades respectivas vienen dados en la tabla* [*1*](#_bookmark10)

*xi* -2 -1 1 2 4

*pi* 0.1 0.25 0.15 0.4 0.1

Cuadro 1: Funci´on de masa de *X*

*La esperanza matem´atica de la variable vendr´a dada por:*

5

*E*[*X*] = *xipi* = *−*0*.*2 *−* 0*.*25 + 0*.*15 + 0*.*8 + 0*.*4 = 0*.*9

*i*=1

*Con Matlab se puede calcular con el comando mean(x), como se puede ver en el siguiente c´odigo:*

function y=genero\_discreta(m) y=[]

for i=1:m

u=rand(1,1); if u<0.1

y(i)=-2;

elseif u<0.35

y(i)=-1;

elseif u<0.5

y(i)=1;

elseif u<0.9

y(i)=2;

else

y(i)=4;

end

end

end

%%%LLAMADA%%%

%Genero 100000 valores de la variable aleatoria: x=genero\_discreta(100000);

%compruebo que est´a bien generada tabulate(x)

%RESULTADO

|  |  |  |  |
| --- | --- | --- | --- |
| % | Value | Count | Percent |
| % | -2 | 9932 | 9.93% |
| % | -1 | 25083 | 25.08% |
| % | 1 | 14957 | 14.96% |
| % | 2 | 40009 | 40.01% |
| % | 4 | 10019 | 10.02% |

%Calculo la media media=mean(x)

%RESULTADO

%media =

%

% 0.9055

*Alternativamente, con Python se puede calcular con el comando mean(x), de numpy, como se puede ver en el siguiente c´odigo:*

import random import numpy as np import pandas as pd

def genero\_discreta(m): y=[]

for i in range(m):

u=random.random() if u<0.1:

y+=[-2]

elif u<0.35:

y+=[-1]

elif u<0.5:

y+=[1]

elif u<0.9:

y+=[2]

else:

y+=[4]

return pd.Series(y) #LLAMADA

#Genero 100000 valores de la variable aleatoria: x=genero\_discreta(100000); tab=x.value\_counts(sort=True)

print(tab) tab\_pct=tab/tab.sum() print(tab\_pct)

#compruebo que est´a bien generada #tabulate(x)

#RESULTADO

|  |  |  |  |
| --- | --- | --- | --- |
| # | Value | Count | Percent |
| # | -2 | 0.10055 | 10.05% |
| # | -1 | 0.24948 | 24.95% |
| # | 1 | 0.14991 | 14.99% |
| # | 2 | 0.40086 | 40.08% |
| # | 4 | 0.09920 | 9.92% |

#Calculo la media media=np.mean(np.array(x)) print("media",media) #RESULTADO

#media = #

# 0.89.79

**Ejemplo 1.15.** *Sea X una variable aleatoria con funci´on de densidad*

1 (*x* + 1) *−*1 *≤ x ≤* 3

0 *en el resto*

*f* (*x*) =

8

*La esperanza de dicha variable vendr´a dada por*

*µ* = *E*[*X*] =

1 3

*x*(*x* + 1)*dx* =

1 *x*3

*x*2 3 5

+ =

8 *−*1

8 3 2 *−*1 3

**Varianza de una v.a** La **varianza** de una variable aleatoria es una medida para determinar el nivel de dispersi´on que tienen los valores de la variable respecto de su media. Se nota por var(X) o *σ*2, y se define como la media de las desviaciones a la esperanza al cuadrado. As´ı, su definici´on y un m´etodo alternativo de c´alculo son:

*σ*2 = *E*[(*Xi − µi*)2] = *E*[*X*2] *− µ*2

*i*

*i*

Tambi´en habr´a dos modos de calcularla en funci´on del tipo de variable:

*V* [*X*] = *σ*2 =

****

****

(*xi − µ*)2*p*(*xi*) siendo *Xi* variable discreta

*xi*

(*x µ*)2*f* (*x*)*dx* siendo *Xi* variable continua

*−*

R

Por definici´on, la varianza es siempre no negativa y es un operador cuadr´atico, de modo que

*var*(*cX*) = *c*2*var*(*X*)

Obs´eserve que la varianza va en unidades al cuadrado; para dar una medi- da relativa a las unidades que se est´an manejando se utiliza la **desviaci´on t´ıpica o est´andar**, que se define como la ra´ız cuadrada de la varianza de la variable, y se denota por *σ*.

**Ejemplo 1.16.** *Consideremos el experimento del ejemplo* [*1.8*](#_bookmark7)*. Por el ejemplo anterior sabemos que µ* = *E*[*X*] = 3*/*2*. Dado que*

*σ*2 = *E*[(*Xi − µi*)2] = *E*[*X*2] *− µ*2

*i*

*i*

*para calcular la varianza s´olo tenemos que calcular E*[*X*2]*. Como*

4

*E*[*X*2] = *x*2*p*

*i*

*i*

*i*=1

= 02 1 + 12 3 + 22 3 + 32 1 = 24 = 3

*Se tiene que σ*2 = 3 *−* (3*/*2)2 = 0*.*75*.*

8

8

8

8

8

*De nuevo, esto se puede calcular con Matlab, en este caso mediante el co- mando var(x). N´otese que este comando devuelve la pseudovarianza es decir:*

*S*2 = 1 (*x*

2

*x*

*n −* 1

*i*

*i*=1

*— x*¯)

*Por tanto, para obtener la varianza, bastar´a realizar la trasformaci´on:*

2

*n*

*i*

*σ*2 = *n −* 1 *S*2 = *n −* 1 *·* 1 (*x*

*x*

*n*

*x*

*n*

*n −* 1

*i*

*i*=1

*— x*¯)2 = 1 (*x*

*i*=1

*— x*¯)

*El c´odigo correspondiente con Matlab ser´a:*

function y=genero\_discreta2(m) y=[]

for i=1:m

u=rand(1,1); if u<(1/8)

y(i)=-0;

elseif u<(4/8) y(i)=1;

elseif u<(7/8) y(i)=2;

else

y(i)=3;

end

end

end

%%%%LLAMADA%%

%Genero valores de f(x)

%Genero 100000 valores de la variable aleatoria: x=genero\_discreta2(100000);

media=mean(x) pseudovarianza=var(x)

varianza=pseudovarianza\*99999/100000

%RESULTADO

%media =

%

% 1.4975

%

%

%pseudovarianza =

%

% 0.7529

%varianza =

%

% 0.7529

*El c´odigo correspondiente con Python ser´a:*

import random import numpy as np import pandas as pd

def genero\_discreta2(m): y=[]

for i in range(m):

u=random.random() if u<(1/8):

y+=[0]

elif u<(4/8): y+=[1]

elif u<(7/8): y+=[2]

else:

y+=[3]

return np.array(y) #LLAMADA

#Genero 100000 valores de la variable aleatoria: x=genero\_discreta2(100000);

#Calculo la media media=np.mean(x) pseudovarianza=np.var(x)

varianza=pseudovarianza\*99999/100000 print("media",media) print("pseudovarianza",pseudovarianza) print("varianza",varianza)

#RESULTADO

#media 1.49963

#pseudovarianza 0.7532198631000001

#varianza 0.7532123309013692

**Ejemplo 1.17.** *Sea X una variable aleatoria con una distribuci´on continua cuya funci´on de densidad viene dada por:*

*Como*

*f* (*x*) =







*x/*9 0 *≤ x ≤* 3

(6 *− x*)*/*9 3 *< x ≤* 6

0 *x ≥* 6

*E*[*X*] =

1 3

*x*2*dx* +

1 6

*x*(6 *− x*)*dx* =

9 0

1 *x*3 3 1

9 3

*x*3 6

= +

9 3 0 9

y

3*x*2

3

*−*

= 1 + (12 *−* 8 *−* 3 + 1) = 3

3

*E*[*X*2] =

1 3

*x*3*dx* +

6

(6 *− x*)*x*2*dx* =

9 9 21

+ 48 *−* 36 *−* 6 + =

9 0 3 4 4 2

resulta que *V* (*X*) = 21*/*2 *−* 9 = 3*/*2

##### Variables Multimensionales

Si las variables, en lugar de estar definidas en R, est´an definidas en R*n*, se denominan variables multidimensionales. Las definiciones anteriores ahora pueden ser conjuntas o marginales, obten´ıendose las siguientes definiciones:

**Probabilidad conjunta** Las definiciones de probabilidad conjunta son una extensi´on de las definiciones anteriores.

En efecto, la *funci´on de distribuci´on conjunta* es aquella que asigna a ca- da conjunto la probabilidad de que las variables aleatorias, *X*1*, X*2*, · · · , Xn*, tomen un valor igual o inferior a unos valores dados, *x*1*, x*2*, · · · , xn*. Es decir

*F*(*X*1*,··· ,Xn*) = *P* (*X*1 *≤ x*1*, · · · , Xn ≤ xn*)

Ana´logamente, la **funci´on de probabilidad conjunta** asigna a cada con- junto de sucesos su probabilidad. De este modo se definir´a la **funci´on de masa conjunta** como

*p*(*X*1*,··· ,Xn*)(*x*1*, · · · , xn*) = *P* (*X*1 = *x*1*, · · · , Xn* = *xn*)

Y la **funci´on de densidad conjunta** como

*f*(*X*1*,··· ,Xn*)(*x*1*, · · · , xn*)

**Probabilidad Marginal** Alguna veces, conocemos la funci´on de distribu- ci´on sobre el conjunto de variables, pero queremos saber la distribuci´on sobre un subconjunto de ellas. La distribuci´on de probabilidad sobre un subcon- junto se conoce como *distribuci´on marginal de probabilidad*.

Por ejemplo, supongamos que tenemos las variables discretas *X* e *Y* , y co- nocemos *P* (*X, Y* ). Se puede conocer P(X) con la ”regla de la suma”:

*∀x ∈ X, P* (*X* = *x*) = *P* (*X* = *x, Y* = *y*)

*y*

Ana´logamente, para variables continuas se calcular´ıa como:

*f* (*x*) = *P* (*X* = *x*) = *f* (*x, y*)*dy*

**Ejemplo 1.18.** *Un dado y una moneda equilibrados son lanzados. Sean X= resultado obtenido en el lanzamiento del dado e Y* = 0 *si se obtiene la cruz, y 1 si se obtiene cara en el lanzamiento de la moneda. Tomamos D*(*X,Y* ) = *{*(*i, j*) : *i ∈ {*1*, · · · ,* 6*}, j ∈ {*0*,* 1*}} y pij* = *P* (*X* = *i, Y* = *j*) = 1 *,* (*i, j*) *∈ D*(*X,Y* )*. Entonces la distribuci´on conjunta viene dada por,*

*{ }*

12

*F* (*x, y*) =

0 *x <* 1 *o y <* 0

1*/*12 (1 *≤ x <* 2*,* 0 *≤ y <* 1)

****

****

1*/*6 (2 *≤ x <* 3*,* 0 *≤ y <* 1) *o* (1 *≤ x <* 2*, y ≥* 1)

1*/*4 (3 *≤ x <* 4*,* 0 *≤ y <* 1)

****

1*/*3 (4 *≤ x <* 5*,* 0 *≤ y <* 1) *o* (2 *≤ x <* 1*, y ≥* 1)

5*/*12 (5 *≤ x <* 6*,* 0 *≤ y <* 1)

****

1*/*2 (6 *≤ x,* 0 *≤ y <* 1) *o* (3 *≤ x <* 4*, y ≥* 1)

2*/*3 (4 *≤ x <* 5*, y ≥* 1)

****

5*/*6 (5 *≤ x <* 6*, y ≥* 1)

1 (6 *≤ x, y ≥* 1)

*Ahora, para calcular las distribuciones marginales, tenemos en cuenta que*

*FX*(*x*) = *F* (*X ≤ x, Y* = 0) + *F* (*X ≤ x, Y* = 1)

*FY* (*y*) = *F* (*X* = 1*, Y ≤ y*) + *F* (*X* = 2*, Y ≤ y*) + *F* (*X* = 3*, Y ≤ y*)+

+*F* (*X* = 4*, Y ≤ y*) + *F* (*X* = 5*, Y ≤ y*) + *F* (*X* = 6*, Y ≤ y*)

*As´ı, representando la funci´on de masa mediante la siguiente tabla es f´acil calcular las probabilidades marginales:*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *X:* | *1* | *2* | *3* | *4* | *5* | *6* | *p•j* |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Y:* | *0* | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  2 |
|  | *1* | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  12 | 1  2 |

*pi•*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 |

*Es decir,*

*FX*(*x*) =

0 *x <* 1

1*/*6 1 *≤ x <* 2

****

****

2*/*6 2 *≤ x <* 3

3*/*6 3 *≤ x <* 4

4*/*6 4 *x <* 5

*≤*

5*/*6 5 *≤ x <* 6

****

*FY* (*y*) =







0 *y <* 0

1*/*2 0 *≤ y <* 1

1 *y ≥* 1

 1 *x ≥* 6

**Ejemplo 1.19.** *Sea (X,Y) un vector aleatorio con funci´on de densidad*

*e−*(*x*+*y*) *x >* 0*, y >* 0

*Entonces*

*f* (*x, y*) =

*x y*

0 *en caso contrario*

*F*(*X,Y* ) =

*f* (*u, v*)*dvdu*

*x y*

*−∞*

*−∞*

=

*e−*(*u*+*v*)*dvdu* =

0 0

= (1 *− e−x*)(1 *− e−y*) *si x <* 0*, y <* 0; *y 0 en caso contrario*

*Ahora, las funciones de densidad marginales se calcular´an como*

*F*(*X*) =

=

*f* (*u, v*)*du*

*x −*

*−∞*

*x*

*e*

(*u*+*y*)*du* =

0

= (1 *− e−x*) *si x <* 0*, y 0 en caso contrario*

*F*(*X*) =

=

*y*

*f* (*u, v*)*dv*

*y −*

*−∞*

*e*

(*c*+*v*)*dv* =

0

= (1 *− e−y*) *si y <* 0*, y 0 en caso contrario*

#### Distribuciones de Probabilidad.

Antes de comenzar la pr´actica vamos a describir algunas de las distribuciones m´as importantes. Adem´as, se ver´a con Matlab como obtener y dibujar las funciones de probabilidad, las funciones de distribuci´on. Asimismo, c´omo calcular la media y la varianza y como generar datos aleatorios de dichas distribuciones.

##### Distribuciones Discretas

**Bernouilli(p)** : La distribuci´on de Bernouilli es una distribuci´on sobre

una

u´nica variable binaria. Se caracteriza por un

u´nico par´ametro *p* que

representa la probabilidad de que la variable aleatoria tome el valor 1. Por ello, esta distribuci´on se utiliza para representar el resultado de experimentos con 2 posibles estados. Es decir, esta distribuci´on se caracteriza por:

Rango:*{*0*,* 1*}* Media: *p*.

Varianza: *p*(1 *− p*)

Par´ametros: *p ≡* probabilidad del evento 1.

*−*

Funci´on de masa: *p*(*x*) = 1 *p x* = 0

*p x* = 1

**Binomial(n,p)** : La distribuci´on de Binomial se define como la suma de *n* variables de Bernouilli. Es decir, en este caso representa que el experimento estudiado se ha repetido *n* veces de manera independiente, y se quiere conocer el nu´mero de ”´exitos”. En concreto, estos ”´exitos”podr´ıan ser el nu´mero de elementos defectuoso de un lote de taman˜o *n*. Se caracteriza por los par´ame- tro *p*, que representa la probabilidad de que la variable aleatoria tome el valor 1; y por el par´ametro *n*, que representa el nu´mero de veces que se ha repetido el experimento. Es decir, esta distribuci´on se caracteriza por:

Rango: *{*0*,* 1*, · · · , n}*

Media: *np*. Varianza: *np*(1 *− p*)

Par´ametros: *p* probabilidad del evento ; *n*, nu´mero de veces que se repite el experimento.

*≡*

Funci´on de masa:

*p*(*x*) =

 *n* *px*(1 *− p*)*n−x x ∈* 0*,* 1*, · · · n*

 0 en otro caso

*x*

Con las funciones binopdf, binocdf,binostat y binornd se puede calcular las funciones de masa, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on binomial como se muestra en el siguiente c´odigo de Matlab:

x=0:0.1:10;

%Binonimal

%%%%%%%

%parametros n=10; p=0.2;

%title(’Funci´on de masa Binomial’) y\_masa\_bin=binopdf(x,n,p);

%title(’Funci´on de distribuci´on Binomial’) y\_dist\_bin=binocdf(x,n,p);

%graficas f6=figure; figure(f6); ax=subplot(1,2,1);

plot(x,y\_masa\_bin,’\*’) ax=subplot(1,2,2); plot(x,y\_dist\_bin,’\*’)

% estad´ısticos [m,v]=binostat(n,p);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas x\_bino= binornd(n,p,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´an:

from scipy.stats import binom import matplotlib.pyplot as plt import random

import numpy as np x=np.arange(0,10,0.1) n=10

p=0.2

y\_masa\_bin=binom.pmf(x,n,p) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa binomial") plt.show()

y\_dist\_bin=binom.cdf(x,n,p) plt.plot(x,y\_dist\_bin,’bo’) plt.title("Funci´on de distribuci´on binomial") plt.show()

#Estad´ısticos mean,var=binom.stats(n,p)

print("media",mean,"y varianza, ", var)

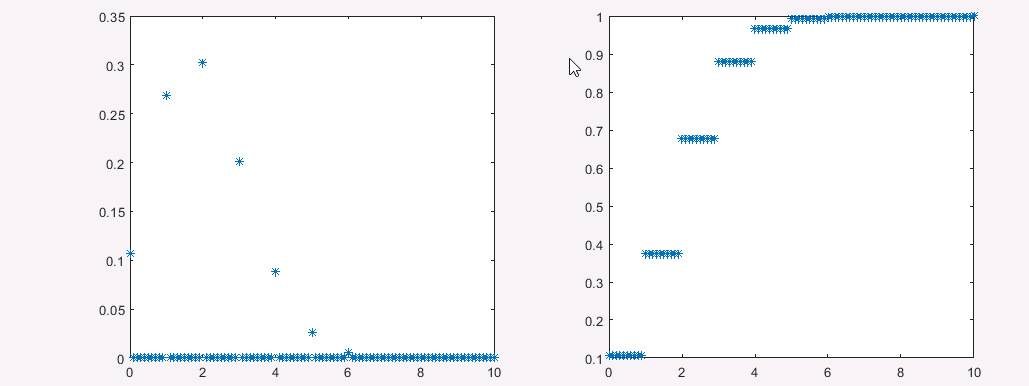


Figura 3: Funci´on de masa (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Binomial(10,0.2)

#Generar Valores r=np.random.binomial(n,p,size=1000)

Las funciones de masa y de distribuci´on obtenidas se pueden ver en [3](#_bookmark14). N´otese que para generar valores de una bernouilli, basta hacer n=1.

**Ejemplo 1.20.** *Supongamos que en una determinada poblaci´on, la probabi- lidad de que un beb´e reci´en nacido sea nin˜a es de 0.48.*

*El estudio de la variable X* =*nu´mero de nin˜as entre n beb´es reci´en naciones en esa poblaci´on, puede modelizarse mediante una distribuci´on binomial, en la que cada nacimiento sea considerado como una prueba de Bernoulli y el suceso ´exito sea A* =*nacer nin˜a. Es decir, X ≡ Binom*(*n,* 0*.*48)

*De esta forma, admitiendo que la probabilidad p de nacer nin˜as se mantiene constante de un nacimiento a otro, se podr´an determinar f´acilmente proba- bilidades como la de que en 5 nacimientos de esa poblaci´on elegidos al azar haya 2 nin˜as. E´sta probabilidad ser´a:*

*P* (*X* = 2) = 5 (0*.*48)2(0*.*52)3 = 0*.*324

2

**Geom´etrica(p)** : La distribuci´on geom´etrica tambi´en se basa en la dis- tribuci´on de Bernouilli. En este caso representa el nu´mero de repeticiones necesarias hasta obtener el primer ”´exito”. Un ejemplo ser´a el nu´mero de pro- dutos inspeccionados hasta encontrar el primer defectuoso. Esta distribuci´on tiene como caracter´ıstica la falta de memoria. De este modo, se caracteriza por la probabilidad *p* de tener un ´exito.

Rango: *{*1*,* 2*, · · · }*

Media: 1. Varianza: 1*−p*

*p*

*p*2

Par´ametros: *p ≡* probabilidad del evento 1.

Funci´on de masa:

*p*(*x*) =

 *p*(1 *− p*)*x−*1 *x ∈* 1*,* 2 *· · ·*

 0 en otro caso

Con las funciones geopdf, geocdf,geostat y geornd se puede calcular las fun- ciones de masa, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on geom´etrica como se muestra en el siguiente c´odi- go de Matlab:

x=0:0.1:10;

%Geom´etrica

%%%%%%%%%%%

%par´ametros p=0.2;

%title(’Funci´on de masa Geom´etrica’) y\_masa\_geo=geopdf(x,p);

%title(’Funci´on de distribuci´on Geom´etrica’) y\_dist\_geo=geocdf(x,p);

%graficas f7=figure;

figure(f7); ax=subplot(1,2,1); plot(x,y\_masa\_geo,’\*’) ax=subplot(1,2,2); plot(x,y\_dist\_geo,’\*’)

%estad´ısticos [m,v]=geostat(p);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas x\_geo= geornd(p,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´a:

from scipy.stats import geom import matplotlib.pyplot as plt import random

import numpy as np x=np.arange(0,10,0.1)

p=0.2

y\_masa\_bin=geom.pmf(x,p) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa geom´etrica") plt.show()

y\_dist\_bin=geom.cdf(x,p) plt.plot(x,y\_dist\_bin,’bo’)

plt.title("Funci´on de distribuci´on geom´etrica") plt.show()

#Estad´ısticos mean,var=geom.stats(p)

print("media",mean,"y varianza, ", var) #Generar Valores

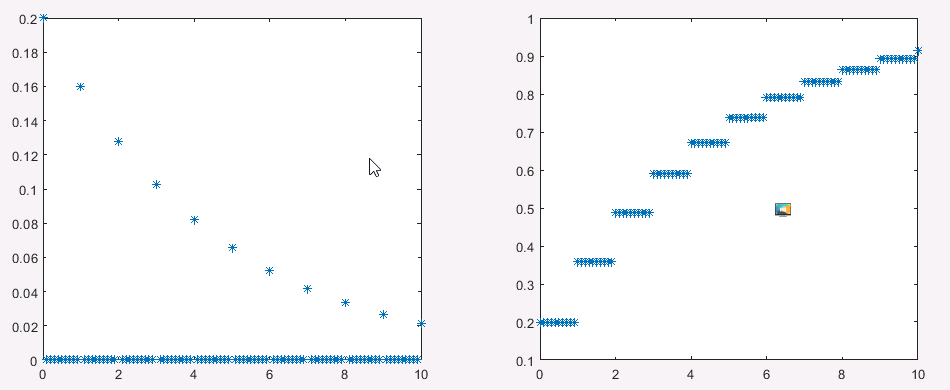


Figura 4: Funci´on de masa (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Geom´etrica(0.2)

r=np.random.geometric(p,size=1000)

Las funciones de masa y de distribuci´on obtenidas se pueden ver en [4](#_bookmark16).

**Ejemplo 1.21.** *Siguiendo con el Ejemplo* [*1.20*](#_bookmark15)*, si se quisiera modelizar el nu´mero de nin˜os antes de tener la primera nin˜a, suponiendo de nuevo que la probabilidad de nacer nin˜a es del 0.48, entonces la variable que modelizar´ıa este suceso ser´ıa una X Geom*(0*.*48)*. As´ı, por ejemplo, la probabilidad de tener 3 nin˜os antes de nacer la primera nin˜a ser´a:*

*≡*

*P* (*X* = 3) = 0*.*523 *·* 0*.*48 = 0*.*06749

**Poisson(L)** : La distribuci´on de Poisson representa el nu´mero de eventos que ocurren en un intervalo de tiempo cuando ocurren con tasa constante. Por ejemplo,el nu´mero de personas que llegan a una gasolinera. As´ı, se ca- racterizar´a por el par´ametro *L*, que representar´a el nu´mero medio de eventos entre dos momentos fijos de tiempo.

Rango: *{*0*,* 1*,* 2*, · · · }* Media: *L*.

Varianza: *L*

Par´ametros: *L ≡* tasa de ocurrencia. Funci´on de masa:

*p*(*x*) =







*e−LLx*

*x*! *x ∈* 0*,* 1*,* 2 *· · ·*

0 en otro caso

Con las funciones poisspdf, poisscdf,poisstat y poissrnd se puede calcular las funciones de masa, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on de Poisson como se muestra en el siguiente c´odigo de Matlab:

x=0:0.1:10;

%Poisson:

%%%%%%

%par´ametros L=5;

%title(’Funci´on de masa Poisson’) y\_masa\_pois=poisspdf(x,L);

%title(’Funci´on de distribuci´on Poisson’) y\_dist\_pois=poisscdf(x,L);

%graficas

f8=figure; figure(f8); ax=subplot(1,2,1);

plot(x,y\_masa\_pois,’\*’) ax=subplot(1,2,2); plot(x,y\_dist\_pois,’\*’)

%estad´ısticos [m,v]=poisstat(L);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas x\_pois= poissrnd(L,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´a:

from scipy.stats import poisson import matplotlib.pyplot as plt import random

import numpy as np x=np.arange(0,10,0.1)

L=5

y\_masa\_bin=poisson.pmf(x,L) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa Poisson") plt.show()

y\_dist\_bin=poisson.cdf(x,L) plt.plot(x,y\_dist\_bin,’bo’) plt.title("Funci´on de distribuci´on Poisson") plt.show()

#Estad´ısticos mean,var=poisson.stats(L)

print("media",mean,"y varianza, ", var)

#Generar Valores r=np.random.poisson(L,size=1000)

Las funciones de masa y de distribuci´on obtenidas se pueden ver en [5](#_bookmark18).

**Ejemplo 1.22.** *Durante la Segunda Guerra Mundial, cuando el ej´ercito alem´an bombardeaba intensamente Londres desde Calais con las bombas V2, el ´area central de Londres fue dividida en 576 cuadrados de 0.25 km*2 *cada uno. Como hab´ıan ca´ıdo en total 537 bombas V2 en esa zona, el nu´mero me- dio observado de bombas por cuadrado fue* 537*/*576 = 0*.*932*.*

*De este modo, considerando como tiempos fijos el momento antes de co- menzar el bombardeo y justo despu´es, y suponiendo que las bombas que ti- rar´an los alemanas ser´a similares, se puede modelizar la variable X* =*nu´me- ro de bombas que caen en Londres en un cuadrado con una distribuci´on*

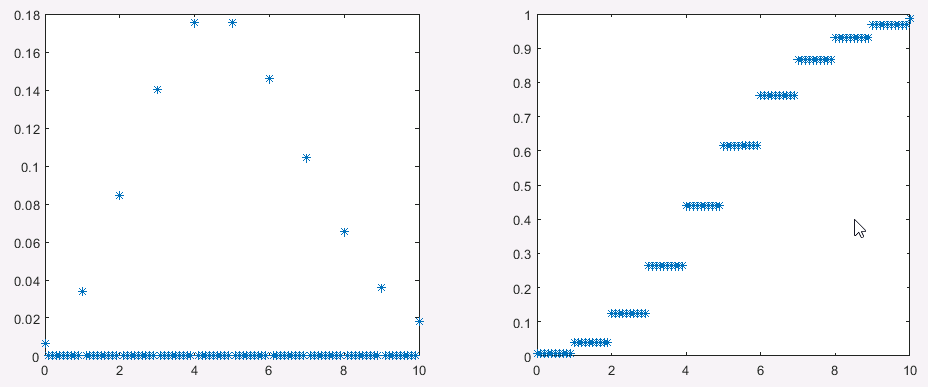


Figura 5: Funci´on de masa (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Poisson(5)

*X Poisson*(0*.*932)*. As´ı, se puede calcular la probabilidad de que caiga alguna bomba como:*

*≡*

*P* (*X >* 0) = 1 *− P* (*X* = 0) = 1 *−*

*e−*0*.*932 0*.*9320

0! = 1 *− e*

*·*

*−*0*.*932

= 0*.*6062346

##### Distribuciones Continuas

**Uniforme(a,b)** : La distribuci´on uniforme se caracteriza porque todos los eventos tienen la misma probabiblidad. Es decir, la probabilidad est´a igual- mente distribuida en un intervalo. Para definir esta distribuci´on se utilizan los par´ametros *a* y *b*, que representan los valores m´ınimos y m´aximos que toman los eventos que se est´an considerando.

Rango: (*a, b*) Media: *a*+*b* . Varianza: (*b−a*)2

2

12

Par´ametros: *a* primero valor, y *b* u´ltimo valor del intervalo

Funci´on de densidad: *f* (*x*) =  1

*b−a*

Con las funciones unifpdf, unifcdf,unifstat y unifrnd se puede calcular las fun- ciones de densidad, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on uniforme como se muestra en el siguiente c´odigo de Matlab:

x=-3:0.01:3;

%Uniforme

%%%%%%%%%

%par´ametros a=-1;

b=1;

%title(’Funci´on de densidad Exponencial’) y\_dens\_unif=unifpdf(x,a,b);

%title(’Funci´on de distribuci´on Exponencial’) y\_dist\_unif=unifcdf(x,a,b);

%gr´aficas f9=figure; figure(f9); ax=subplot(1,2,1);

plot(x,y\_dens\_unif,’linewidth’,3) ax=subplot(1,2,2); plot(x,y\_dist\_unif,’linewidth’,3)

%estad´ısticos [m,v]=unifstat(a,b);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas x\_unif= unifrnd(a,b,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´a:

from scipy.stats import uniform import matplotlib.pyplot as plt import random

import numpy as np

x=np.arange(-3,3,0.01) a=-1

b=1

y\_masa\_bin=uniform.pdf(x,a,b) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa uniforme") plt.show()

y\_dist\_bin=uniform.cdf(x,a,b) plt.plot(x,y\_dist\_bin,’bo’) plt.title("Funci´on de distribuci´on uniforme") plt.show()

#Estad´ısticos mean,var=uniform.stats(a,b) print("media",mean,"y varianza, ", var)

#Generar Valores r=np.random.uniform(a,b,1000)

Las funciones de densidad y de distribuci´on obtenidas se pueden ver en [6](#_bookmark20).

**Ejemplo 1.23.** *Sup´ongase que la concentraci´on de cierto contaminante pue- de ser, de manera equiprobable, de 0 a 20 pares de mill´on. Si se considera t´oxica una concentraci´on de 8 o m´as pares. ¿Cu´al ser´a la probabilidad de que al tomarse una muestra la concentraci´on de ´esta sea t´oxica?*

*Defiendo la variable X* =*concentraci´on del contaminante, como puede tomar los valores 0 a 20 con la misma probabilidad, X U* (0*,* 20)*. As´ı, la probabi- lidad de que una muestra sea t´oxica ser´a:*

*≡*

*P* (*X* 8) = 1 *P* (*X <* 8) = 1 8 *−* 0

*≥ − −*

20 *−* 0

= 0*.*6

**Exponencial(***λ***)** : La distribuci´on exponencial se utiliza para obtener tiem- po entre sucesos independientes. Por ejemplo, tiempo hasta el fallo de un componente, tiempo de llegada a un sistema, tiempos para completar una

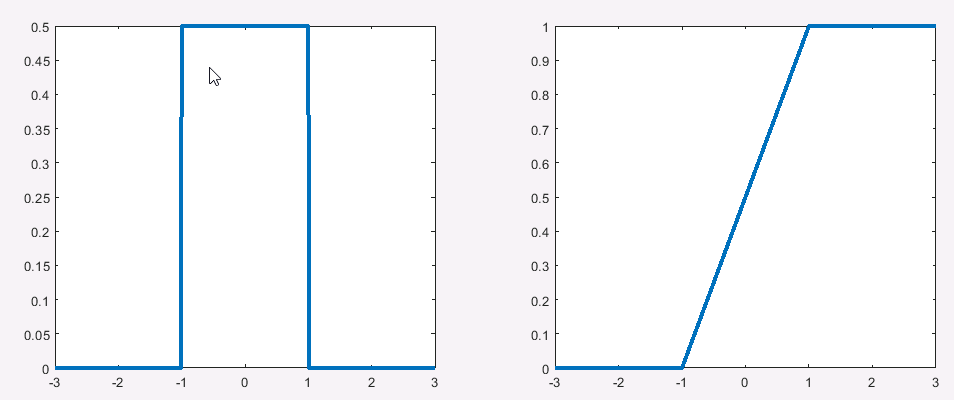


Figura 6: Funci´on de densidad (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Uniforme(-1,1)

tarea, etc. Se caracteriza porque no tiene memoria. En este caso el par´ame- tro que caracteriza la distribuci´on es *λ* representar´a la inversa del tiempo medio entre dos eventos.

Rango: [0*, ∞*) Media: 1 .

*λ*

Varianza: 1

*λ*2

Par´ametros: *λ ≡* tasa de ocurrencia. Funci´on de densidad: *f* (*x*) = 0 *x <* 0

*λe−λx x ≥* 0

Con las funciones exppdf, expcdf,expstat y exprnd se puede calcular las fun- ciones de densidad, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on exponencial como se muestra en el siguiente c´odigo de Matlab:

x=-3:0.01:3;

%Exponencial

%%%%%%%%%%%%%

%par´ametros lambda=0.5;

%title(’Funci´on de densidad Exponencial’) y\_dens\_exp=exppdf(x,lambda);

%title(’Funci´on de distribuci´on Exponencial’) y\_dist\_exp=expcdf(x,lambda);

%gr´aficas f10=figure; figure(f10); ax=subplot(1,2,1);

plot(x,y\_dens\_exp,’linewidth’,3) ax=subplot(1,2,2); plot(x,y\_dist\_exp,’linewidth’,3)

%estad´ısticos [m,v]=expstat(lambda);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas x\_exp= exprnd(lambda,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´a:

from scipy.stats import expon import matplotlib.pyplot as plt import random

import numpy as np x=np.arange(-3,3,0.01) L=1/0.5

y\_masa\_bin=expon.pdf(x,0,L) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa exponencial") plt.show()

y\_dist\_bin=expon.cdf(x,0,L)

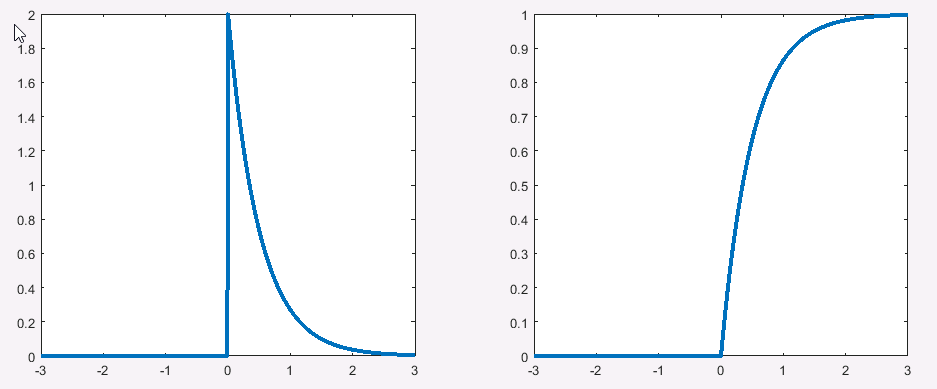


Figura 7: Funci´on de densidad (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Exponencial(0.5)

plt.plot(x,y\_dist\_bin,’bo’)

plt.title("Funci´on de distribuci´on exponencial") plt.show()

#Estad´ısticos mean,var=expon.stats(0,L)

print("media",mean,"y varianza, ", var)

#Generar Valores r=np.random.exponential(L,1000)

Las funciones de densidad y de distribuci´on obtenidas se pueden ver en [7](#_bookmark21).

**Ejemplo 1.24.** *Siguiendo con el Ejemplo* [*1.22*](#_bookmark17)*, y supongamos que el bom- bardeo dur´o 60 minutos y se quiere saber la probabilidad de que pasen menos de 5 minutos entre dos bombas. Si cayeron 0.932 bombas de media en 60 minutos, 1 bomba cae de media cada* 60*/*0*.*932 = 64*.*377 *minutos. Por tanto, la variable Y =tiempo medio entre dos bombas ser´a una X Exp*(1*/*64*.*377)*. De este modo, la probabilidad de que pasen menos de 5 minutos entre dos bombas ser´a:*

*≡*

*P* (*X <* 5) = 0*.*07472

**Normal**(*µ, σ*) : La distribuci´on normal es una de la distribuciones m´as utilizadas y con mayores propiedades. Una de las propuedades m´as impor- tantes es que, si dos normales son incorreladas, entonces son independientes. Adema´s la combinaci´on lineal de normales es normal. En esta distribuci´on los valores se acumulan en torno a un valor medio (*µ*), y var´ıan respecto a una desviaci´on t´ıpica (*σ*), que son los par´ametros que definen la distribuci´on.

Rango: (*−∞,* +*∞*) Media: *µ*.

Varianza: *σ*2

Par´ametros: *µ*, *σ*.

1 (*x−µ*)2)

Funci´on de densidad: *f* (*x*) = *√*

*e*

2*πσ*

2*σ*2

Con las funciones normpdf, normcdf,normstat y normrnd se puede calcular las funciones de densidad, de distribuci´on, la media y la varianza, y generar valores aleatorios de la distribuci´on normal como se muestra en el siguiente c´odigo de Matlab:

x=-3:0.01:3;

%Normal

%%%%%%%%%

%par´ametros mu=0; sigma=1;

%title(’Funci´on de dens Exponencial’) y\_dens\_norm=normpdf(x,mu,sigma);

%title(’Funci´on de distribuci´on Exponencial’) y\_dist\_norm=normcdf(x,mu,sigma);

%gr´aficas f11=figure; figure(f11); ax=subplot(1,2,1);

plot(x,y\_dens\_norm,’linewidth’,3) ax=subplot(1,2,2);

plot(x,y\_dist\_norm,’linewidth’,3)

%estad´ısticos [m,v]=normstat(mu,sigma);

%generar valores nrow=1; % numero filas

ncol=10; %numero de columnas

x\_norm= normrnd(mu,sigma,nrow,ncol);

Ana´logamente, el c´odigo en Python ser´a:

from scipy.stats import norm import matplotlib.pyplot as plt import random

import numpy as np x=np.arange(-3,3,0.01) mu=0

sigma=1

y\_masa\_bin=norm.pdf(x,mu,sigma) plt.plot(x,y\_masa\_bin,’bo’) plt.title("Funci´on de masa normal") plt.show()

y\_dist\_bin=norm.cdf(x,mu,sigma) plt.plot(x,y\_dist\_bin,’bo’) plt.title("Funci´on de distribuci´on normal") plt.show()

#Estad´ısticos mean,var=norm.stats(mu,sigma) print("media",mean,"y varianza, ", var)

#Generar Valores r=np.random.normal(mu,sigma,1000)

Las funciones de densidad y de distribuci´on obtenidas se pueden ver en [8](#_bookmark22).

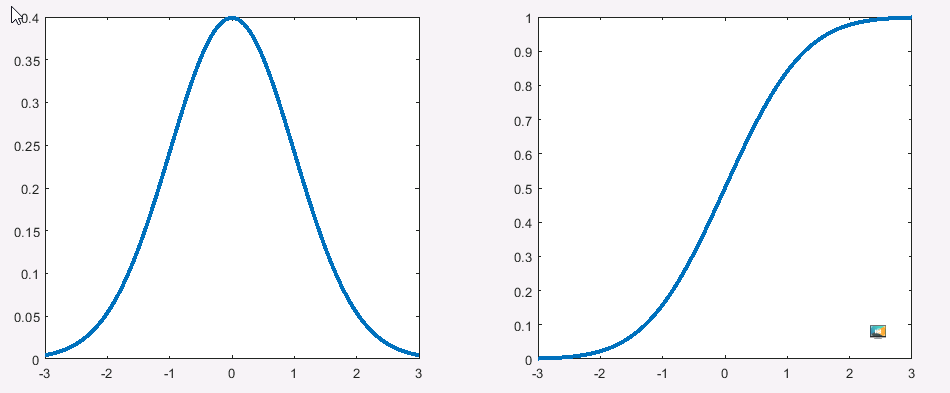


Figura 8: Funci´on de densidad (izquierda) y funci´on de distribuci´on (derecha) de una distribuci´on Normal(0,1)

**Ejemplo 1.25.** *En un quiosco de peri´odicos se supone que el nu´mero de ventas diarias se distribuye normalmente con media 30 y varianza 2. ¿Cu´al ser´a la probabilidad de que en un d´ıa se vendan entre 13 y 31 peri´odicos?*

*Se define la variable X N* (30*, σ* = *√*2)*, que representa las ventas diarias del quiosco. De este modo, la probabilidad requerida ser´a:*

*≡*

*P* (13 *≤ X ≤* 31) = *P* (

*donde Z* = *N* (0*,* 1)*. As´ı,*

13 30

*√*2 *≤ Z ≤*

*−*

31 30

*√*2 )

*−*

*P* (*−*12*.*02082 *≤ Z ≤* 0*.*7071068) = *P* (*Z ≤* 0*.*7071068)*−P* (*Z ≤ −*12*.*02082) = 0*.*76024990

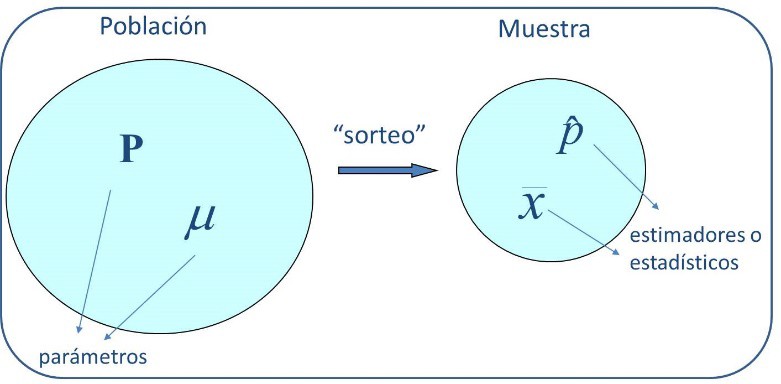
## II Clase

#### Estimadores

Los estimadores son un concepto fundamental que nos permitir´a plantear los problemas de Machine Learning y Deep Learning. Los estimadores con mejo- res propiedades son, en funci´on de la aproximaci´on utilizada, los estimadores de m´axima verosimilitud y los estad´ısticos bayesianos. Los u´ltimos en espe- cial, ser´an los que m´as se utilicen durante el curso. Los estimadores, adem´as, nos permitir´an definir los conceptos de sesgo y varianza, los cu´ales son u´tiles para caracterizar las nociones de generalizaci´on, sobre y bajo entrenamiento que veremos a continuaci´on.

##### Estimaci´on Puntual

Normalmente, a la hora de realizar un experimento, partimos de una po- blaci´on y queremos conocer alguna caracter´ıstica de la misma. Esta carac- ter´ıstica, que no es conocida, se conoce como **par´ametro poblacional**. Sin embargo, no se puede o no interesa analizar el conjunto de la **poblaci´on**. Esto puede deberse tanto a falta de recursos t´ecnicos o econ´omicos como a la dificultad de obtener individuos de la poblaci´on. Por tanto, se trabajar´a con una **muestra** de la poblaci´on de estudio, entendiendo por tal una parte representativa de la misma. As´ı, a partir de la muestra se observar´an las ca- racter´ısticas de estudio y se har´a una estimaci´on del par´ametro poblacional. Esta estimaci´on se conoce como **estad´ıstico muestral** y s´ı es conocido.



Por tanto, la **inferencia estad´ıstica** se ocupa de los m´etodos para inferir propiedades sobre toda la poblaci´on a partir de una muestra, es decir, de las t´ecnicas para sacar conclusiones sobre el todo mirando s´olo a una parte de la poblaci´on.

La estimacion puntual es la t´ecnica estad´ıstica que proporciona la mejor pre- dicci´on de una cantidad (o par´ametro) de inter´es. Esta cantidad suele ser el conjunto de par´ametros del modelo, como los pesos en la regresi´on lineal.

Denotando por *θ* el conjunto de par´ametros reales y *θ*ˆ el conjunto de par´ame- tros estimados, y sea **x**(1)*,* **x**(2)*, , ,* **x**(*m*) el conjunto de *m* datos indepen- dientes e identicamente distribuidos (i.i.d), *estimador puntual* o *estad´ıstico* de cualquier funci´on de datos ser´a:

*{ · · · }*

*θ*ˆ*m* = *g*(**x**(1)*,* **x**(2)*, , · · · ,* **x**(*m*))

Es decir, los estad´ısticos son **funciones de los datos de la muestra** para estimar las caracter´ısticas de la poblaci´on de inter´es, dado que suele ser im- posible analizar a toda la poblaci´on y obtener los valores verdaderos.

As´ı, si por ejemplo se quiere estimar la distancia media que tienen que via- jar los estudiantes de este campus desde que salen de sus casas hasta que llegan al aula, llamemos D a la variable ”la distancia que han de viajar los

estudiantes de este campus desde su casa hasta el aula”. Es razonable asumir que *D* ⇝ *N* (*µ, σ*) donde *µ* y *σ*2 son desconocidos. Para estimar el par´ametro *µ* podemos escoger una muestra de 10 estudiantes y medir la distancia que

viaja cada uno de ellos:

2.43, 3.12, 7.34, 6.33, 2.65, 10.32, 0.55, 6.02, 1.12, 4.33

y calcular su media *x*¯ = 4*.*421. Este valor aproxima el verdadero valor desco- nocido del par´ametro *µ* . Pero tambi´en podr´ıamos escoger otra muestra,

4.32, 2.12, 8.65, 4.32, 12.55, 3.02, 1.18, 6.25, 4.44, 8.31

y en este caso la media, *x*¯ = 5*.*516, es otra aproximaci´on del verdadero valor del par´ametro *µ*.

Como puede observarse, el estad´ıstico media muestral var´ıa de una muestra a otra y es, por tanto, una variable aleatoria que como tal tiene su distribuci´on

de probabilidad. A la distribuci´on de probabilidad de un estad´ıstico muestral se conoce como **distribuci´on muestral**. Por tanto, se podr´a hablar de me- didas de centralizaci´on y de dispersi´on asociadas a los estad´ısticos muestrales.

Como hemos visto, si repetimos el experimento de seleccionar una muestra taman˜o *n* y calcular la media un gran nu´mero de veces, obtendremos una gran cantidad de valores distintos para *X*¯ . A partir de esta “variedad” de valores distintos obtenidos para la media muestral, podemos obtener la distribuci´on de la media muestral. En concreto, podemos calcular el valor esperado y la varianza de estas distribuciones muestrales. De aqu´ı, se define el **error** est´andar como una medida de dispersi´on que mide la variabilidad esperada del promedio muestral como estimaci´on de la media poblacional. Depende del taman˜o de la muestra. Se calcula como

*S*

*SE* = *√n* (5)

donde *S* =  1 L*n* (*xi − x*¯)2 es la cuasivarianza muestral y *n* es el taman˜o

de la muestra. Por tanto, a mayor taman˜o muestral menor variabilidad.

*n−*1

*i*=1

Del mismo, podemos calcular la esperanza y la varianza muestral de cual- quier estad´ıstico.

Recapitulando, nuestro objetivo es estimar los par´ametros de una poblaci´on. Dado que lo estimar´e a partir de la muestra, necesitar´e calcular algu´n es- tad´ıstico para realizar esa estimaci´on. Sin embargo, no todos los estad´ısticos sirven para estimar los mismos par´ametros. S´olo servir´an los estad´ısticos que son **estimadores** de ese par´ametro.

Formalmente, sea X una v.a. con funci´on de densidad *fθ*(*x*) siendo *θ*, el par´ametro a estimar, llamaremos **estimador** T de *θ* a cualquier funci´on de la muestra que contiene el par´ametro desconocido.

*θ* = *T* (*x*1*, · · · , xn*)

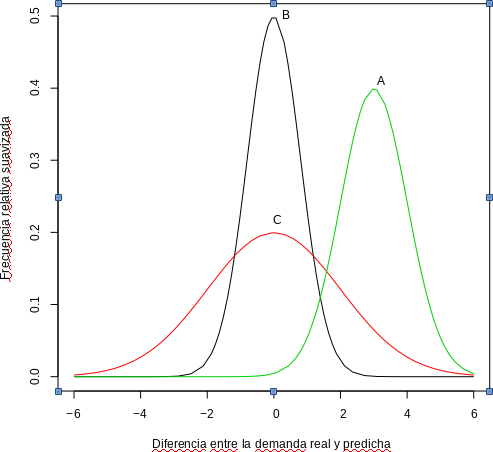
**Ejemplo 2.1.** *Por ejemplo, si queremos estimar la media µ, fµ* = 3*µ* + 4

*podr´ıa ser un estimador de la media mientras que fµ* = 3*σ*2 + 4 *no lo ser´ıa.*

Por tanto, el objetivo de la estimaci´on es usar una muestra para obtener nu´meros que, en algu´n sentido, sean los que mejor estiman a los verdaderos

valores desconocidos de los par´ametros de la poblaci´on.

Por ejemplo, para estimar la demanda de un cierto producto una empresa dispone de tres especialistas A, B y C. En el siguiente gr´afico aparecen histo- gramas aproximados por densidades, para las diferencias entre las demandas que se han producido en la realidad a lo largo de los u´ltimos 36 meses, y la demanda prevista por cada especialista.



Comparando las distintas distribuciones para la diferencia entre la demanda real y la prevista por cada especialista, ¿qu´e especialista podemos decir que se acerca m´as a la demanda real, y por tanto acierta m´as en la estimaci´on? Para que un estimador sea bueno es deseable las siguientes **propiedades**:

Insesgadez: Un estimador es insesgado si la distribuci´on en el muestreo del estimador de *θ* tiene como esperanza un valor igual a *θ*, esto quiere decir que la distribuci´on estar´a centrada alrededor de *θ*.

Eficiencia: La varianza del estimador debe ser la menor posible. Entre dos estimadores T1 y T2, si ambos son insesgados, es mejor aquel con menor varianza.

Consistencia: Al aumentar el taman˜o muestral (m´as datos) mayor debe ser la concentraci´on de la distribuci´on del estimador entorno a *θ*.

**Ejemplo 2.2.** *En el ejemplo anterior, la estimaci´on de los especialistas B y C es insesgada, puesto que est´a centrada en torno al 0, pero la estimaci´on de A no.*

*Adem´as, la estimaci´on m´as eficiente es la B puesto que es la que tiene menor variabilidad.*

*Para analizar la consistencia deber´ıamos repetir el experimento con distintos taman˜os muestrales.*

##### Estimador de M´axima Verosimilud

Hasta ahora hemos visto la definici´on de estimador y visto algunas de sus propiedades, suponiendo que conoc´ıamos la distribuci´on de los datos, es de- cir, la funci´on *f* que genera los datos. Sin embargo, a menudo dicha funci´on no se suele conocer. Por tanto, en lugar de adivinar de qu´e funci´on provienen y calcular despu´es su sesgo y su varianza, nos gustar´ıa tener algu´n tipo de regla que nos diga qu´e funciones son buenos estimadores para los diferentes modelos. Esta ”regla” se conoce como el *principio de m´axima verosimilutd*.

Intuitivamente, la verosimilitud nos dice *c´omo de probable es que los par´ame- tros que he estimado sean correctos a partir de mis datos de entrenamiento*. Es decir, no s´olo elegiremos un estimador con buenas propiedades, sino aquel que sea m´as probable que sea bueno.

El m´etodo de m´axima verosimilitud se basa en la funci´on de verosimilitud. ea *f* (*x|θ*) la funci´on de densidad o funci´on de probabilidad conjunta de *X* = (*X*1*, . . . , Xn*). Entonces, si *X* = *x* ha sido observado, la funci´on de *θ* definida como:

*L*(*θ|x*) = *f* (*x|θ*)

es la **verosimilitud**. Si asumimos que las observaciones (*xi*) son variables aleatorias independientes e id´enticamente distribuidas de la misma funci´on de distribuci´o, entonces:

*L*(*θ|x*) = *f* (*x|θ*) = *f* (*x*1*, . . . , xn|θ*) = *f* (*x*1*|θ*) *∗ f* (*x*2*|θ*) *∗ . . . ∗ f* (*xn|θ*)

Formalmente, consideramos un conjunto de *m* ejemplos **X** = **x**(1)*,* **x**(2)*, ,* **x**(*m*) generados de manera independiente, y de los que desconocemos su distribu- ci´on *pdata*(**x**). Sea *pmodelo*(**x**; *θ*) una familia de funciones de probabilidad sobre

*{ · · · }*

el espacio indexado por *θ*. En otras palabras, *pmodelo*(**x**; *θ*) mapea los valores de **x** en un nu´mero real estimando la probabilidad real *pdata*(**x**). Entonces, el estimador de m´axima verosimilitud de *θ* se define como

*m*

IT

*θML* = *argmaxθpmodel*(**X**; *θ*) = *argmaxθ pmodel*(**x**(*i*); *θ*)

*i*=1

Sin embargo, tomando logaritmo, el resultado no cambia y nos transforma el producto en una suma. Por tanto, tenemos que

*m*

*θML* = *argmaxθ log*(*pmodel*(**x**(*i*); *θ*))

*i*=1

Hemos visto c´omo se calcula el estimador de m´axima verosimilitud de la funci´on de probabilidad de los datos. Este estimador se puede generalizar para estimar una probabilidad condicionada *P* (**Y***|***X***, θ*):

*m*

*θML* = *argmaxθ logP* (**y**(*i*)*|***x**(*i*)*, θ*)

*i*=1

**Ejemplo 2.3.** *Una compan˜´ıa del gas quiere modelizar el nu´mero de llamadas necesarias hasta que un cliente les coja el tel´efono. As´ı, llamaron a 14 clien- tes, siendo los siguientes valores el nu´mero de intentos hasta que consiguieron contactar con ellos.*

10 3 1 11 3 2 2 7 9 4 18 6 3 34

*¿Qu´e distribuci´on deber´ıan utilizar para modelar el nu´mero de llamadas?*

*¿Qu´e par´ametro es necesario para caracterizar a dicha distribuci´on? ¿Cu´al ser´ıa un buen estimador de dicho par´ametro?*

*Considerando ´exito una llamada donde se consigue hablar con el cliente, puesto que hay que repetir el experimento hasta conseguir un ´exito, la com- pan˜´ıa de gas deber´ıa utilizar la variable aleatoria X* = *nu´mero de llamadas hasta que cogen el tel´efono Geom*(*p*)*. As´ı, p, que representa la probabilidad de ´exito (una llamada en la que se habla con el cliente), ser´a el par´ametro que caracteriza la distribuci´on.*

*≡*

*Para estimar este par´ametro se utiliza la estimaci´on m´aximo verosimil. As´ı:*

* + - 1. *Se calcula la funci´on de verosimilitud:*

*P* (*p|x*) = *P* (*p|x*1) *· · · · · P* (*p|x*14) = (6) (1 *− p*)*x*1*−*1 *· p · · · · ·* (1 *− p*)*x*14*−*1 *· p* = (7)

*i*=1

14

*i*=1

(1 *− p*)*xi−*1 *· p* = (1 *− p*) 14

*xi−*14 *· p*14 (8)

* + - 1. *Se calcula la funci´on de log-verosimilitud:*

*log*(*P* (*p|x*)) = *log* ((1 *− p*) 14

*i*=1

*xi−*14 *· p*14 = (9)

= *log* ((1 *− p*) 14 *xi−*14 *log*(*p*14) = (10)

*i*=1

L(

14

*i*=1

*xi −* 14) *· log*(1 *− p*) + 14 *· log*(*p*) (11)

* + - 1. *Se maximiza la funci´on de log-verosimilitud respecto del par´ametro (p):*

*∂log*(*P* (*p|x*))

*∂p*

= *−*(L14 *x −* 14) *·* + 14 *·* = 0 *⇔* (12)

*−p·* 14 *xi*+14*p*+14*·*(1*−p*) = 0 *⇔* (13)

1 1

*i*=1

*i*

1*−p*

*p*

*i*=1

*p*(1*−p*)

*−p ·* L14 *xi* + 14*p* + 14 *·* (1 *− p*) = 0 *⇔* (14)

*i*=1

*p ·* (14 *−* 14 + L14 *xi*) = 14 *⇔* (15)

*i*=1

14

*p* = (

14

113

*i*=1

= 14 = 0*.*1238 (16)

*xi*)

**Ejemplo 2.4.** *Por ejemplo, la regresi´on lineal, como se estudiar´a m´as ade- lante, es un algoritmo que aprende de una entrada* **x** *y produce un valor de*

*salida*

*y*ˆ*. El mapeo de* **x** *a*

*y*ˆ *se hace a trav´es del m´ınimo error cuadr´atico*

*medio (MSE):*

2

*MSE* = 1 (*y*ˆ

*n*

*n*

*i*

*i*

*i*=1

*— y* )

*Como no conocemos la distribuci´on de los datos, supondremos que siguen una distribuci´on Normal (la cual estudiaremos en la clase del pr´oximo martes). As´ı, el EML ser´a:*

*θML* = *argmaxθ*

*m*

*i*=1

*logp*(*y*(*i*)*|***x**(*i*); *θ*) =

= *argmaxθ*

*m*

*i*=1

*log*

1

*√*

2*πσ*

*−||y*ˆ(*i*)*−y*(*i*)*||*2)

*e* 2*σ*2 =

= *argmaxθ*

*−mlogσ −*

*m*

1. *log*(2*π*) *−*

*m*

*i*=1

*||y*ˆ(*i*) *− y*(*i*)*||*2) 2*σ*2

*donde y*ˆ(*i*) *es la salida de la regresi´on del i-´esima entrada* **x**(*i*) *y m es el nu´me- ro de datos de entrada. Al comparar el estimador de m´axima verosimilitud con el MSE, se comprueba que maximizar la log-verosimilutd con respecto de ω es equivalente a minimizar el error cuadr´atico medio. Por tanto, son procedimientos equivalentes.*

##### Estad´ısticos Bayesianos

Los estimadores vistos hasta ahora se corresponden con la aproximaci´on fre- cuentista. As´ı, se trataba de estimar un solo valor de *θ*, y hacer todas las pre- dicciones a partir de esa estimaci´on. Es decir, se tiene un vector de par´ametros *θ* fijo pero desconocido, y se estima *θ*ˆ como una variable aleatoria a partir de los datos. Sin embargo, el problema se puede enfocar mediante la apro- ximaci´on Bayesiana. Es este caso, se considera todos los posibles valores de *θ* para hacer la predicci´on. Es decir, en lugar de utilizar datos (valores de *θ* observado) para realizar la predicci´on utilizaremos distribuciones.

En la aproximaci´on Bayesiana, se considera que *θ* es desconocido y por tanto se representa como una variable aleatoria. Sin embargo, si se tiene alguna informaci´on previa de *θ*, se utilizar´a para construir la distribuci´on de proba- bilidad de *θ*.

Es decir, antes de observar los datos se representa el conocimiento que tene- mos de *θ* a trav´es de la distribuci´on de probabilidad a priori *p*(*θ*). Geneal- mente se escoge distribuci´on con mucha incertidumbre como la uniforme, donde todos los sucesos tienen la misma probabilidad y por tanto no an˜ades

realmente informaci´on. A continuaci´on, se considera que tenemos un conjun- to de datos de muestras *x*(1)*, x*(2)*, , x*(*m*) , y se calcula la probabilidad de obtener dicha muestra a partir de *θ*, *p*(*x*(1)*, x*(2)*, , x*(*m*) *θ*). Sin embar- go, lo que realmente nos interesa es la probabilidad de que esos par´ametros sean ciertos dados esos datos. De este modo, a partir de la regla de Bayes se puede calcular la probabilidad condicionada o probabilidad a posteriori, *p*(*θ|x*(1)*, x*(2)*, · · · , x*(*m*)):

*· · · |*

*{ · · · }*

*p*(*θ|x*

(1)

*, x*(2)

*, · · · , x*

(*m*)) =

*p*(*x*(1)*, x*(2)*, · · · , x*(*m*)*|θ*)*p*(*θ*)

*p*(*x*(1)*, x*(2)*, · · · , x*(*m*))

No´tese que es la idea que subyace similar a la del estimador de m´axima ve- rosimilitud. Sin embargo**, el estimador de m´axima verosimilutd hace predicciones usando una estimaci´on de** *θ***, mientras que la aproxi- maci´on Bayesiana hace predicciones usando una distribuci´on sobre** *θ*. Adema´s, en el enfoque Bayesiano se utiliza la informaci´on que se tiene sobre *θ*, mientras que en el enfoque frecuentista nunca se utiliza.

Una de las ventajas de los m´etodos Bayesianos es que suelen generalizar mu- cho mejor cuando los datos de entrenamiento est´an limitados (se dispone de pocos), pero tienen mayor coste computacional cuando hay muchos datos.

**Ejemplo 2.5.** *Vamos a estudiar de nuevo el problema de regresi´on lineal, pero desde el punto de vista Bayesiano.*

*En este caso, de nuevo los par´ametros del modelo corresponden con el vector ω donde y*ˆ = *ωT* **x***. Tomando la distribuci´on normal o gaussiana como la dis- tribuci´on de las predicciones, se tiene que:*

*p*(**y***|***X**) *≈ exp −* 1 (**y** *−* **X***ω*)*T ·* (**y** *−* **X***ω*)

2

*Suponiendo que la varianza es 1, se obtiene la ecuaci´on del MSE. Para de- terminar la probabilidad a posteriori, especificamos la distribuci´on a priori. Tomando de nuevo la distribuci´on normal, tenemos que:*

*p*(*ω*) *≈ exp −* 1 (*ω − µ* )*T* **Λ***−***1** (*ω − µ* )

**0**

**0**

**0**

2

*donde µ***0** *y* **Λ0** *son el vector de medias y la matriz de covarianzas. Por tanto, la probabilidad a posteriori ser´a*

*p* (*ω* **X***,* **y**) ∝ *p*(**y X***, ω*)*p*(*ω*)

*| |*

∝ *exp −* 1(**y** *−* **X***ω*)*T* (**y** *−* **X***ω*) *exp −* 1(*ω − µ*

2

2

)Λ*−*1(*ω − µ* )

∝ *exp −* 1(*−*2**yTX***ω* + *ω***TXTX***ω* + *ω***TΛ***−***1***ω −* **2***µ* **Λ***−***1***ω*)

**0**

0

**0**

2

**0**

**0**

**0**

*Si definimos ahora* **Λm** = (**XTX** + **Λ***−***0 1**)*−***1** *y µ***m** = **Λm**(**XTy** + **Λ***−***0 1***µ***0**)*, lo anterior se puede reescribir como:*

*p*(*ω|***X***,* **y**) ∝ *exp −* 1(*ω − µ*

1

)Λ*−*1(*ω − µ* ) + *µT* Λ*−*1*µ*

2 **m**

∝ *exp −* 1(*ω − µ*

2

**m**

*m* **m** 2 *m m m*

)Λ*−*1(*ω − µ***m**)

*m*

*Todos los t´erminos que no contengan los par´ametros ω se pueden omitir ya que se consideran constantes y no hay que minimizarlas.*

Al igual que los ejemplos de regresi´on que hemos visto, a menudo se con- siderar como gaussianas (Normales) tanto la distribuci´on a priori como la verosimilitud.

#### Aprendizaje Autom´atico.

##### Capacidad, Sobre Entrenamiento y Bajo Entrenamiento

Hasta ahora hemos visto que se utilizan varios conjuntos de entrenar, con el fin de construir el modelo y comprobar si es generalizable. Para ello, med´ıamos el error en el conjunto de entrenamiento y el de validaci´on y si eran parecidos, se deduc´ıa que el modelo funcionaba bien con datos nuevos y por tanto que era generalizable.

De este modo, si el modelo no es capaz de obtener un error suficientemente bajo, se dice que el modelo est´a *bajoentrenado*. Si el error es bueno pero la diferencia entre el error de entrenamiento y el error de validaci´on es muy grande, se dice que el modelo est´a *sobreentrenado*.

Se puede controlar cuando un modelo es m´as propenso a estar sobre o bajo entrenado modificando su *capacidad*. Es decir, modificando su habilidad para ajustar funciones. La habilidad de las funciones suele venir por el nu´mero de par´ametros que contiene. Los modelos con pocos par´ametros pueden tener dificultades para adaptarse al conjunto de entrenamiento, mientras que los modelos con muchos par´ametros pueden sobredimensionarse al memorizar las propiedades del conjunto de entrenamiento.

**Ejemplo 2.6.** *Uno de los algoritmos de aprendizaje autom´atico m´as simple es la regresi´on lineal. En este algoritmo, a partir de unos datos de entrada* **x** *∈* R*n, se predicen los valores del target y ∈* R *con la f´ormula*

*y*ˆ = **w***T* **x**

*donde* **w** R*n es el vector de par´ametros. Es decir, los valores que controlan el comportamiento del sistema.*

*∈*

*Para ajustar el modelo podemos ajustar la ecuaci´on y*ˆ = *b*+*;*

*y*ˆ = *b* +*ω*1*x*+*ω*2

*; y as´ı sucesivamente podr´ıamos aumentar el polinomio. La funci´on que es- cojamos definir´a la capacidad del modelo y, si no es la adecuada, habr´a sobre entrenamiento o bajo entrenamiento. En la figura* [*9*](#_bookmark30)*, se ve c´omo la funci´on lineal no captura la curva real del problema, es decir, est´a bajo entrenada. Por otra parte, la funci´on con nueve variables representa la funci´on correc- ta, pero tambi´en a las infinitas funciones que pasan por esos puntos, ya que tenemos m´as variables que datos de entrenamiento. Sin embargo, la funci´on cuadr´atica se ajusta a los datos y generaliza bien.*

2

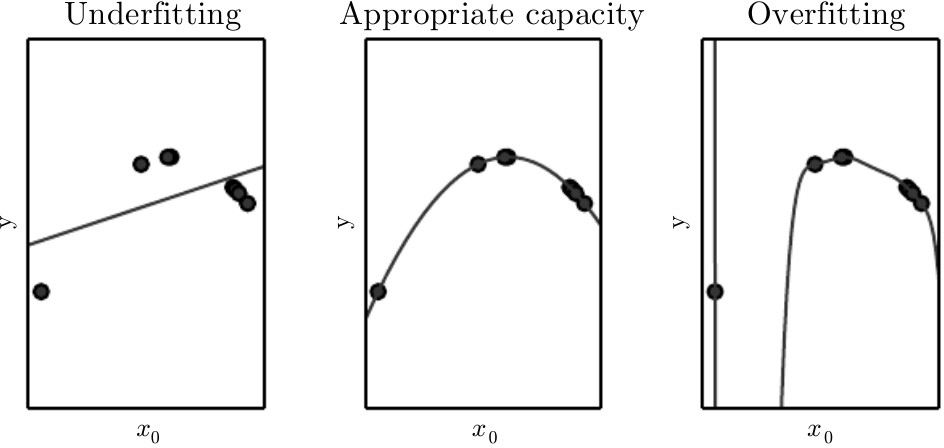


Figura 9: Ejemplo de capacidad, sobre entrenamiento y bajo entrenamiento

##### Sesgo -Varianza

En esta secci´on vamos a definir el sesgo, la varianza, y el compromiso sesgo- varianza.

Supongamos que tenemos una muestra de datos generados por una funci´on *f* y ruido. A partir de esos datos, construimos una funci´on, *f*ˆ que ”genera” esos datos con el menor error posible. Intuitivamente, el *sesgo* hace referencia a la diferencia entre *f* y *f*ˆ. Es decir, a la diferencia entre la funci´on real y la funci´on estimada. Sin embargo, la funci´on real *f* a menudo es desconocida.

Formalmente, el sesgo (*bias*) de un estimador se define como:

*bias*(*θ*ˆ*m*) = *E θ*ˆ*m − θm*

1 l

Es decir, es la diferencia entre los valores medios que se esperan de los datos estimados y los datos reales. No´tese que aqu´ı *θm* hace referencia a los par´ame- tros que definen *f* , y *θ*ˆ*m* hace referencia a los par´ametros que estimamos para construir *f*ˆ.

Por otro lado, la *varianza* (*V* (*θ*)) hace referencia a cu´anto esperamos que varien los datos como funci´on de los datos de entrada, es decir, cu´anto var´ıan los datos en *f*ˆ. La ra´ız cuadrada de la varianza se conoce como *error est´andar* (*SE*(*θ*ˆ)).

Un buen estimador tendr´a un sesgo pequen˜o y una varianza pequen˜a. Sin embargo, cu´anto menor es el sesgo, mayor es la varianza y viceversa, cu´anto mayor es el sesgo menor es la varianza. Por eso, se dice que hay que llegar a un compromiso de sesgo-varianza. Es decir, elegir un modelo que, au´n pudiendo mejorar el sesgo y la varianza por separado, ambas medidas sean lo m´as pequen˜as posibles.

**Ejemplo 2.7.** *En a figura* [*10*](#_bookmark32) *se comprueba c´omo disminuye o aumenta el sesgo al ajustar modelos con m´as o menos capacidad predictiva. En concreto, al ajustar par´abolas el sesgo es menor (ya que se ajusta mejor a los datos) pero la varianza es alta (el modelo var´ıa mucho con los datos) Sin embargo, si ajustamos una recta el modelo tendr´a un sesgo m´as alto (ya que se ajusta peor a los datos), pero menor varianza (puesto que el model var´ıa poco con los datos).*

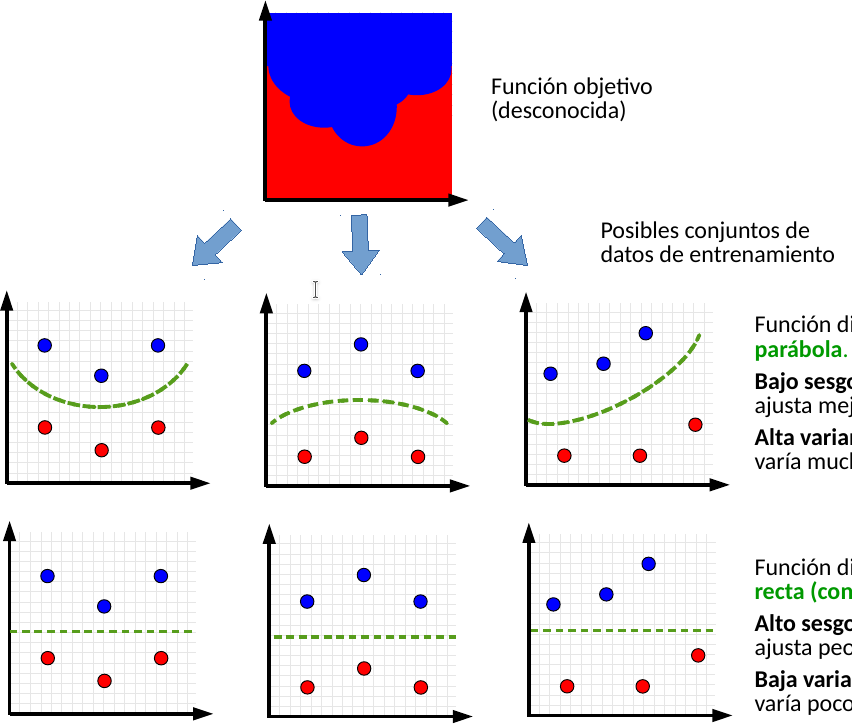


Figura 10: Compromiso sesgo-varianza

#### Ejercicios Propuestos

1. Un dado ha sido manipulado de tal manera que la probabilidad de obtener cada puntuaci´on es directamente proporcional a la misma. Sea X la variable aleatoria *puntuaci´on obtenida* al lanzar el dado. Se pide:
   1. Construir con Matlab/Python las funciones de masa y de distri- buci´on y representarlas gr´aficamente.
   2. Construir una funci´on para generar valores de dicha variable. Ge- nerar 10, 100, 1000 y 10.000 valores y comprobar las frecuencias de los resultados. ¿Qu´e diferencias se observan?
   3. Calcular te´oricamente la esperanza y la varianza de la distribuci´on.
   4. ) Calcular con Matlab/Python la esperanza y la varianza de la dis- tribuci´on. ¿Se parece a la calculada te´oricamente?
2. Sea X una variable aleatoria con funci´on de densidad

1 *ex* si *x ≤* 0

*f* (*x*) =

2

1*e−x* si *x >* 0

2

* 1. Comprueba que es una funci´on de densidad.
  2. Representa gr´aficamente su funci´on de densidad y su funci´on de distribuci´on

1. Generar 1000 nu´meros de una normal de media 5 y varianza 2, y otros 1000 nu´meros de una normal de media 4 y varianza 3. Calcular con Matlab/Python la covarianza y determinar si las distribuciones pueden o no ser independientes.
2. El porcentaje de piezas defectuosas es 4 %. Las piezas se empaquetan en lotes de 100. El cliente rechaza el lote si contiene m´as de dos piezas defectuosas. Calcular con Matlab/Python el porcentaje de lotes recha- zados que puede esperar el fabricante si el proceso de fabricaci´on no ha sufrido modificaciones.
3. En dos grupos de segundo an˜o de carrera se ha medido el coeficiente de inteligencia de los alumnos. En el grupo A la medida fue de 100 y la desviaci´on t´ıpica de 10, mientras que en el grupo B estas medidas fue- ron 105 y 12 respectivamente. Supongamos que ambos grupos tienen el mismo nu´mero de alumnos. Se escoge un alumno al alzar y se comprue- ba que su coeficiente es mayor que 120. Calcular con Matlab/Python, suponiendo normalidad, la probabilidad e que el citado alumno perte- nezca al grupo B
4. En una guarder´ıa a la que acuden un gran nu´mero de nin˜os, el 62 % de los asistentes no tiene ningu´n hermano. En el patio de la guarder´ıa hay 7 nin˜os.
   1. ¿Cu´al es la probabilidad de que al menos uno de ellos tenga algu´n hermano?
   2. ¿Cu´al es la probabilidad de que como mucho dos de ellos tengan hermanos?
   3. ¿Cu´al es la probabilidad de que tres de los nin˜os tengan hermanos y el resto sean hijos u´nicos?
5. El nu´mero de llegadas de votantes a cierta mesa electoral sigue una distribuci´on de Poisson con una media de 0.6 votantes por minuto.
   1. Calcular la probabilidad de que entre las 12.17 y las 12.19 de la man˜ana llegue algu´n votante a la mesa.
   2. Hallar la probabilidad de que transcurran m´as de 3 minutos entre las llegadas de dos votantes consecutivos.
6. La vida activa de un f´armaco, medida en d´ıas, sigue una distribuci´on uniforme en el intervalor (400,550)
   1. Calculala probabilidad de que una unidad de este f´armaco dure m´as de 500 d´ıas.
   2. Sabiendo que una unidad de dicho f´armaco ha durado ya m´as de 450 d´ıas, calcula la probabilidad de que su duraci´on total sea de al menos 500 d´ıas.
   3. Calcula la duraci´on esperada de las unidades de f´armaco y su varianza.
   4. ) Un lote contiene 10 unidades de este f´armaco. Calcular la proba- bilidad de que al mnoes 8 de ellos duren m´as de 430 d´ıas.
7. A partir de los siguientes datos, ¿dir´ıas que el modelo est´a sobreentre- nado, bajoentrenado o bien ajustado?

|  |  |  |
| --- | --- | --- |
| Modelo | Error Entrenamiento | Error en Validaci´on |
| Modelo 1 | 0.69 | 0.80 |
| Modelo 2 | 0.89 | 0.74 |
| Modelo 3 | 0.99 | 0.65 |
| Modelo 4 | 0.56 | 0.48 |

1. Una fotocopiadora est´a programada para sacar 5 copias de cada origi- nal.
   1. Est´ımese por m´axima verosimilitud el porcentaje de fallos de la fotocopiadora. Para ello se toma una m.a.s de 100 de los origi- nales fotocopiados en una semana, siendo defectuosas 5 copias de distintos originales.
   2. Un t´ecnico lleva a cabo determinados ajustes en la fotocopiadora para reducir su porcentaje de fallo. Se toma una m.a.s de 50 de los originales fotocopiados con posterioridad. Se examinan sus copias y resultan ser defectuosas 2. ¿Se ha reducido el porcentaje de fallo?
2. La llegada de veh´ıculos a una gasolinera se distribuye segu´n una ley Poisson donde *λ* refleja el nu´mero de veh´ıculos que llegan a la esta- ci´on de servicios en 1 minuto. Durante un mes se escogen al azar 100

intervalos de 1 minuto en el horario comercial de la gasolinera, siendo el promedio observado de veh´ıculos por minuto igual a 2. Realiza una estimaci´on m´aximo verosimil de *λ* segu´n la informaci´on obtenida.

1. En un centro comercial el tiempo transcurre entre la llegada de dos clientes consecutivos T viene dado por la siguiente distribuci´on de pro- babilidad:

1

*f* (*t, λ*) =

*λ*

*t*

*e λ , t >*

*−*

0*, λ >* 0

siendo *λ* un par´ametro que representa el tiempo medio entre dos clien- tes. A lo largo de un mes se ha procedido, de forma aleatoria y en 100 ocasiones, a la medici´on del tiempo transcurrido entre dos clientes con- secutivos siendo el tiempo medio de 30 segundos. Estime por el m´etodo de m´axima verosimilitud la probabilidad de que pase m´as de un minuto entre la llegada de un cliente y la del siguiente.

1. La distribuci´on de las ventas diarias de una empresa sigue una distri- buci´on Normal (50, *σ*). Estima por el m´etodo de m´axima versomilitud la probabilidad de que, en un d´ıa cualquiera, las ventas de la empresa se encuentren comprendidas entre 75 y 100 a partir de la muestra:

10*,* 20*,* 30*,* 100*,* 110*,* 110*,* 135*,* 150*,* 90*,* 50*,* 20*,* 10

1. Obt´en un estimador m´aximo veros´ımil de para el par´ametro *b* en la funci´on de distribuci´on:

*f* (*x*) =

*b · xb−*1 0 *≤ x ≤* 1

0

1. El nu´mero de cigu¨en˜as que llegan a Manzanares el Real es, en promedio, 3 a la semana.
   1. Hallar la probabilidad de que llegue alguna cigu¨en˜a a Manzanares el Real.
   2. ¿Cu´al es la probabilidad de que lleguen exactamente 2 cigu¨en˜as?
   3. Si ya han llegado 2 cigu¨en˜as ¿cu´al es la probabilidad de que lleguen m´as de 4 cigu¨en˜as?
   4. ) ¿Cu´al es la probabilidad de que el tiempo transcurrido entre la llegada de dos cigu¨en˜as sea mayor de 1 semana?
2. En una central oleica de Sevilla se realiza una selecci´on secuencial de todas las botellas de aceite de oliva que reciben al d´ıa, mediante un estudio sabemos que la probabilidad de que salga una botella en mal estado es 0.03.
   1. Calcular la probabilidad de que la primera botella que salga de- fectuosa sea la octava.
   2. ¿Qu´e botella analizada es de esperar que sea la primera defectuo- sa?
3. Un profesor ha calculado que el tiempo invertido por los estudiantes en hacer su examen sigue una distribuci´on normal de media 150 minutos y varianza 1600 minutos2.
   1. ¿Cu´al es la probabilidad de que un alumno elegido al azar tarde menos de 2 horas en completar el examen?
   2. Si se eligen dos estudiantes al azar, ¿cu´al es la probabilidad de que al menos uno de ellos complete el el examen en menos de dos horas?
4. La variable aleatoria *H* tiene soporte *SH*=*{*-0.1,0,0.2*}* y funci´on de masa de probabilidad *fH*(*t*) = *ct* + 0*.*3, donde *c* es una constante.
   1. Hallar el valor de *c* para que *fH* sea una funci´on de masa de pro- babilidad.
   2. Encontrar la funci´on de distribuci´on de *H*, su esperanza y su va- rianza.

*c*) Calcular *P* [*H ≤* 0], *P* [*H <* 0*.*1], *P* [*−*0*.*1 *< H <* 0], *P* [*H >* 0*.*5],

*P* [*H ≥ −*0*.*03].

1. En cierto experimento qu´ımico, la temperatura de reacci´on (medida en grados cent´ıgrados) es una variable aleatoria con funci´on de densidad

*bt*2 si *−* 1 *< t <* 2*,*

*fX*(*t*) =

0 en otro caso *.*

* 1. Hallar el valor de *b* que hace que *fX* sea una funci´on de densidad.
  2. Calcular *P* (0 *< X ≤* 1).
  3. Hallar la temperatura media durante la reacci´on.

1. En un bote de caramelos hay de tres tipos:

Tipo 1: 20 caramelos de menta Tipo 2: k caramelos de caf´e

Tipo 3: 21 caramelos de ar´andanos

Del tipo1 hay un 65 % con centro de chocolate, mientras que del tipo2 un 89 % con centro de chocolate, y del tipo3 un 36 %. Me ofrecen uno, y al mirar dentro del bote, veo que hay un caramelo partido al cual se le ve que tiene centro de chocolate. Sabiendo que la probabilidad de que ese caramelo sea de caf´e es de 0.1., obt´en el nu´mero de caramelos de caf´e del bote.

# Master en Visio´n Artificial

### Curso 2023-2024



**APUNTES DE FUNDAMENTOS MATEMA´TICOS: II PARTE**

### Autores:

Emanuele Schiavi, Iv´an Ram´ırez y Victoria Ruiz Parrado

1

©2023 Emanuele Schiavi, Iv´an Ram´ırez D´ıaz, Victoria Ruiz Parrado. Al- gunos derechos reservados. Este documento se distribuye bajo la licencia “Atribuci´on-CompartirIgual 4.0 Internacional” de Creative Commons,

disponible en <https://creativecommons.org/licenses/by-sa/4.0/deed.es>.

## ´Indice

1. [III Clase](#_bookmark34) 6
   1. [Preliminares: A´lgebra Lineal.](#_bookmark35) 6
      1. [Escalares, Vectores, Matrices y Tensores](#_bookmark36) 6
      2. [Multiplicando Matrices y Vectores](#_bookmark38) 12
      3. [Sistemas Lineales](#_bookmark39) 14
      4. [Matriz Identidad e Inversa](#_bookmark40) 15
      5. [Espacios Vectoriales](#_bookmark41) 17
      6. [Dependencia e Independencia Lineal](#_bookmark42) 18
      7. [Conjuntos de Generadores, Bases y Bases Ortogonales](#_bookmark43) 20
      8. [Matrices y Transformaciones Lineales](#_bookmark44) 26
      9. [Espacio Imagen y Nu´cleo](#_bookmark45) 27
      10. [Normas](#_bookmark46) 30
2. [IV Clase](#_bookmark47) 34
   1. [Tipos Especiales de Matrices y formas cuadr´aticas](#_bookmark48) 34
      1. [Formas Cuadr´aticas](#_bookmark49) 35
   2. [El Problema de Autovalores](#_bookmark50) 39
      1. [Diagonalizabilidad](#_bookmark52) 41
      2. [Operador Traza y Determinante](#_bookmark55) 46
      3. [Condicionamiento de un Sistema](#_bookmark56) 48
      4. [Factorizaci´on de Cholesky y Factorizaci´on QR](#_bookmark57) 48
      5. [Descomposici´on Espectral](#_bookmark58) 50
3. [V Clase](#_bookmark59) 52
   1. [Descomposici´on por Valores Singulares (SVD)](#_bookmark60) 52
   2. [La Pseudo Inversa de Moore-Penrose](#_bookmark62) 57
   3. [Aplicaciones del A´lgebra Lineal](#_bookmark66) 60
      1. [Compresi´on de Im´agenes Digitales](#_bookmark67) 60
      2. [An´alisis de Componentes Principales (PCA)](#_bookmark68) 62
4. [VI Clase](#_bookmark69) 65
   1. [A´lgebra Lineal: Teor´ıa de los Sistemas Lineales](#_bookmark70) 65
      1. [Principio de Hadamard y Dependencia Continua](#_bookmark71) 66
      2. [Teorema de Rouch´e-Frobenius](#_bookmark73) 67
      3. [F´ormula de la Dimensi´on](#_bookmark74) 68
      4. [Sistemas Sobredeterminados e Indeterminados](#_bookmark76) 68
   2. [El Problema de M´ınimos cuadrados](#_bookmark77) 69
      1. [Ecuaciones Normales](#_bookmark78) 69
      2. [Aplicaci´on de la Pseudo Inversa de Moore-Penrose](#_bookmark79) 70
5. [VII Clase](#_bookmark82) 83
   1. [Aplicaciones](#_bookmark83) 83
      1. [Regresi´on Lineal](#_bookmark84) 83
      2. [Aplicaci´on: Regresi´on Lineal in machine learning](#_bookmark86) 85
      3. [Aplicaci´on: Calibrado de C´amara](#_bookmark87) 87
      4. [El problema aproximado](#_bookmark92) 89
      5. [Resoluci´on](#_bookmark100) 94
6. [VIII Clase](#_bookmark103) 98
   1. [A´lgebra Num´erica](#_bookmark104) 98
      1. [M´etodos Directos: Factorizaci´on LU](#_bookmark105) 98
      2. [La t´ecnica del Pivoteo](#_bookmark107) 102
   2. [A´lgebra Num´erica: M´etodos Iterativos](#_bookmark108) 103
      1. [M´etodos de Resoluci´on Iterativos](#_bookmark109) 103
      2. [Precondicionadores](#_bookmark113) 104
      3. [El M´etodo de Jacobi](#_bookmark116) 105
      4. [El M´etodo de Gauss-Seidel](#_bookmark118) 107
      5. [M´etodos de Relajaci´on](#_bookmark120) 109
      6. [M´etodos de Richardson y del Gradiente](#_bookmark121) 111
      7. [Algoritmo del Gradiente](#_bookmark123) 112
      8. [El M´etodo del Gradiente Conjugado: CG y PCG](#_bookmark124) 114
      9. [Sistemas No Sim´etricos: M´etodos de Krylov. GMRES](#_bookmark125) 114
7. [IX Clase](#_bookmark126) 116
   1. [C´alculo y Optimizaci´on Multi-variable](#_bookmark127) 116
      1. [Introducci´on y Resultados B´asicos](#_bookmark128) 116
      2. [Optimizaci´on basada en el c´alculo del gradiente](#_bookmark129) 117
      3. [M´etodo del Descenso](#_bookmark132) 127
      4. [Matriz Jacobiana y Hessiana](#_bookmark134) 129
      5. [M´etodo del Descenso revisitado](#_bookmark135) 133
   2. [C´alculo y Optimizaci´on Multi-variable](#_bookmark138) 136
      1. [El M´etodo de Newton para minimizaci´on](#_bookmark139) 136
8. [X Clase](#_bookmark142) 140
   1. [Sistemas no Lineales: resoluci´on num´erica](#_bookmark143) 140
      1. [El M´etodo de Newton para sistemas](#_bookmark144) 141
9. [XI Clase](#_bookmark145) 145
   1. [Problemas de M´ınimos Cuadrados No Lineales](#_bookmark146) 145
      1. [El M´etodo de Descenso Gradiente para m´ınimos cua-](#_bookmark152) [drados](#_bookmark152) 148
      2. [El M´etodo de Gauss-Newton](#_bookmark153) 148
      3. [El M´etodo de Levenberg-Marquard](#_bookmark155) 150
10. [XII Clase](#_bookmark156) 154
    1. [C´alculo Variacional](#_bookmark157) 154
       1. [Funcionales de Energ´ıa](#_bookmark158) 154
       2. [Modelos lineales: eliminaci´on de ruido y reconstrucci´on](#_bookmark161) 156
       3. [La Regularizaci´on de Tikhonov. Generalizaci´on](#_bookmark163) 158
       4. [Ecuaciones de Euler-Lagrange](#_bookmark167) 160
       5. [Descenso Gradiente y Filtrado](#_bookmark173) 162
       6. [Formulaci´on Bayesiana](#_bookmark176) 165
       7. [C´alculo riguroso de las Ecuaciones de Euler-Lagrange](#_bookmark180) . 167
       8. [Discretizaci´on: F´ormulas en diferencias finitas](#_bookmark182) 170
11. [XIII Clase](#_bookmark183) 172
    1. [C´alculo Variacional: Planteamiento de la Pr´actica](#_bookmark184) 172
       1. [Resoluci´on Num´erica: Minimizaci´on de Funcionales de](#_bookmark185) [Energ´ıa](#_bookmark185) 172
12. [XIV Clase](#_bookmark186) 173
    1. [Recuperaci´on, dudas y ejercicios](#_bookmark187) 173
13. [Ap´endice: Aplicaciones del gradiente de una imagen](#_bookmark188) 174

|  |  |  |
| --- | --- | --- |
| [13.1. C´alculo del Gradiente de una imagen digital](#_bookmark189) | . . . . . . . . . . | 174 |
| [13.2. Determinaci´on de los bordes de una imagen](#_bookmark190) | . . . . . . . . . . | 178 |

1. [Ap´endice: Ecuaciones no lineales](#_bookmark191) 180
   1. [Motivaci´on y generalidades](#_bookmark192) 180
   2. [M´etodos generales para la resoluci´on de una u´nica ecuaci´on](#_bookmark193)

[no lineal](#_bookmark193) 182

* 1. [El m´etodo de bipartici´on](#_bookmark194) 184
  2. [El m´etodo de aproximaciones sucesivas (punto fijo)](#_bookmark195) 189
  3. [El m´etodo de Newton-Raphson](#_bookmark196) 192
  4. [Variantes del m´etodo de Newton-Raphson: m´etodos de la se-](#_bookmark197) [cante y de “regula falsi”](#_bookmark197) 194
  5. [M´etodo de la secante.](#_bookmark198) 195
  6. [El m´etodo de “Regula Falsi”.](#_bookmark199) 196
  7. [Aceleraci´on de la convergencia de los m´etodos iterativos: m´eto-](#_bookmark200)

[do ∆2 de Aitken](#_bookmark200) 197

* 1. [Ejercicios](#_bookmark201) 197

1. [**Ap´endice: Optimizacion Convexa**](#_bookmark202) **199**
   1. [Optimizaci´on Convexa](#_bookmark203) 199
      1. [Criterio de Optimalidad](#_bookmark206) 202
      2. [Slack Variables](#_bookmark213) 206
      3. [Epigraph Formulation](#_bookmark215) 207
      4. [Dualidad](#_bookmark216) 208
      5. [Las Condiciones de Optimalidad KKT](#_bookmark219) 211
      6. [Resumen Optimizaci´on con Restricciones](#_bookmark220) 212
   2. [Programaci´on Convexa](#_bookmark222) 215
      1. [Programaci´on Lineal](#_bookmark223) 215
      2. [Problemas Lineales](#_bookmark228) 220
      3. [Programaci´on Cuadr´atica](#_bookmark229) 230
      4. [Problemas Cuadr´aticos](#_bookmark234) 240

## III Clase

#### Preliminares:

**A´lgebra Lineal.**

Empezaremos introduciendo las definiciones y notaciones cl´asicas del a´lgebra lineal. Servir´an de base para estructurar y operar con la informaci´on, es decir los datos. Ser´a bueno complementar esta informaci´on con algu´n texto b´asico de ´algebra lineal que hayais visto durante la carrera.

##### Escalares, Vectores, Matrices y Tensores

Todas las magnitudes, f´ısicas, artificiales o l´ogicas se pueden almacenar y representar a trav´es Escalares, Vectores, Matrices y Tensores. Por escalar entendemos a un nu´mero real *λ* R. Por escalar complejo entendemos a un nu´mero *z* = *a* + *ib* = (*a, b*) R2 en donde hemos utilizado un vector (*a, b*) R2 para almacenar el nu´mero complejo, siendo *a* R la parte real y *b* R la parte imaginaria del nu´mero complejo *z* R2. Podemos por tan- to visualizar un nu´mero complejo como un punto en el plano real. M´as en general definimos a un vector mediante una colecci´on ordenada de escalares denotada por **v** = (*v*1*, ..., vn*) *∈* R*n* siendo *vi ∈* R la *i−*´esima componente. Una colecci´on ordenada de *m* vectores *{***v1***, ...,* **vm***}* de R*n* se puede almacenar en una matriz de *n* filas y *m* columnas denotada por *A ∈* R*n,m*. Un elemento gen´erico de una matriz se denota por *ai,j* y su valor se corresponde a la in- tensidad de gris del pixel correspondiente.

*∈ ∈*

*∈ ∈*

*∈*

*∈*

Por ejemplo para generar (aleatoriamente) una matriz *M* R*m,n* y represen- tarla gr´aficamente podemos utilizar las instrucciones

*∈*

m=5; n=3; A=rand(m,n);

figure, colormap gray, imagesc(A)

**Ejemplo 1.1.** *Sea dada una imagen en escala de grises, por ejemplo la del cuadro de Dali que se encuentra en el campus virtual. Leer la imagen en matlab, almacenarla en una matriz, pasarla a doble precisi´on en el rango* [0*,* 1] *y visualizarla. Determinar las dimensiones de la imagen.*

Tras almacenar la imagen en una carpeta la podemos leer y almacenar en matlab mediante el comando

f=imread(’./imagenes/dali-bw.jpg’);

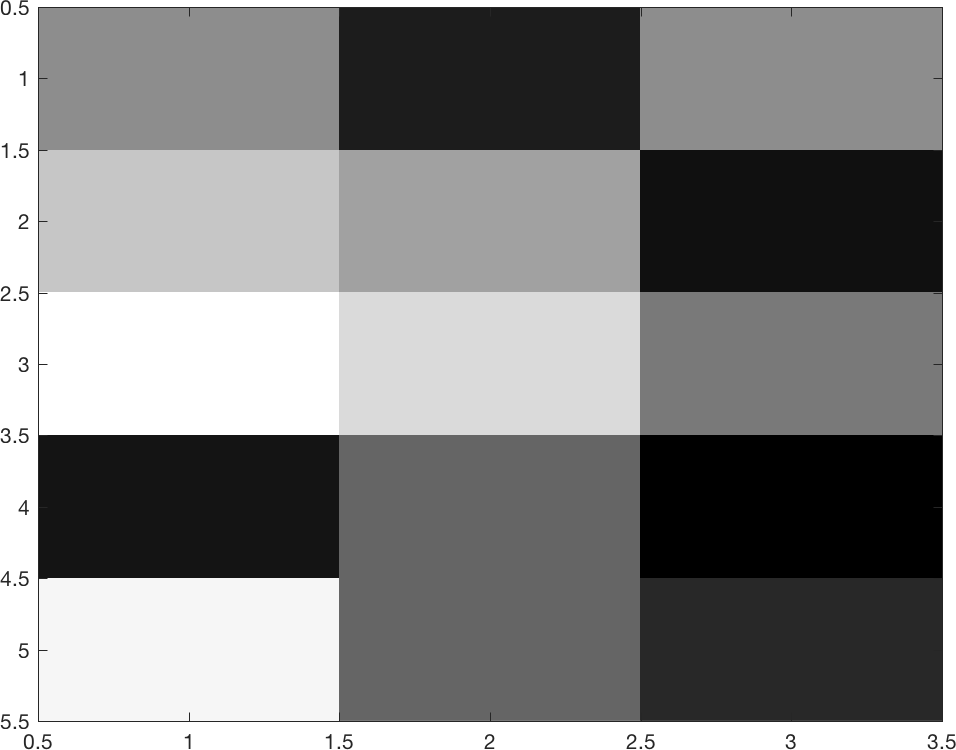


Figura 1: Matriz aleatoria de dimensiones 5 *×* 3.



Figura 2: Cuadro de Dal´ı. La imagen se codifica como una matriz de dimen- siones 364 *×* 509.

en donde *f M* 364*,*509. La pasamos a a doble precisi´on en el rango [0*,* 1] mediante el comando

*∈*

f=im2double(f);

Podemos visualizarla (figura [2](#_bookmark37)) mediante

figure imshow(f,[])

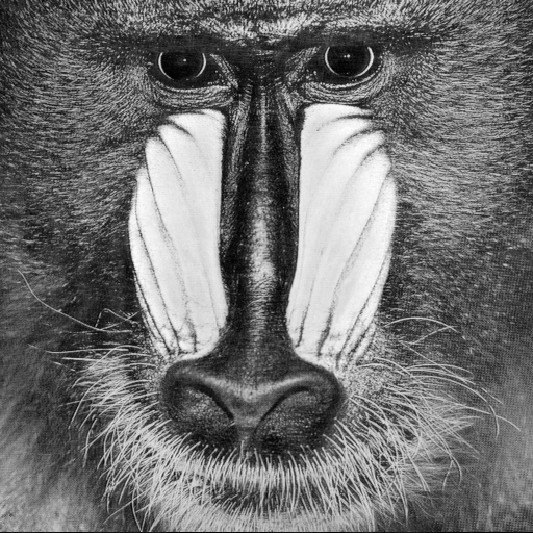
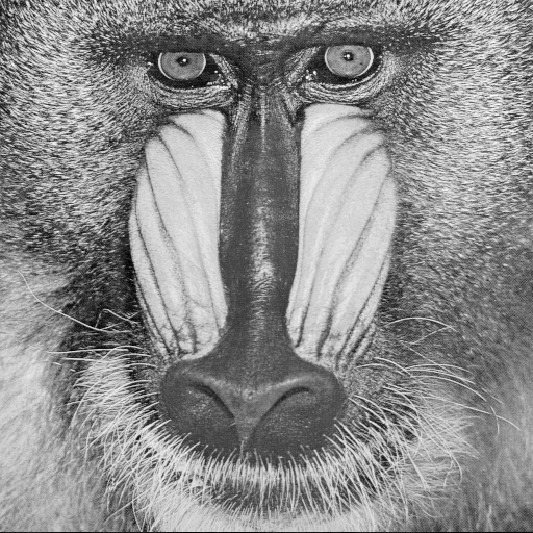
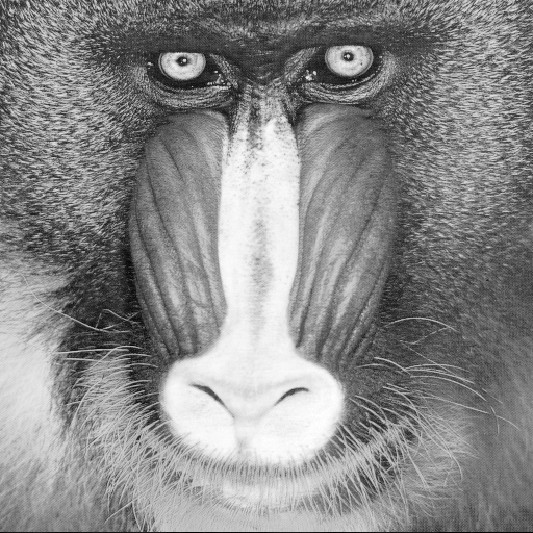


Figura 3: La imagen en figura, en color, se almacena como un tensor **A**.

Una matriz con una u´nica columna representa un vector. Si tiene una u´nica fila es un vector fila. Un vector con una u´nica componente es un escalar. Si dotamos a una matriz de una ulterior dimensi´on obtenemos un **tensor** que denotamos con letra mayuscola y en negrita **A** siendo *ai,j,k* **A** un gen´erico elemento. Esta estructura se puede generalizar a cualquier dimensi´on.

*∈*

Como consecuencia de este formalismo algebraico vemos que todas las im´age- nes digitales se pueden identificar con una matriz si son im´agenes en escala de grises o un tensor si son im´agenes en color del tipo RGB. M´as en ge- neral se pueden codificar im´agenes multi-canales del tipo las que aparecen en imagen m´edica cuando se consideran distintas modalidades de adquisici´on.



* + - 1. Canal Rojo *i* = 1 (b) Canal Verde *i* = 2 (c) Canal Azul *i* = 3

Figura 4: Codificaci´on de una imagen en color RGB en un tensor **A**(:*,* :*, i*), *i* = 1*,* 2*,* 3

#### Suma de matrices y tensores

Las matrices y los tensores se pueden sumar elemento a elemento lo que im- plica una condici´on de compatibilidad. Tienen que tener la misma dimensi´on. Se tiene, operando elemento a elemento

*A* + *B* = *C, ai,j* + *bi,j* = *ci,j*

#### Rescalamiento

Tambi´en se pueden multiplicar por un escalar *λ* R obteniendo una matriz o tensor de la misma dimensi´on:

*∈*

*B* = *λA, bi,j* = *λai,j*

#### Combinaciones lineales

En general, dados escalares *α* y *β* podemos realizar una combinaci´on lineal de matrices y tensores en la forma:

*C* = *αA* + *βB*

obteniendo una matriz o tensor de las mismas dimensiones.

#### Espacio vectorial

Dotado de las operaciones de suma y producto por un escalar el conjunto de las matrices de las mismas dimensiones *V* = *Mm,n*(R) tiene estructura de es- pacio vectorial. Lo mismo ocurre con el conjunto de tensores *V* = *Mm,n,k*(R).

#### Trasposici´on

Una operaci´on importante en ´algebra lineal es la operaci´on de Trasposici´on. Sea *A* = (*ai,j*). Definimos entonces la matriz traspuesta como *AT* = (*aj,i*), es decir, simplemente intercambiando las filas por las columnas.

Una de las razones fundamentales de la importancia y aplicabilidad del a´lge- bra a la Visi´on Artificial es que las matrices codifican las im´agenes digitales permitiendo aplicar el a´lgebra lineal al procesado de im´agenes.

**Ejercicio 1.1.** *Se consideran las imagenes del babuino baboonBW.tif y del cameraman cameraman.tif. Realizar una combinaci´on lineal y visualizar la imagen obtenida comparando con la imagen de abajo.*

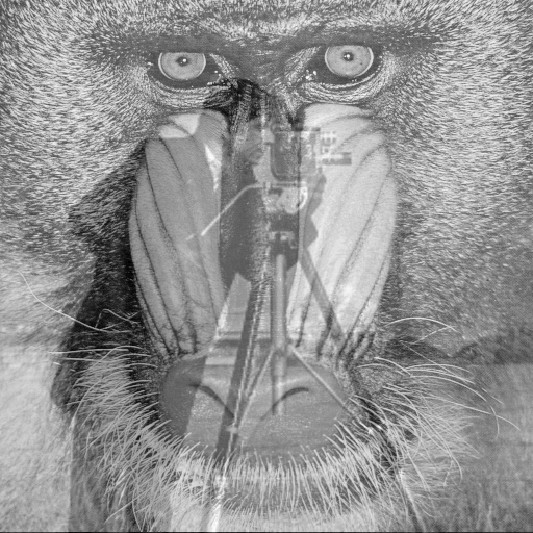


Figura 5: Combinaci´on lineal entre matrices

**Ejercicio 1.2.** *Repetir el ejercicio para una imagen en color almacen´ando- la en un tensor siendo* **A**(:*,* :*, i*)*, i* = 1*,* 2*,* 3 *las componentes del tensor que corresponden a los canales RGB.*



Figura 6: Combinaci´on lineal de im´agenes en RGB

#### Modelo de ruido gaussiano

Las operaciones de suma y multiplicaci´on nos permitir´an por ejemplo definir un proceso de contaminaci´on aditivo (y gaussiano) para im´agenes digitales. Para ello podemos generar una imagen conteniendo s´olo ruido gaussiano me- diante los comandos

f=imread(’./imagenes/lena\_bw.png’); f=im2double(f); noise=randn(size(f)); lambda=max(f(:))\*0.1;

A continuaci´on podemos definir un modelo aditivo de ruido uniforme de tipo gaussiano mediante

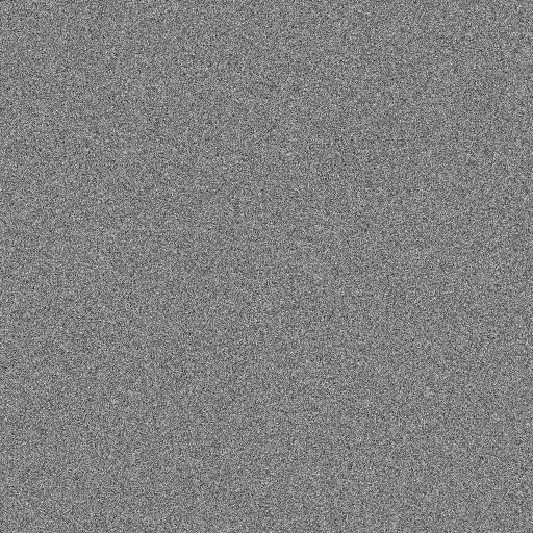
fn=f+lambda\*noise;

en donde *λ* R+ es un par´ametro real positivo que pondera la intensidad del ruido. Observa que la cantidad de ruido se rescala en funci´on del m´aximo de la imagen y la imagen del ruido gaussiano (normal) tiene las mismas dimensiones de la imagen de entrada.

*∈*

**Ejercicio 1.3.** *Sea dada A, una matriz que codifica la imagen original lenabw.png en escala de grises. Generar B una imagen de ruido gaussiano. Contaminar con la imagen de ruido aditivo y gaussiano mediante la f´ormula C* = *A* + *λB,*

*λ* = 0*.*15*. Mostrar la imagen original y la ruidosa obtenida.*



(a) Imagen original (b) Imagen del ruido (c) Imagen ruidosa

Figura 7: Imagen original, Ruido aditivo, Imagen contaminada

En la parte del curso de fundamentos del procesamiento de im´agenes median- te m´etodos variacionales aprenderemos a eliminar parte del ruido presente minimizando adecuados funcionales de energ´ıa.

##### Multiplicando Matrices y Vectores

Definimos ahora la operaci´on de multiplicaci´on entre matrices, conocida como producto filas por columnas:

*Am,nBn,p* = *Cm,p, ci,j* = *ai,kbk,j*

*k*

La condici´on de compatibilidad es que el nu´mero de filas de *A* sea igual al nu´mero de columnas de *B*. La operaci´on producto no es *conmutativa*, es decir *AB ̸*= *BA* en general. Sin embargo vale la propiedad *distributiva*

*A*(*B* + *C*) = *AB* + *AC*

y la *asociativa*

*A*(*BC*) = (*AB*)*C*

La Traspuesta de un producto de matrices viene dada por el producto de las traspuestas en orden invertido:

(*AB*)*T* = *BT AT*

#### El Producto de Hadamard

Tambi´en es posible definir una producto entre matrices de las mismas dimen- siones multiplicando elemento a elemento lo que se conoce como Producto de Hadamard y se denota por

*A ⊙ B* = *C, ci,j* = *ai,jbi,j*

El Producto de Hadamard es conmutativo.

**Ejercicio 1.4.** *Sean dadas las matrices*

 1 2   2 0 

   

*A* = 3 4 *, B* = 1 *−*1

*Calcular con matlab el producto AB y verificar que no es conmutativo. Cal- cular el producto de Hadamard A ⊙ B y verificar que es conmutativo.*

#### Producto escalar

Los vectores se pueden multiplicar mediante la operaci´on de producto escalar que nos da como resultado un escalar. Existen varias formas de denotar el producto escalar

*n*

*⟨***u***,* **v***⟩* = *uivi* = **u** *·* **v** = **u***T* **v** = **v***T* **u** = *|***u***||***v***|* cos(*θ*) *∈* R

*i*=1

en donde hemos enfatizado que el producto escalar es conmutativo ya que el resultado de la operaci´on es un escalar. Se tiene en particular

*⟨***u***,* **u***⟩* = **u** *·* **u** = **u***T* **u** = *|***u***|*2 *∈* R

es decir, el producto escalar de un vector consigo mismo es igual al m´odulo al cuadrado. La operaci´on de producto escalar permite definir el concepto de ortogonalidad entre vectores y entre sub-espacios vectoriales. Si dos vectores son no nulos se tiene que son ortogonales s´ı y s´olo s´ı

*⟨***u***,* **v***⟩* = **u** *·* **v** = **u v** = **v u** = *|***u***||***v***|* cos(*θ*) = 0 ya que esta ecuaci´on tiene soluci´on *θ* = *π/*2.

*T T*

**Ejercicio 1.5.** *Sean dados los vectores* **u** = [2*,* 1]*T ,* **v** = [*−*1*,* 3]*T . Calcular el producto escalar ⟨***u***,* **v***⟩ y el ´angulo en grados entre los dos vectores.*

#### Complemento ortogonal

Sea ahora *V* R*m* un sub-espacio vectorial de R*m*. Llamaremos Comple- mento Ortogonal de *V* al espacio *V ⊥* tal que

*⊂*

*V ⊥* = *{***x** *| ⟨***z***,* **x***⟩* = 0*, ∀* **z** *∈ V }*

Se tiene que *V* y su complemento ortogonal est´an en suma directa (ortogonal)

*V ⊕ V ⊥* = R*m*

denotando por ello que cada vector **w** R*m* se puede escribir de manera u´nica en la forma

*∈*

**w** = **z** + **x***,* **z** *∈ V,* **x** *∈ V ⊥*

**Ejercicio 1.6.** *Sea dado el subespacio vectorial V* R3 *generado por los vectores* **u**1 = (1*,* 2*,* 1)*T ,* **u**2 = ( 1*,* 0*,* 0)*T , es decir V* = *L*[**u**1*,* **u**2]*. Demostrar que su complemento ortogonal es V ⊥* = *L*[**u**3] *siendo* **u**3 = (0*,* 1*,* 2)*T . Veri- ficar que un gen´erico vector* **w** R3 *se puede escribir de manera u´nica como suma de un elemento de V y uno de V T . Verificar finalmente que el vector* **w** = (0*,* 1*,* 2)*T s´olo se puede escribir en la forma*

*∈*

*−*

*−*

*⊂*

 1 

 *−*1 

0 0 0

3













 0 

5  2  + 5  0

4

4

1

0

 *−* 5  1

 =  8*/*5  +  *−*3*/*5  =  1 

4*/*5

6*/*5

2

Observamos finalmente que es posible definir tambi´en un producto entre vectores del tipo columnas por filas cuyo resultado es una matriz.

*−*2

Sean **u***n,*1, **v***n,*1 vectores de R*n*. Entonces

**u***n,*1(**u***n,*1)*T* = **u***n,*1**u***T*

1*,n*

= *An,n*

Esta operaci´on ser´a utilizada para descomponer una matriz mediante la t´ecni- ca de valores singulares.

##### Sistemas Lineales

Con estas notaciones podemos definir el problema fundamental del A´lgebra Lineal que consiste en la resoluci´on del sistema de Ecuaciones Lineales defi- nido por

*A***x** = **b**

en donde *A* R*m,n* es la matriz del sistema, **x** R**n** es el vector de inc´ognitas y **b** R**m** es el vector de datos. Si **b** = **0** el sistema *A***x** = **0** es homog´eneo. El conjunto de soluciones de un sistema homog´eneo es un espacio vectorial. Dependiendo de los valores de *m* y *n* el an´alisis y resoluci´on del sistema pre- senta distintos escenarios que iremos considerando a lo largo de las pr´oximas sesiones. Existe una multitud de aplicaciones cuya resoluci´on nos reconducir´a a resolver sistemas lineales y su resoluci´on computacionalmente eficiente me- diante algoritmos es uno de los retos m´as importantes del c´alculo cient´ıfico. Dependiendo del nu´mero de ecuaciones *m* y del nu´mero de variables inc´ogni- tas *n* los sistemas lineales se clasifican en **Sobredeterminados** si *m > n* e **Indeterminados** si *m < n*. Si *m* = *n* se trata de sistemas cuadrados.

*∈*

*∈ ∈*

Un sistema lineal se dice **Bien Planteado en el sentido de Hadamard**

si se cumplen 3 condiciones:

1. Existe soluci´on
2. Es u´nica
3. Depende con continuidad de los datos del problema

Recordamos que un sistema lineal se denomina de **Compatible** si existe al menos una soluci´on y de **Incompatible** si no existe ninguna. Si la soluci´on es u´nica hablaremos de sistema **Compatible Determinado**.

La u´ltima de las 3 propiedades que verifica un Sistema Bien Planteado, la de Dependencia Continua, se resume diciendo que si los datos del problema tienen pequen˜as perturbaciones tambi´en la soluci´on del problema tendr´a pe- quen˜as perturbaciones en sus componentes. Se puede analizar considerando el nu´mero de condicionamiento del sistema cuya definici´on veremos m´as ade- lante ya que se apoya en el concepto de autovalor. Si una cualquiera de las 3 condiciones fallara entonces el sistema se dir´ıa de **mal planteado**.

##### Matriz Identidad e Inversa

Una herramienta fundamental para la resoluci´on de sistemas lineales de tipo cuadrado (*m* = *n*) es la operaci´on de inversi´on matricial. Para ello es nece- sarion introducir la definici´on de Matriz Identidad. Se trata de una matriz cuadrada *In ∈* R*n,n* tal que

*In***x** = **x***, ∀* **x** *∈* R**n**

es decir que deja invariante el vector sobre el que actu´a. Por ejemplo, en R3

se tiene

 1 0 0   *x*1   *x*1 

 0 1 0   *x*2

0 0 1

*x*3

 =  *x*2 

*x*3

y, en general, denotando *ai,j ∈ In* a la gen´erica componente se tiene *ai,j* = 0 si *i ̸*= *j* y *ai,i* = 1 *∀ i*. La matriz inversa (si existe...) de una matriz gen´erica *A ∈* R*n,n* se define ahora mediante la matriz *A−*1 *∈* R*n,n* tal que

*A−*1*A* = *AA−*1 = *In*

Resolvemos por tanto (cuando posible..) el sistema cuadrado

*A***x** = **b**

multiplicando por la inversa *A−*1 obteniendo su soluci´on **x** en la forma

*A−*1*A***x** = *In***x** = **x** = **A***−***1b**

El algoritmo para el c´alculo de la inversa de una funci´on se aplica en matlab mediante el comando **inv.m** y est´a basado en la factorizaci´on LU de una matriz que veremos m´as adelante al definir el proceso de **Eliminaci´on de Gauss** para la resoluci´on de sistemas lineales.

Pasamos ahora al concepto de Dependencia e Independencia Lineal entre vectores los que nos permite definir los Sistemas de Generadores y extraer de ellos unas bases. Gracias a la operaci´on de producto escalar podremos caracterizar as´ı las bases ortogonales. Fijadas las bases podemos asociar a una matriz su u´nica representaci´on en t´erminos de una aplicaci´on lineal y definir los espacios imagen y nu´cleo que caracterizan la transformaci´on.

Pasaremos luego a medir vector y matrices mediante el concepto de norma y, finalmente, energ´ıa. Presentaremos las normas m´as comunmente utilizadas y la especial relaci´on entre la norma eucl´ıdea y el producto escalar. Al definir al- gunos tipos y estructuras de matrices particularmente simples y la importante operaci´on de trasposici´on introduciremos las matrices sim´etricas, las ortogo- nales y las definidas positivas. Esto permitir´a definir las formas cuadr´aticas y empezar su clasificaci´on atendiendo al signo de las formas cuadr´aticas.

Para su clasificaci´on se considera el problema de autovalores y la informaci´on fundamental que obtenemos al resolverlo. El conocimiento de los autovalores del problema ser´a fundamental para determinar el condicionamiento de un sistema lo que nos permitir´a saber si el sistema est´a bien planteado.

Tras introducir las Factorizaciones de Cholesky y QR veremos como resol- ver, mediante descomposici´on espectral el problema de la diagonalizaci´on y generalizaremos esta teor´ıa al caso no cuadrado mediante la t´ecnica de Descomposici´on en Valores Singulares para finalizar con el concepto funda- mental de pseudo-inversa de Moore-Penrose para la resoluci´on del problema de m´ınimo cuadrados.

##### Espacios Vectoriales

Empezamos introduciendo el concepto de Espacio Vectorial como el de un conjunto infinito, es decir con infinitos elementos o vectores, *cerrado* con res- pecto a las operaciones de suma de vectores y multiplicaci´on por un escalar. Se trata pues del *contenedor* en el cual nos vamos a mover. Matem´aticamente,

**Definici´on 1.1.** *Dados 2 gen´ericos vectores* **u***,* **v** *en el conjunto V diremos que V es un Espacio Vectorial si*

*α***u** + *β***v** *∈ V, ∀ α, β ∈* R

La operaci´on anterior recibe el nombre de Combinaci´on Lineal y permite generar, variando *α*, *β*, a todos los vectores del espacio.

**Ejemplo 1.2.** *Los t´ıpicos espacios vectoriales en donde se trabaja son los V* = R*n. Tienen dimensi´on finita n. Hay espacios vectoriales de dimensi´on infinita, por ejemplo el espacio de las funciones continuas en un intervalo, denotado por V* = *C*(*I*)*. El conjunto de matrices m,n*(*K*) *es un espacio vectorial sobre el cuerpo K de los nu´meros reales o complejos.*

*M*

**Definici´on 1.2.** *Si U y V son espacios vectoriales y U V entonces U es un sub-espacio vectorial.*

*⊂*

**Ejemplo 1.3.** *Sean dado 2 vectores de* R3*, digamos* **u***,* **v***. Si los vectores no son paralelos (es decir no son uno mu´ltiplo del otro) entonces generan un espacio U de dimensi´on 2 (un plano que pasa por el origen) que es un sub- espacio de V* = R3*. Si los vectores son paralelos (es decir son uno mu´ltiplo del otro) entonces generan un espacio U de dimensi´on 1 (una recta que pasa por el origen) que es un sub-espacio de V* = R3*.*

**Ejemplo 1.4.** *Sea dado el espacio vectorial definido por la ecuaci´on*

*V* = *{*(*x*1*, x*2*, x*3) *∈* R3*/x*1 *−* 4*x*2 *−* 5*x*3 = 0*}*

*Vamos a definirlo en t´erminos de combinaciones lineales. Para ello explici- tamos x*1 *en la ecuaci´on*

*x*1 = 4*x*2 + 5*x*3

*y escribimos un gen´erico vector de V en la forma*

(*x*1*, x*2*, x*3) = (4*x*2 + 5*x*3*, x*2*, x*3) = *α*(4*,* 1*,* 0) + *β*(5*,* 0*,* 1)

*en donde α* = *x*2*, β* = *x*3 *son par´ametros reales.*

##### Dependencia e Independencia Lineal

Mediante la operaci´on de Combinaci´on Lineal podemos definir el concepto de Dependencia e Independencia Lineal de un conjunto de vectores.

**Definici´on 1.3.** *Diremos que un conjunto de m vectores* **v1***, ...,* **vm** *de* R*n*

*{ }*

*es Linealmente Independiente si el sistema de n ecuaciones*

*m*

*αi***v***i* = *α*1**v**1 + *...* + *αm***v***m* = **0**

*i*=1

*tiene s´olo la soluci´on trivial* **0** *es decir, todos los coeficientes αi son nulos:*

*α*1 = = *αm* = 0

Observa que la operaci´on de suma de vectores multiplicados por un escalar (o peso)

*m*

*αi***v***i*

*i*=1

utilizada en la definici´on anterior es una Combinaci´on Lineal de vectores y genera un vector. Establecer si un conjunto de vectores es linealmente independiente se reconduce a demostrar que el sistema homog´eneo anterior

*m*

*αi***v***i* = **0**

*i*=1

admite s´olo la soluci´on trivial **0**. Observa que un sistema lineal homog´eneo siempre es Compatible porqu´e siempre tiene la soluci´on trivial **0** entre sus soluciones, por definici´on de sistema homog´eneo. La Independencia Lineal se obtiene cuando la soluci´on trivial **0** es la u´nica soluci´on del sistema.

**Ejemplo 1.5.** *Supongamos tener los vectores* **v**1 = (1*,* 1*,* 0)*T ,* **v**2 = (0*,* 1*,* 2)*T y* **v**3 = (1*,* 0*,* 2)*T y queremos estudiar si son Linealmente Independientes. Construimos la combinaci´on lineal*

*−*

3

*αi***vi** = *α*1**v1** + *α*2**v2** + *α*3**v3** = **0**

*i*=1

*y resolvemos el sistema homog´eneo*

 1 

*α*1  *−*1  + *α*2  1  + *α*3  0  =  0 

 0 

 1 

 0 

0

*definido por las ecuaciones*



2 2 0

*α*1 + *α*3 = 0

*−α*1 + *α*2 = 0



2*α*2 + 2*α*3 = 0

*que tiene las infinitas soluciones*

*α*¯ = (*α*1*, α*2*, α*3)*T* = (*−α*3*, −α*3*, α*3)*T ∈ L*[(*−*1*, −*1*,* 1)]

*luego los* 3 *vectores son Linealmente Dependientes.*

**Ejercicio 1.7.** *Sean dados los vectores* **v**1 = (1*,* 1*,* 0)*T ,* **v**2 = (0*,* 1*,* 1)*T y*

**v**3 = (*−*1*,* 2*,* 1)*T . Estudiar si son Linealmente Independientes.*

#### Rango

Utilizando los conceptos de Dependencia e Independencia Lineal podemos definir el **rango de una matriz** como

**Definici´on 1.4.** *El rango de una matriz es el nu´mero m´aximo de vectores columna de la matriz que son Linealmente Independientes.*

Veremos que este nu´mero nos dar´a la dimensi´on del espacio imagen (*A*) de la aplicaci´on lineal asociada a la matriz. En general se tiene

*R*

**Definici´on 1.5.** *La Dimensi´on de un conjunto de vectores S viene dada por el rango de la matriz definida por los vectores de S.*

**Definici´on 1.6.** *Una matriz A ∈* R*m,n se dice de* **Rango M´aximo** *si*

*rank*(*A*) = m´ın(*m, n*)

Una manera r´apida de establecer la dependencia/independencia entre vec- tores consiste en generar la matriz con los vectores en columna y aplicar el algoritmo de reducci´on a forma escalonada por filas implementado en el comando de Matlab **rref(A)**. Este comando est´a basado en el proceso de

Eliminaci´on de Gauss que veremos en el contexto de la Factorizaci´on LU (Lower-Upper) de una matriz. De momento es suficiente observar que tras aplicar el algoritmo el primer elemento no nulo de cada fila de una matriz se llama **cabecera** de fila. Siempre es posible obtener **cabeceras unitarias** realizando operaciones elementales (combinaciones lineales entre las ecuacio- nes).

**Definici´on 1.7.** *Una matriz se define en forma reducida por filas si las cabeceras unitaria est´an ordenadas (la cabecera de la fila superior aparece siempre a la izquierda de la cabecera de las filas inferiores).*

En cada columna aparece una u´nica cabecera y los vectores columna con

cabecera son linealmente independientes.

**Ejercicio 1.8.** *Sean dados los vectores*

*v*1 = (4*, −*5*,* 7)*T , v*2 = (2*, −*3*,* 4)*T , v*3 = (1*,* 1*, −*2)*T , v*4 = (2*, −*1*,* 1)*T ,*

*Extraer el nu´mero m´aximo de vectores linealmente independientes.*

##### Conjuntos de Generadores, Bases y Bases Ortogonales

Mediante las definiciones de Dependencia e Independiencia Lineal podemos caracterizar los conjuntos de vectores en conjuntos de Generadores, Bases y Bases Ortogonales.

#### Conjuntos de Generadores

**Definici´on 1.8.** *Sea G un conjunto de m vectores G* = *{***v1***, ...,* **vm***} de* R*n*

*siendo m ≤ n. El sub-espacio V ⊂* R*n generado se denota por*

*V* = *L*[**v1***, ...,* **vm**] *⊂* R*n*

*y los infinitos vectores de V se generan mediante la operaci´on de combinaci´on lineal:*

*m*

*V* = *L*[**v1***, ...,* **vm**] = *{***w** *∈* R*n /* **w** = *αi***vi***, αi ∈* R*} ⊂* R*n*

*i*=1

Si los vectores son Linealmente Independiente el espacio *V* R*n* generado es de dimensi´on *m*. Si los vectores son Linealmente Dependientes la dimen- si´on de *V* ser´a *p < m* siendo *p* el nu´mero m´aximo de vectores linealmente independientes del conjunto.

*⊂*

**Ejemplo 1.6.** *Sean dados m* = 2 *vectores* **u**1 = ( 1*,* 1*,* 0)*T ,* **u**2 = (0*,* 1*,* 1)*T de* R3*, siendo n* = 3*. Calcular las ecuaciones del subespacio U* = *L*[**u**1**u**2] *generado y determinar su dimensi´on.*

*−*

Empezamos definiendo un elemento gen´erico del subespacio *U*

**v** = *α***u**1 + *β***u**2*, α, β ∈* R

En componentes se tiene el sistema de 3 ecuaciones lineales

*v*1 = *−α, v*2 = *α* + *β, v*3 = *β*

Eliminando par´ametros

*α* = *−v*1*, β* = *v*3

se deduce la ecuaci´on que define el subespacio (lineal):

*U* = *{***v** = (*v*1*, v*2*, v*3)*T /v*2 = *v*3 *− v*1*} ⊂* R3

Al ser un subespacio *U* R3 y al tener que satisfacer una ecuaci´on se deduce que la dimensi´on del subespacio *U* es dim(*U* ) = 3 1 = 2. Observando desde el comienzo que los dos vectores no son mu´ltiplos se pod´ıa deducir que son independientes luego generan un subespacio de dimensi´on 2.

*−*

*⊂*

#### Bases

Si *m > n* un conjunto de Generadores constituido por *m* vectores no puede ser Linealmente Independiente (es decir una base). Por ejemplo *m* = 3 vectores en dimensi´on *n* = 2 no pueden ser linealmente independientes.

**Definici´on 1.9.** *Se define a una Base B de un espacio Vectorial V como a un conjuntos de generadores Linealmente Independientes del espacio V .*

Suponiendo independencia lineal si *m* = *n* entonces el conjunto de vectores **v1***, ...,* **vm** constituye una base del espacio R*m*. La Dependencia Lineal es una forma de redundancia en la informaci´on. En una base no hay redundancia y el nu´mero *m* de vectores independientes define la dimensi´on del espacio generado.

#### Bases Ortogonales

**Definici´on 1.10.** *Si el producto escalar entre dos cualesquieras de los vec- tores de una base es nulo entonces la base es ortogonal:*

*⟨***vi***,* **vj***⟩* = **vi** *·* **vj** = 0*, i ̸*= *j*

*Si adem´as*

*la base es ortonormal.*

*⟨***vi***,* **vi***⟩* = 1*, ∀ i*

Veamos ahora como podemos construir f´acilmente una base de un espacio vectorial a partir de un conjunto de generadores. Para ello se utiliza el algo- ritmo de eliminaci´on de Gauss.

#### Formas escalonadas reducidas pr filas

Una manera muy directa consiste en construir la matriz cuyas columnas son los vectores del conjunto y luego aplicar el comando **rref(A)** para obtener una matriz escalonada por filas cuyas cabeceras definen los vectores indepen- dientes.

**Ejemplo 1.7.** *Sean dados los vectores*

**u**1 = (4*, −*5*,* 7)*T ,* **u**2 = (2*, −*4*,* 4)*T ,* **u**3 = (1*,* 1*, −*2)*T ,* **u**4 = (2*, −*1*,* 1)*T*

*Determinar el espacio generado, es decir L*[(**u**1*,* **u**2*,* **u**3*,* **u**4)]*. Extraer una base de* R3 *usando la forma de matriz escalonada por filas*

Utilizando la siguiente sintaxis definimos la matriz (en formato racional)

u\_1=[4,-5,7]’; u\_2=[2,-4,4]’;

u\_3=[1,1,-2]’; u\_4=[2,-1,1]’;

format rat

A=[u\_1 u\_2 u\_3 u\_4]

La Forma Escalonada Reducida por Filas se obtiene mediante

A\_red=rref(A)

y se deduce que los primeros 3 vectores de R3 son independientes luego

*L*[(**u**1*,* **u**2*,* **u**3*,* **u**4)] = R3*.*

Determinamos la combinaci´on lineal que permite escribir **u**4 mediante **u**1 y **u**2, **u**3 (Coeficientes o pesos de la Combinacion lineal) extrayendo la u´ltima columna de la matriz escalonada

c=A\_red(:,4)

#### Sistemas de coordenadas

Una vez introducidos los conceptos de base y base ortogonal podemos definir el de **coordenadas** en una base. Sea **x** = (*x*1*, ..., xn*)*T* un vector de R*n* en donde (*x*1*, ..., xn*) son las coordenadas en la base can´onica **e***i* = (0*, ..*1*, ..*0), con **e***i,i* = 1 y **e***i,j* = 0. Sea *B* = *{***v**1*, ...,* **v***n}* una base de R*n*. Entonces si definimos *B* = (**v**1*|...|***v***n*) como la matriz asociada a la base *B* las nuevas coordenadas de **x** en la base *B* vienen dadas por la (u´nica) soluci´on del sistema lineal

*B***y** = **x**

que es

**y** = *B−*1**x***.*

Esto quiere decir que *P* = *B−*1 es una matriz de paso o cambio de coordena- das del sistema can´onico a un sistema de coordenadas definido por . Este razonamiento se puede extender para pasar de un sistema de coordenadas asociado a la base *B* a uno asociado a la base *B′*.

*B*

**Ejemplo 1.8.** *Sean dadas las bases*

*BC* = *{*(1*,* 0)*T ,* (0*,* 1)*T }, B*1 = *{*(1*,* 0)*T ,* (1*,* 1)*T }, B*2 = *{*(*−*1*,* 1)*T ,* (0*,* 2)*T }*

*Escribir las bases en forma matricial. Calcular las matrices de paso*

*P* (*BC, B*1)*, P* (*B*1*, B*2)*, P* (*B*2*, BC*)

*Suponer tener un dato* **x**1 = (2*,* 2)*T en la base can´onica. Calcular sus coor- denadas en las bases B*1 *y B*2*.*

Empezando escribiendo las bases en forma matricial

 1 1   1 1 

    

 *−*1 0 

*BC* = 0 1 *, B*1 = 0 1 *, B*2 =

Para calcular la primera matriz de paso

*P*1 = *P* (*BC, B*1) = (**p**1*|***p**2)

1 2 

tenemos que escribir los vectores de *C* en t´erminos de los vectores de 1. Esto equivale a resolver los 2 sistemas

*B B*

*B*1**p**1 = **e1***, B*1**p**2 = **e2***,*

que se pueden escribir en forma compacta *B*1*P*1 = *Id* luego *P*1 = *B*1*−*1. Para calcular la matriz de paso

*P*2 = *P* (*B*1*, B*2) = (**p**1*|***p**2)

tenemos que escribir los vectores de 1 en t´erminos de los vectores de 2. Esto equivale a resolver los 2 sistemas

*B B*

*B*2**p**1 = **b**11*, B*2**p**2 = **b**12*,*

que se pueden escribir en forma compacta *B*2*P*2 = *B*1 luego *P*2 = *B*2*−*1*B*1. Finalmente, para calcular la matriz de paso

*P*3 = *P* (*B*2*, Bc*) = (**p**1*|***p**2)

tenemos que escribir los vectores de 2 en t´erminos de los vectores de *C*. Esto equivale a resolver los 2 sistemas

*B B*

*BC***p**1 = **b**21*, BC***p**2 = **b**22*,*

que se pueden escribir en forma compacta *BCP*3 = *B*2 luego *P*3 = *B*2.

Se tiene

*−*1  1 *−*1 

*−*1 

*−*1 *−*1 

*P*1 = *B*1 =  0 1  *, P*2 = *B*2 *B*1 =  1*/*2 1  *,*

siendo

 *−*1 0 

*P*3 = *B*2 = 

1 2 

Estas matrices, permiten obtener las coordenadas en las otras bases mediante simple proyecci´on. Dado por ejemplo **x**1 = (2*,* 2)*T* en la base can´onica si quiero conocer sus coordenadas **y***B*1 en *B*1 hago la proyecci´on

**y***B*1 = *P*1**x**1 = (0*,* 2)*T*

Si quiero conocer sus coordenadas **y***B*2 en *B*2 hago la proyecci´on

**y***B*2 = *P*2*P*1**x**1 = *P*2**y***B*1 = (*−*2*,* 2)*T*

**Ejercicio 1.9.** *Sean dadas las bases*

*BV* = *{*(1*,* 0)

*T*

*,* (1*,* 1)*T*

*}, BN* = *{*(*−*1*,* 1)

*,* (0*,* 2)*T }*

*Sea* **u** = (3*,* 7)*T un vector expresado en t´erminos de la base can´onica BC* =

*T*

*{*(1*,* 0)*T ,* (0*,* 1)*T }. Calcular las coordenadas* **x** *de* **u** *en la base BV . Definir la matriz de paso P* = *P* (*BV , BN* ) *y calcular las coordenadas* **y** *de* **u** *en la base BN .*

##### Bases Ortogonales

El tener una base ortogonal implica que un gen´erico vector **w** R*n* se escribe (codifica) de manera u´nica en esta base mediante sus componentes:

*∈*

*n*

**w** = *wi***v***i*

*i*=1

y se deduce por las propiedades de ortogonalidad y normalidad la f´ormula para los coeficientes (pesos) en una base ortonormal

*n*

*⟨***w***,* **v***i⟩* = **w** *·* **v***i* = *wi***v***i ·* **v***i* = *wi||***v***i||*2 = *wi*

*i*=1

Si la base es s´olo ortogonal la f´ormula general es

*w* = *⟨***w***,* **v***i⟩* R

*∈*

*i*

*||***v***i||*2

##### Bases Ortonormales

Se trata de bases ortogonales cuyos vectores tienen norma (longitud) unitaria. La base can´onica es el ejemplo m´as utilizado de base ortogonal. Mediante el algoritmo de ortonormalizaci´on de Gram-Schmidt de cualquier base se puede extraer (calcular) una base ortogonal. Este algoritm se basa en la factorizaci´on *QR* de una matriz que veremos en la pr´oxima clase.

##### Matrices y Transformaciones Lineales

Las matrices *A* R*m,n* se pueden identificar con Tranformaciones Lineales definidas por la *actuaci´on* (o efecto) que tiene la operaci´on de multiplicaci´on matriz por vector sobre todos los vectores de un espacio vectorial. En concreto se tiene:

*∈*

**Definici´on 1.11.** *Sean dadas las bases can´onicas de los espacios vectoria-*

*les* R*m y* R*n y sea dada una matriz A ∈* R*m,n. Entonces existe una u´nica*

*Transformaci´on Lineal definida, mediante las Bases can´onicas, en la forma*

*T* (**x**) = **Ax** *∈* R**m***, ∀* **x** *∈* R**n**

*La aplicaci´on as´ı construida, T* : R*n* R*m tal que T* (**x**) = **Ax** *es lineal ya que verifica la propiedad*

*→*

*T* (*α***x** + *β***y**) = *α***T**(**x**) + *β***T**(**y**)*, ∀ α, β ∈* R

**Ejemplo 1.9.** *Sea dada la aplicaci´on T* : R2 *→* R2 *definida por*

*T* (*x, y*) = (*x − y, x*2)

*Demostramos que no es lineal. Para ello sean* **u** = (*u*1*, u*2)*T ,* **v** = (*v*1*, v*2)*T y*

*α, β ∈* R*. Se tiene*

*T* (*α***u** + *β***v**) = (*αu*1 + *βv*1 *− αu*2 *− βv*2*,* (*αu*1 + *βv*1)2)

*Por otra parte*

*αT* (**u**) + *βT* (**v**) = (*αu*1 + *βv*1 *− αu*2 *− βv*2*, αu*2 + *βv*2)

1

1

*luego la aplicaci´on no es lineal ya que las segundas componentes son, en general, distintas:*

(*αu*1 + *βv*1)2 *̸*= *αu*2 + *βv*2

1

1

Por construcci´on, una aplicaci´on lineal siempre manda el cero al cero, es decir,

*T* (**0**) = **0**

Esta propiedad es necesaria pero no suficiente (como demuestra el ejemplo anterior) para que una aplicaci´on sea lineal.

**Ejercicio 1.10.** *Estudiar si las siguientes aplicaciones son lineales:*

*f* (*x, y*) = (*x − y, x* + 1)*, g*(*x, y*) = (*x − y, x*)

Respuesta: Se demuestra, aplicando la definici´on, que *g*(*x, y*) es lineal. Basta obseervar que es la aplicaci´on lineal asociada a la matriz

 1 *−*1 

 1 0 

Por otra parte *f* (0*,* 0) = (0*,* 1) = (0*,* 0) luego *f* no es lineal. Se verifica que *f*

*̸*

es una aplicaci´on afin:

 1 *−*1   *x*   0 

*f* (*x, y*) =  1 0   *y*

##### Espacio Imagen y Nu´cleo

 +  1 

Dada una matriz *A* R*m,n* podemos definir dos sub-espacios vectoriales que caracterizan el comportamiento de la matriz y de la aplicaci´on lineal asociada. Observamos que dada una matriz siempre podemos considerar la aplicaci´on lineal asociada *f* : R*n →* R*m* definida por

*∈*

*f* (**x**) = *A***x**

**Definici´on 1.12.** *Sea A* R*m,n una matriz y f* : R*n* R*m la aplicaci´on lineal asociada. Definimos el espacio imagen de f , im*(*f* )*, al subconjunto de* R*m dado por*

*∈ →*

*Im*(*f* ) = *R*(*A*) = *{***y** = *A***x** *∈* R*m |* **x** *∈* R*n} ⊂* R*m*

*El conjunto de vectores que pertenecen a la Imagen de f es un sub-espacio vectorial de* R*m cuya dimensi´on viene dada por el rango de A.*

**Definici´on 1.13.** *Sea A* R*m,n una matriz y f* : R*n* R*m la aplicaci´on lineal asociada. Definimos el espacio del nu´cleo de f , ker*(*f* ) *al subconjunto de* R*n dado por*

*∈ →*

*Ker*(*f* ) = *N* (*A*) = *{***x** *∈* R*n | A***x** = **0** *∈* R*m} ⊂* R*n*

*El conjunto de vectores que pertenecen al Nu´cleo (Kernel) es un sub-espacio vectorial de* R*n.*

La imagen de la transformaci´on lineal es el espacio imagen

*Im*(*f* ) = *R*(*A*)

y viene dado por todos los vectores **y** = R*m* tales que el sistema **y** = *A***x** es compatible. El espacio imagen viene generado por los vectores imagen de la base can´onica. El nu´cleo de la transformaci´on lineal es el espacio

*∈*

*ker*(*f* ) = *N* (*A*)

y viene definido por las soluciones del sistema de ecuaciones lineales

*f* (**x**) = *A***x** = **0***m*

Una f´ormula que utilizaremos para estudiar los sistemas lineales es la **F´ormu- la de la dimensio´n** que explica la relaci´on entre los sub-espacios imagen y nu´cleo:

**Teorema 1.1.** *Sea A* R*m,n una matriz y f* : R*n* R*m la aplicaci´on lineal asociada. Se tiene entonces*

*∈ →*

*n* = *dim R*(*A*) + *dim N* (*A*) (1)

Observa que d*im R*(*A*) *≤ m* (porqu´e es un sub-espacio). Lo mismo ocurre con el espacio nu´cleo de la aplicaci´on: d*im N* (*A*) *≤ n*.

**Definici´on 1.14.** *Sea A* R*m,n una matriz y f* : R*n* R*m la aplicaci´on lineal asociada. Si* (*A*) = **0** *entonces diremos que la aplicaci´on lineal es* **inyectiva***. Si* (*A*) = R*m diremos que la aplicaci´on lineal es* **suprayectiva***. Una aplicaci´on inyectiva y suprayectiva se dice* **biyectiva** *o isomorfismo.*

*R*

*N { }*

*∈ →*

En matlab existen los comandos **null(A)**, **orth(A)** que permiten calcular una base ortonormal de los subespacios vectoriales (*A*) y (*A*) respectiva- mente.

*N R*

**Ejercicio 1.11.** *Sea dada la matriz*

 1 *−*2 0 

*A* =  0 0 0 

2 *−*4 0

3*,*3

*Definir la aplicaci´on lineal asociada. Calcular la dimensi´on del espacio ima- gen* (*A*) *y del nu´cleo* (*A*)*. Calcular una base de los mismos. Determinar si la aplicaci´on es inyectiva, suprayectiva o biyectiva.*

*R N*

**Ejercicio 1.12.** *Sea dada la matriz*

*A* = 



2 1 

*−*1 1

1 0 

3*,*2

*Definir la aplicaci´on lineal asociada. Calcular la dimensi´on del espacio ima- gen* (*A*) *y del nu´cleo* (*A*)*. Calcular una base de los mismos. Determinar si la aplicaci´on es inyectiva, suprayectiva o biyectiva.*

*R N*

##### Normas

Las normas generalizan el concepto de m´odulo de un vector y permiten medir matrices y distancias entre ellas. Podemos por tanto medir la distancia entre dos im´agenes o calcular el residuo (error) cometido al realizar num´ericamente una descomposici´on o una factorizaci´on. Las normas aparecen en la definici´on de los funcionales de energ´ıa en c´alculo variacional y en las funciones de p´erdida en aprendizaje autom´atico. Empezamos por las normas asociadas a los espacios de Lebesgue que veremos m´as adelante, en c´alculo variacional. Se conocen con el nombre de *normas p*. En el contexto del c´alculo num´erico, discreto, se tiene:

**Definici´on 1.15.** *La norma p de un vector* **x** R*n se calcula mediante la forma*

*∈*

*||***x***||p* =

*i*

*|xi|*

1*/p*

*, ∀ p ∈* R*, p ≥* 1

Para medir la longitud de un vector, en cualquier dimensi´on se utiliza la norma *p*. Dependiendo del valor de *p* se obtienen distintas mediciones. El concepto de norma es m´as general. Cualquier campo escalar que cumpla con las siguientes 3 propiedades es una norma.

*p*

**Definici´on 1.16.** *Sea f* : R*n* R *una funci´on (campo escalar) que verifica las siguientes propiedades:*

*→*

*1.*

*f* (**x**) = 0 *⇔* **x** = **0** *∈* R**n**

*2.*

*f* (**x** + **y**) *≤ f* (**x**) + *f* (**y**)*, ∀* **x***,* **y** *∈* R*n*

*3.*

*f* (*α***x**) = *|α|f* (**x**)*, ∀ α ∈* R*, ∀***x** *∈* R*n*

*Entonces f es una norma.*

Para *p* = 2 tenemos la cl´asica norma Eucl´ıdea o norma 2 denotada normal- mente sin el sub-´ındice.

*||***x***||*2 = *||***x***||* =

*i*

*|xi|*

2

1*/*2

Para *p* = 1 tenemos la norma 1

*||***x***||*1 = *|xi|*

*i*

Para *p* = *∞* tenemos la norma *∞* o norma del m´aximo

*||***x***||∞* = m´ax *|xi|*

*i*

En Matlab el comando **norm** calcula la norma de una matriz o vector. Con- cretamente se tiene:

*||***x***||*2 = **norm**(**x***,* 2) = **norm**(**x**)*,*

*||***x***||*1 = **norm**(**x***,* 1)*,*

*||***x***||∞* = **norm**(**x***,* Inf)*.*

Un vector es unitario si

*||***x***||* = 1

El conjunto de todos los vectores unitarios define la Bola (disco) unitario del espacio considerado. Si todos los vectores de una base ortogonal son unitarios entonces la base se dice ortonormal. Una base ortogonal siempre se puede ortonormalizar mediante el procedimiento de Gram-Schmidt.

Es importante observar que una norma induce siempre una forma de medir, es decir una m´etrica o distancia.

**Definici´on 1.17.** *Sean dados dos vectores* **x***,* **y** *en el espacio vectorial V* =

R*n.*

*La distancia entre ellos se define por*

*d*(**x***,* **y**) = *||***x** *−* **y***||*

*de donde la longitud de un vector que es la distancia del punto (vector) al origen:*

*d*(**x***,* **0**) = *||***x***||*

Muy a menudo se utiliza la norma al cuadrado de un vector lo que que representa la **Energ´ıa** del vector.

**Definici´on 1.18.** *La energ´ıa de un vector se calcula a trav´es del producto escalar del vector consigo mismo:*

*⟨***v***,* **v***⟩* = **v** *·* **v** = *||***v***|| , i ̸*= *j*

2

M´as en general podemos escribir la operaci´on de producto escalar mediante la definici´on de norma:

*⟨***u***,* **v***⟩* = **u** *·* **v** = **u***T* **v** = **v***T* **u** = *||***u***||*2*||***v***||*2 cos(*θ*) *∈* R

Si queremos medir una matriz *A* R*m,n* y definir una distancia en el espacio de las matrices, en aprendizaje autom´atico se utiliza a menudo la norma de Frobenius definida por

*∈*

*||A||F* =

*i,j*

*|ai,j|*

1*/*2

Observamos que la norma de Frobenius es la norma eucl´ıdea del vector ge- nerado concatenando las componentes de la matriz como un vector *mn* di- mensional. Se tienen las siguientes definiciones de las normas matriciales m´as usadas

2

*||A||*1 = m´ax

*j*

*m*

*i*=1

*|ai,j|*

*, ||A||∞* = m´ax

*n*

*j*=1

*|ai,j|*

*||A||*2

= m´ax

*i*

*|A***x** *·* **y***|*

En Matlab se tiene:

**x***̸*=**0***,* **y***̸*=**0** *||***x***||*2*||***y***||*2

*||A||*2 = **norm**(*A,* 2) = **norm**(*A*)

*||A||*1 = **norm**(*A,* 1)

*||A||∞* = **norm**(*A,* Inf)

*||A||*fro = **norm**(*A,* fro)

**Ejercicio 1.13.** *Sean dadas las im´agenes de Lena y del Cameraman dis- ponibles en el campus virtual. Contaminar la imagen de Lena utilizando el modelo de ruido gaussiano con los valores del par´ametro λ* = 1*, λ* = 0*.*1 *y λ* = 0*.*01*. Calcular la distancia de Frobenius entre las im´agenes contamina- das y la original. Calcular la distancia entre las im´agenes originales de Lena y del Camaraman. ¿Que conclusiones puedes obtener ?*

**Ejercicio 1.14.** *Sea dada la imagen en color cadiz*2015*.jpg. Pasarla a doble precisi´on en el intervalo* [0*,* 1] *y a una imagen en escala de grises mediante el comando* **rgb2gray.m***. Calcular sus normas* 1*,* 2*, y la de Frobenius. Calcular la energ´ıa*

*∞*

*E* = 1 *||***x***||*2 = 1 *|x |*2

2

2

2

*i*

*i*



Figura 8: C´alculo de la energ´ıa de la imagen en escala de grises.

## IV Clase

Empezaremos recordando las definiciones de algunas matrices especiales por su simple estructura. Luego definiremos las formas cuadr´aticas y su clasi- ficaci´on lo que es fundamental en problemas de optimizaci´on. Finalmente, pasaremos a la definici´on y resoluci´on de un problema fundamental en el contexto del an´alisis estructural de una matriz, el Problema de Autovalores.

#### Tipos Especiales de Matrices y formas cuadr´aticas

*̸*

Es conveniente conocer algunos tipos especiales de matrices ya que son par- ticularmente u´tiles computacionalmente debido a su estructura. En primer lugar las **matrices diagonales** *D ∈* R*n,n* que se caracterizan por las ecuacio- nes *ai,j* = 0 si *i* = *j*. La **matriz nula** es una matriz diagonal con ceros en la diagonal. Se genera mediante el comando **zeros(n)**. La **matriz identidad** es una matriz diagonal. Se genera en matlab mediante el comando **eye(n)**. Dado un vector **v** R*n* definimos por **diag(v)** a la matriz cuadrada cu- ya diagonal viene dada por las componentes del vector. El mismo operador, aplicado a una matriz en la forma **diag(A)** extrae de la matriz un vector for- mado por los elementos de la diagonal. Multiplicar un vector por una matriz diagonal es computacionalmente eficiente

d*iag*(**v**)**x** = **v** *⊙* **x**

*∈*

y corresponde a rescalar cada componente *xi* por el escalar *vi*. Tambi´en es eficiente invertir una matriz diagonal

d*iag*(**v**)*−*1 = d*iag*([1*/v*1*,* 1*/vn*])

lo que es posible s´olo si *vi* = 0 *i*. Fundamentales son las matrices triangu- lares inferiores y superiores que tienen ceros por arriba (respectivamente por abajo) de la diagonal principal. Los comando que extraen de una matriz *A* su parte triangular inferior y superior son **tril**(A) y **triu**(A) respectivamente. Si la matriz es no cuadrada el mismo comando nos da matrices trapezoida- les. Otra estructura muy im portante es la de **matriz sim´etrica** definida mediante la operaci´on de trasposici´on.

*̸ ∀*

**Definici´on 2.1.** *Sea A* = (*ai,j*) *una matriz cuadrada. Diremos que A es Sim´etrica si ai,j* = *aj,i, es decir si se verifica la ecuaci´on matricial*

*AT* = *A*

Otras matrices cuya estructura es particularmente relevante son las **matri- ces ortogonales** en donde todos los vectores filas (respec. columna) son mutuamente ortogonales. Si adem´as tienen norma unitaria tendremos una matriz ortonormal.

**Definici´on 2.2.** *Sea A* = (*ai,j*) *una matriz cuadrada. Diremos que A es Ortogonal si se verifica la ecuaci´on matricial*

*AT A* = *AAT* = *In*

Una propiedad importante que se deduce a partir de la unicidad de la matriz inversa es que podemos invertir una matriz ortogonal simplemente traspo- niendo sus componentes:

*A−*1 = *AT*

Otras matrices especiales, cuadradas, cuyo reconocimiento permitir´a asegurar la convergencia de muchos algoritmos son las **definidas Positivas**. Tienen un papel fundamental en la teor´ıa de la optimizaci´on Convexa.

##### Formas Cuadr´aticas

Empezamos definiendo la forma cuadr´atica asociada a una matriz cuadrada.

**Definici´on 2.3.** *Sea A ∈* R*n,n una matriz cuadrada. Diremos que la forma cuadr´atica q* : R*n →* R *definida por*

*q*(**x**) = **x***T A***x**

*est´a asociada a la matriz A.*

Observa que podemos escribir equivalentemente:

*n n*

*q*(**x**) = **x***T A***x** = *Ai,jxixj*

*i*=1

*j*=1

**Ejemplo 2.1.** *Sea dada la matriz cuadrada*

*A* = 2 3

3 *−*1

*Calculamos la forma cuadr´atica asociada*

2*,*2

*siendo* **x***T* = (*x, y*)*. Se tiene*

*q*(**x**) = *x y* 2 3

*q*(**x**) = **x***T A***x**

*x* = *x y* 2*x* + 3*y* =

3 *−*1

*y*

3*x − y*

= 2*x*2 + 3*xy* + 3*xy − y*2 = 2*x*2 + 6*xy − y*2

**Ejercicio 2.1.** *Sea dada la matriz cuadrada*

1 *−*2 0

 

*A* =  0 0 0 

2 *−*4 0

*Calcular la forma cuadr´atica asociada.*

3*,*3

Podemos ahora definir las matrices Definidas Positivas a trav´es del signo de la forma cuadr´atica asociada

**Definici´on 2.4.** *Sea A ∈* R*n,n. Diremos que A es* **Definida Positiva** *si la forma cuadr´atica q* : R*n →* R *definida por q*(**x**) = **x***T A***x** *verifica:*

*q*(**x**) = **x***T A***x** *>* 0*, ∀* **x** *∈* R*n,* **x** *̸*= **0**

Definiciones an´alogas se obtienen para matrices **Semi-Definidas Positivas**

que son siempre no negativas

**x***T A***x** *≥* 0*, ∀* **x** *∈* R*n,* **x** *̸*= **0**

Por otra partte las matrices **Definidas Negativas** se caracterizan por la desigualdad

**x***T A***x** *<* 0*, ∀* **x** *∈* R*n,* **x** *̸*= **0**

y la **Semi-Definidas Negativas** se obtienen relajando la desigualdad ante- rior

**x***T A***x** *≤* 0*, ∀* **x** *∈* R*n,* **x** *̸*= **0**

Finalmente, cuando ninguna de las desigualdades anteriores es cierta se tienen las matrices **Indefinidas**. Se observa que la matriz que define una forma

cuadr´atica no tienen porqu´e ser sim´etrica. Sin embargo no hay p´erdida de generalidad considerando que es sim´etrica. Esto es as´ı porqu´e

*⟨***x***, A***x***⟩* = **x** *A***x** = **x** *B***x** = *⟨***x***, B***x***⟩*

*T T*

si

*B* = 1 (*A* + *AT* )*,*

2

es decir si *B* es la parte sim´etrica de *A*.

Observa que la notaci´on de producto escalar introducida en la igualdad **x***T A***x** = **x***T B***x** permite obtener este resultado utilizando la propiedad de simetr´ıa del producto escalar (una funci´on bilineal sim´etrica).

Excepto algunos casos evidentes no es f´acil determinar el car´acter de una forma cuadr´atica. Sin embargo un criterio muy simple se obtiene calculando los autovalores de la matriz, lo que estudiaremos a continuaci´on. De momento vamos a considerar la interpretaci´on gr´afica de estos conceptos que se puede obtener f´acilmente mediante Matlab u Octave.

**Ejercicio 2.2.** *Sean dadas las formas cuadr´aticas*

*q*1(*x, y*) = 2*x*2 + 4*xy* + 2*y*2*, q*2(*x, y*) = *−*2*x*2 + 4*xy −* 2*y*2*, q*3(*x, y*) = *−*2*x* + 4*xy* + 2*y , q*4(*x, y*) = 2*x* + 4*y*

2 2 2 2

*Clasificarlas utilizando m´etodos gr´aficos.*

E´stas consideraciones ser´an fundamentales a la hora de optimizar formas cuadr´aticas o las funciones aproximadas localmente por ellas mediante los Desarrollos de Taylor de II orden.

La gr´afica de las formas cuadr´aticas se puede obtener con los comandos

A=[-2 4 ; 0 2 ]; B=(1/2)\*(A+A’);

syms x y real X=[x y ]’;

q=expand(X’\*B\*X)

Z=0\*x+0\*y % plano de ecuacion z=0 figure

fsurf(q,[-1,1,-1,1],’MeshDensity’,100) hold on

fsurf(Z,[-1,1,-1,1],’FaceAlpha’,0.2,’MeshDensity’,40) xlabel(’x’), ylabel(’y’)

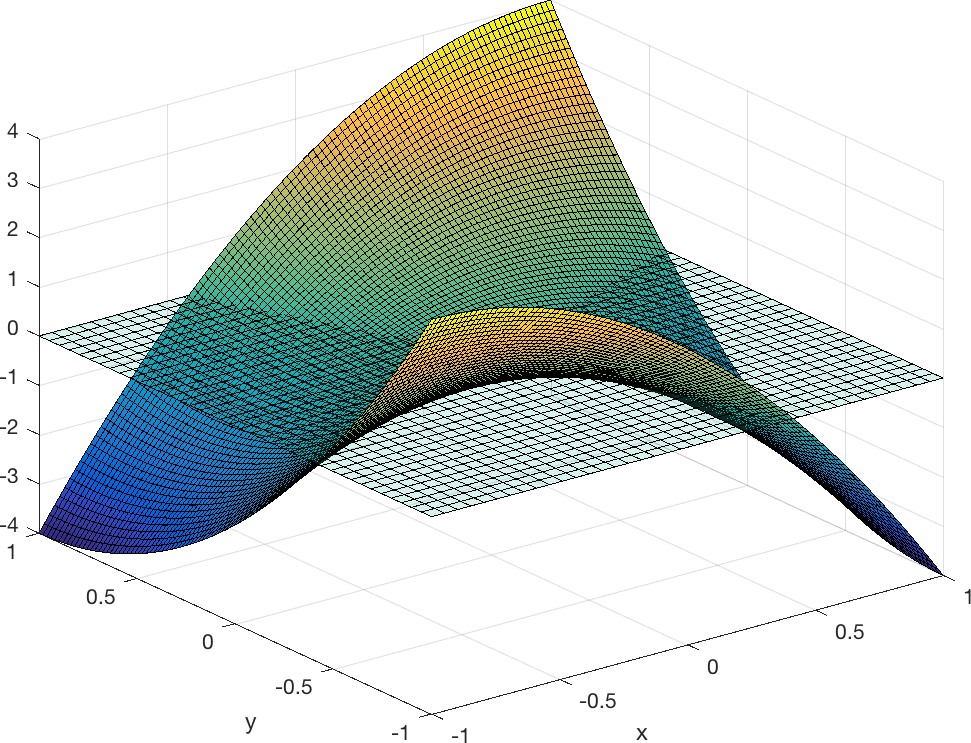


Figura 9: Se verifica gr´aficamente que *q*3(*x, y*) es una forma cuadr´atica inde- finida ya que toma valores por encima y por debajo del plano de ecuaci´on *z* = 0. Observa que esta funci´on no tiene m´ınimos ni m´aximos.

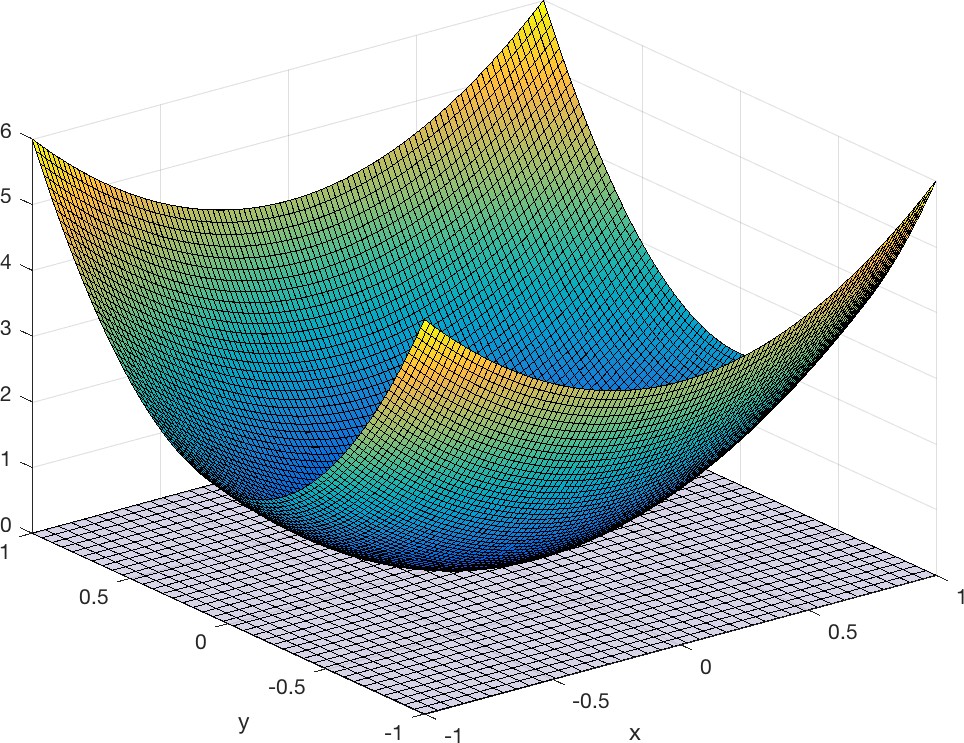


Figura 10: Se verifica gr´aficamente que *q*4(*x, y*) = 2*x*2 + 4*y*2 es una forma cuadr´atica definida positiva ya que toma valores por encima o iguales a los del plano de ecuaci´on *z* = 0. Observa que esta funci´on tiene un u´nico punto de m´ınimo absoluto.

en donde hemos simetrizado la matriz de la formas cuadr´atica. Observa como, de manera simb´olica a trav´es del comando

syms x y real

podemos definir las dos variables reales (*x, y*), almacenarlas en un vector

*X* = (*x, y*) *∈* R2 y calcular explicitamente la forma cuadr´atica mediante

q=expand(X’\*B\*X)

Al querer representar un campo escalar *q*(*x, y*) de varias (2) variables utili- zamos para su gr´afica el comando

fsurf(q,[-1,1,-1,1],’MeshDensity’,100)

en un dominio definido *Dq* = [ 1*,* 1*,* 1*,* 1] y con una densidad de puntos prefijada. El comando **FaceAlpha** confiere transparencia a la gr´afica del plano que se muestra para estudiar el signo de la forma cuadr´atica.

*— −*

#### El Problema de Autovalores

**Definici´on 2.5.** *Dada una matriz cuadrada A ∈* R*n,n el Problema de au- tovalores consiste en encontrar un nu´mero escalar λ ∈* R *(que puede ser complejo) y un vector no nulo* **x** *∈* R*n,* **x** *̸*= **0** *tales que*

*A***x** = *λ***x**

*Llamaremos λ* **autovalor** *y* **x autovector***. El conjunto de autovalores de una matriz se llama* **espectro** *y se denota por*

*σ*(*A*) = *{λ*1*, ..., λn}*

*La cantidad ρ*(*A*) *es el* **Radio Espectral** *definido por el autovalor de m´odulo m´aximo:*

*ρ*(*A*) = m´ax*{|λi|}*

*i*

Los autovectores definen **direcciones invariantes** de la trasformaci´on lineal asociada a la matriz. Si **x** es autovector entonces *α***x**, *α* = 0 es autovector. Observa que puede existir el autovalor nulo *λ* = 0 pero no existe el auto- vector nulo, por definici´on. Los autovalores se pueden calcular resolviendo la **ecuaci´on caracter´ıstica** (polin´omica) de grado *n*:

*̸*

*pA*(*λ*) = det(*A − λI*) = 0 (2)

La teor´ıa de los sistemas lineales permitir´a deducir la ecuaci´on ([2](#_bookmark51)) para la caracterizaci´on de los autovalores. De momento observamos que la ecuaci´on

([2](#_bookmark51)) es **no lineal** y no existen f´ormulas anal´ıticas, exactas para su resoluci´on cuando el grado del polinomio es elevado. M´as en general las ecuaciones no lineales se resolver´an mediante algoritmos num´ericos que nos dar´an una soluci´on aproximada del problema. Al ser polin´omica sin embargo existen resultados te´oricos que utilizaremos a continuaci´on sobre el nu´mero y signo

de sus ra´ıces reales. Por el **Teorema Fundamental del A´lgebra** se tiene:

**Teorema 2.1.** *Toda matriz cuadrada de orden n con coeficientes complejos tiene n autovalores, reales o complejos, iguales o distintos.*

**Definici´on 2.6.** *El nu´mero de veces que un autovalor λ aparece como ra´ız del polinomio caracter´ıstico es la* **multiplicidad algebraica** *y se denota por mA*(*λ*)*.*

Asociado a cada autovalor hay un **autoespacio** definido por:

**Definici´on 2.7.** *Sea A una matriz cuadrada y sea λ un autovector de A. El autoespacio asociado al autovalor se define mediante el nu´cleo de la aplicaci´on lineal asociada a la matriz A−λI, siendo I la matriz identidad. En f´ormulas*

*E*(*λ*) = *N* (*A − λI*) *⊂* R*n*

Para calcular los autovectores que est´an en cada autoespacio hay que calcular una base del subespacio *E*(*λ*) = *N* (*A − λI*).

**Definici´on 2.8.** *La dimensi´on de cada autoespacio,*

*dim*(*E*(*λ*)) = *dim*(*N* (*A − λI*))

*nos da la* **multiplicidad geom´etrica** *mG*(*λ*) *del autovalor.*

Siempre se tiene que

*mG*(*λ*) *≤ mA*(*λ*)

Es decir la multiplicidad geom´etrica nunca puede ser mayor que la algebraica.

**Ejercicio 2.3.** *Sea dada la matriz*

 1 2 2 

 

*A* = *−*1 4 1

*−*2 2 5

3*,*3

*Calcular su espectro y la multiplicidad algebraica y geom´etrica de cada auto- valor. Encontrar explicitamente las ecuaciones que definen a los autoespacios. Determinar su* **factorizaci´on espectral**

*AV* = *V D*

Respuesta Hay 2 autoespacios asociados a los autovalores *λ*1 = 3 (de multi- plicidad algebraica 2) y *λ*2 = 4. Se tiene

*E*(*λ*1) = *E*(3) = *{*(*x, y, z*)*/x* = *y* + *z},*

de dimensi´on 2 luego la multiplicidad geom´etrica es 2. Por otra parte

*E*(*λ*2) = *E*(4) = *{*(*x, y, z*)*/x* = 2*y, z* = 2*y}*

tiene dimensi´on 1 luego multiplicidad geom´etrica es 1. Hay diagonalizabili- dad.

El c´alculo de los autovalores se realiza mediante un algoritmo iterativo que est´a basado en la factorizaci´on QR (que presentaremos m´as adelante) que se obtiene utilizando el algoritmo de ortonormalizaci´on de Gram-Schmidt. En Matlab el m´etodo *QR* est´a implementado en la funci´on **eig**. La sint´axis es: *D* = **eig**(*A*) para obtener un vector *D* que contiene todos los autovalores de

*A*. En la forma

[*V, D*] = **eig**(*A*)

obtenemos tambi´en la matriz *V* cuya columnas son los autovectores de *A* y que satisface la relaci´on

*AV* = *V D*

a la que llamaremos **Factorizaci´on espectral** para la matriz cuadrada *A*.

##### Diagonalizabilidad

El conocimiento del espectro de una matriz permite resolver el problema de la diagonalizabilidad. No todas las matrices cuadradas son diagonalizables.

**Definici´on 2.9.** *Sea A una matriz cuadrada de orden n, A* R*n,n. Diremos que A es* **Diagonalizable** *si* **existe** *una matriz cuadrada no singular V tal que*

*∈*

*V −*1*AV* = *D* = *diag*(*λ*1*, ..., λn*)

La matriz *V* se construye almacenando los autovectores que se correspon- den con los autovalores que est´an en la diagonal de la matriz *D* siguiendo

exactamente el mismo orden. En concreto sea *λi* un autovalor de multiplici- dad algebraica *mA*. Determinaremos sus autovectores asociados resolviendo el sistema cuadrado homog´eneo

*B*(*λi*)**x** = (*A − λiI*)**x** = **0** (3)

e imponiendo la existencia de infinitas soluciones del sistema [3](#_bookmark53). La dimensi´on del espacio de soluciones es la muliplicidad geom´etrica *mG* del autovalor.

**Teorema 2.2.** *Sea A una matriz cuadrada de orden n, A* R*n,n. Una* **con- dici´on suficiente** *para la diagonalizabilidad es que la matriz tenga n auto- valores reales y distintos.*

*∈*

Esto es porqu´e autovalores distintos tienen autovectores linealmente inde- pendientes.

**Ejercicio 2.4.** *Sea dada la matriz*

 *−*2 *−*2 3 

*A* =  *−*5 1 3 

*−*6 0 5

3*,*3

*Demostrar que es diagonalizable. Calcular las matrices V y D tales que*

*A* = *V DV −*1*.*

**Teorema 2.3.** *Una* **condici´on necesaria y suficiente** *para la diagonali- zabilidad es que*

*mA* = *mG*

El teorema afirma que la multiplicidad algebraica *mA* de cada autovalor (nu´mero de veces que aparece como ra´ız del polinomio caracter´ıstico) coincida con su multiplicidad geom´etrica *mG* (dimensi´on del autoespacio). Diremos entonces que la matriz es diagonalizable. La matriz *V* tiene por columnas los autovectores de *A* que forman una base del espacio. En efecto se tiene

**Teorema 2.4.** *Toda matriz A de dimensi´on n y coeficientes reales es diago- nalizable si, y s´olo si, existe una base de* R*n formada por autovectores.*

**Ejemplo 2.2.** *Demostrar que la matriz A es diagonalizable siendo*

0 0 2

 

*A* =  0 2 0 

2 0 0

3*,*3

La matriz es sim´etrica y tiene s´olo 2 autovalores distintos:

*λ*1 = *−*2*, λ*2 = 2

el segundo con multiplicidad algebraica *m*2 = 2. Los autovectores asociados al segundo autovalor son del tipo

**v**2 = (*a, b, a*)*T* = *a*(1*,* 0*,* 1) + *b*(0*,* 1*,* 0)

luego hay diagonalizabilidad ya que el auto-espacio asociado tiene dimensi´on 2 que es la multiplicidad geom´etrica de *λ*2.

**Ejemplo 2.3.** *Establecer si la matriz A es diagonalizable siendo*

*A* = 



1 0 0 

*−*1 1 3

2 2 2 

3*,*3

La matriz no es diagonalizable. Razonamos la respuesta. Calculando los au- tovalores (a mano o con el algoritmo **eig** de Matlab) se obtiene

y se observa en la matriz *D*

*λ*1 = 1*, λ*2 = 4

4 0 0

 

*D* =  0 1 0 

0 0 1

3*,*3

que el primero, *λ*1 = 1, es *doble*, es decir con multiplicidad algebraica *ma*(*λ*1) = 2 y el segundo, *λ*2 = 4 es *simple*, con multiplicidad algebraica *ma*(*λ*2) = 1. Los autovectores asociados al segundo autovalor (que es simple) son del tipo

**v**2 = (0*, a, a*)*T , a ∈* R

y las multiplicidades geom´etrica y algebraica coinciden y dim(*E*(*λ*2)) = 1 (di- mensi´on del autoespacio). Esto siempre ocurre cuando el autovalor es simple. De hecho no hace falta siquiera comprobarlo. Si el autovalor es simple enton- ces el autoespacio tiene dimensi´on 1.

Los autovectores asociados al primer autovalor (que es doble) son del tipo

**v**1 = (0*, −*2*a, a*)*T , a ∈* R

luego el auto-espacio *E*(*λ*1) tiene dimensi´on 1, dim(*E*(*λ*1)) = 1, no se puede obtener una base de autovectores y no hay diagonalizabilidad.

**Ejercicio 2.5.** *Demostrar que la matriz A no es diagonalizable siendo*

*A* = 1 1

0 1

2*,*2

*Comprobar que la matriz V dada por la descomposici´on espectral no es in- vertible.*

La definici´on de diagonalizabilidad se puede interpretar diciendo que la ma- triz *A* es **diagonalizable** si es **semejante** a una matriz **diagonal**. En efecto se tiene la siguiente definici´on de **Semejancia**:

**Definici´on 2.10.** *Dos matrices cuadradas A, B, de las mismas dimensiones se dicen* **semejantes** *y se denota por*

*A ∼ B*

*si existe una matriz no singular P tal que*

*P−*1*AP* = *B*

Todas las matrices semejantes tienen el mismo polinomio caracter´ıstico y por tanto el mismo espectro (los mismos autovalores).

**Ejercicio 2.6.** *Sean dadas las matrices*

 *−*3 *−*2 1   *−*4 *−*1 *−*1 

*A*1 = 



1 *−*6 1

1 *−*2 *−*3

3*,*3

*, A*2 = *−*1 *−*5 1

*−*1 0 *−*3

3*,*3

*Estudiar su diagonalizabilidad.*

Para matrices sim´etricas se tiene un resultado m´as fuerte

**Teorema 2.5.** *Toda matriz sim´etrica de coeficientes reales es ortogonalmente diagonalizable y sus valores propios son reales.*

##### Formas Cuadr´aticas

Otra aplicaci´on importante del espectro de una matriz es que permite clasi- ficar la forma cuadr´atica asociada

*q*(**x**) = **x***T A***x**

En las secciones anteriores v´ımos como pod´ıamos hacerlo por m´etodos gr´afi- cos. Ahora veremos como hacerlo con m´etodos anal´ıticos y num´ericos utili- zando la teor´ıa del problema de autovalores. En t´erminos del espectro de una matriz se tienen las siguientes definiciones

**Definici´on 2.11.** *Diremos que una matriz cuadrada A es* **Definida Posi- tiva** *si su espectro es positivo. Es* **Semi-Definida Positiva** *si su espectro es no negativo (al menos un autovalor es nulo). Definiciones an´alogas se obtienen para matrices* **Semi-Definidas Positivas***, en donde*

**x***T A***x** *≥* 0

*y los autovalores son todos no negativos, las* **Definidas Negativas***, en donde*

**x***T A***x** *<* 0

*y los autovalores son todos negativos, las* **Semi-Definidas Negativas** *en donde*

**x***T A***x** *≤* 0

*y los autovalores son todos no positivos y las* **Indefinidas** *que no tienen signo definido (cuando hay autovalores positivos y negativos).*

Veremos en la parte de optimizaci´on que esta informaci´on es muy u´til ya que una matriz Definida Positiva genera una forma cuadr´atica

*q*(**x**) = **x***T A***x**

que es estrictamente convexa y tendr´a por tanto un u´nico m´ınimo global.

**Ejercicio 2.7.** *Sea dada la matriz*

 10 2 1 

*A* = 

2 1 0

1 0 10

3*,*3

*Demostrar que es sim´etrica y definida positiva. Calcular una base de auto- vectores ortonormal. Dado el vector* **u** = (1*,* 0*,* 1)*T calcular sus coordenadas en la base ortonormal.*

##### Operador Traza y Determinante

El operador Traza es un campo escalar que calcula la suma de los elementos diagonales de una matriz cuadrada.

**Definici´on 2.12.** *Sea A ∈* R*n. Se define la Traza de A por*

**Tr**(*A*) = *ai,i*

*i*

La Traza de una matriz cuadrada se puede calcular en Matlab mediante el comando **trace(A)**. El operador Traza es invariante frente a la operaci´on de trasposici´on:

**Tr**(*A*) = **Tr**(*AT* )

Permite una definici´on alternativa de la Norma de Frobenius de una Matriz

*||A||F* = (**Tr**(*AA*

*T*

))1*/*2

El operador Determinante de una matriz cuadrada *A* R*n,n* es otro campo escalar con numerosas aplicaciones y se define de forma recursiva mediante la *Regla de Laplace*.

*∈*

**Definici´on 2.13.** *Sea A* R*n,n. El determinante de la matriz se define mediante la expresi´on*

*∈*

 *a*1*,*1 *n* = 1

*det*(*A*) = *n*

*j*=1

∆*i,jai,j, n >* 1*, ∀ i* = 1*..n*

 ¿

*siendo* ∆*i,j los co-factores de la matriz A que se calculan mediante la f´ormula*

∆*i,j* = (*−*1)*i*+*jdet*(*Ai,j*)

*y donde Ai,j denota la matriz que se obtiene de A eliminando su i-´esima fila y j-´esima columna.*

El algoritmo correspondiente a la f´ormula recursiva tiene complejidad fac- torial 2(*n* + 1)! y s´olo ser´ıa utilizable para matrices de pequen˜a dimensi´on. Un algoritmo mucho m´as eficiente y que se usa en la pr´actica (es la base del algoritmo que se implementa en Matlab, **det(A)**) se obtiene mediante la t´ecnica de **Factorizaci´on LU** o proceso de **Eliminaci´on de Gauss** que ser´a presentado m´as adelante. Su complejidad es de orden polin´omico: *n*3. El c´alculo del determinante de una matriz cuadrada permite establecer si esta es invertible.

**Definicion 2.14.** *Sea A* R*n,n una matriz cuadrada. Si det*(*A*) = 0 *la matriz se dice* **singular** *y no es por tanto invertible. Si det*(*A*) = 0 *la matriz es* **invertible***.*

*̸*

*∈*

La condicion de anulaci´on del determinante se utiliza para forzar que el siste- ma [3](#_bookmark53) tenga infinitas soluciones. Otra importante propiedad del determinante es que si *A* = *BC* entonces

det(*A*) = det(*B*)det(*C*)

Conociendo el espectro de una matriz, es decir sus autovalores, podemos calcular el determinante y la traza mediante las f´ormulas:

*n n*

det(*A*) = rr *λi,* **Tr**(*A*) = *λi*

*i*=1

*i*=1

y tambi´en las normas espectral (la norma eucl´ıdea para matrices cuadradas)

*||A||*2 = ✓*λ*m´ax(*AT A*)*,*

y de Frobenius

*||A||F* =

*n*

*i*=1

1*/*2

2

*λ*

*i*

##### Condicionamiento de un Sistema

Si conocemos los autovalores m´aximo y m´ınimo de una matriz (que define un sistema) podemos calcular una constante llamada **nu´mero de condici´on espectral** de la matriz definida por

*K*(*A*) = *λ*m´ax

*λ*m´ın

Este nu´mero se obtiene en Matlab mediante el comando **cond**(*A*). **Ejercicio 2.8.** *Sea dada la matriz de Hilbert definida por*

1

*ai,j* = *i* + *j −* 1 *, i, j* = 1*...n*

*de orden n que se puede construir con el comando* **hilb(n)***. Calcular el condi- cionamiento de la matriz de hilbert de orden n para n* = 1*..*100 *y representarl gr´aficamente su comportamiento.*

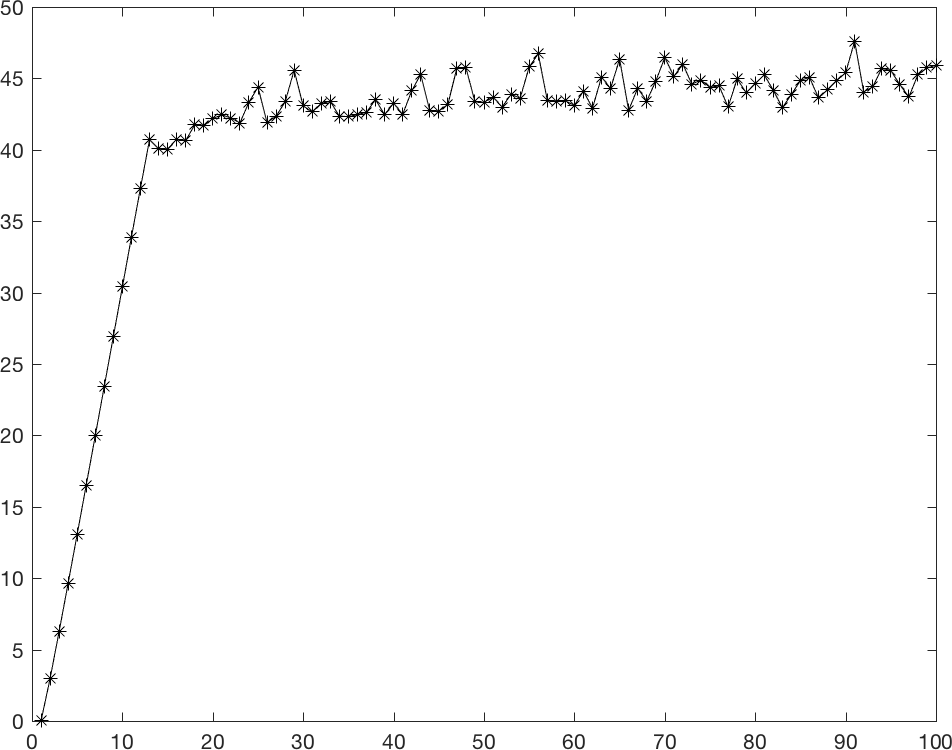


Figura 11: C´alculo del condicionamiento de la matriz de hilbert. Represen- taci´on en escala lorgar´ıtmica

##### Factorizaci´on de Cholesky y Factorizaci´on QR

Para un tipo especial de matrices es posible realizar una factorizaci´on muy u´til computacionalmente.

**Definici´on 2.15.** *Sea A una matriz cuadrada. Si A es* **sim´etrica** *y* **definida positiva** *existe una* **Factorizaci´on de Cholesky** *en la forma*

*A* = *HHT*

*en donde H es una matriz triangular inferior con elementos positivos en la diagonal.*

El coste computacional de esta factorizaci´on es del orden *n*3*/*3 (la mitad de una Factorizaci´on *LU* ). Este algoritmo est´a implementado en Matlab me- diante el comando **chol**(**A**) que nos da como output la matriz triangular superior *HT* . De manera informal podemos decir que estamos calculando la ra´ız cuadrada de una matriz siempre que sea sim´etrica y definida positiva.

**Ejercicio 2.9.** *Sean dadas las matrices*

2 1

= *, A*

*A*1 2

1 5

= 2 1 *,*

1 0

*Calcular su factorizaci´on de Cholesky si existe.*

Al trabajar con sistemas Sobredeterminados (*m > n*) se utiliza a menudo otro tipo de factorizaci´on, llamado **Factorizaci´on QR**.

**Definici´on 2.16.** *Dada A* R*m,n de rango m´aximo existe una u´nica Fac- torizaci´on QR, es decir existe Q* R*m,m ortogonal y una matriz R* R*m,n trapezoidal superior tales que*

*∈ ∈*

*∈*

*A* = *QR*

Esta construcci´on se puede encontrar en el cap. 5 pg 146 del libro de Quar- teroni junto a la resoluci´on de un problema modelo sobre sistemas sobrede- terminados. El comando en Matlab para obtener la Factorizaci´on es

[*Q, R*] = **qr**(*A*)*.*

**Ejercicio 2.10.** *Calcular la factorizaci´on QR de la matriz*

 *−*1 *−*1 1 

*A* = 

1 3 3



 *−*1 *−*1 5 

1 3 7

4*,*3

*Comprobar que la norma de Frobenius de la matriz residuo E* = *A QR es del orden de* 10*−*15*.*

*−*

##### Descomposici´on Espectral

Sea dada una matriz cuadrada diagonalizable. Entonces vimos anteriormente que existe una matriz cuadrada no singular *V* tal que

*V −*1*AV* = *D* = diag(*λ*1*, ..., λn*)

Es decir, *A* es semejante a una matriz diagonal y es por tanto diagonalizable. La matriz *V* tiene por columnas los autovectores de *A* que forman una base del espacio (ya que es no singular por ser diagonalizable). Se tiene por tanto que *A* se puede factorizar mediante su espectro

*A* = *V DV −*1

lo que permite estudiar la estructura de la matriz, su car´acter y comporta- miento. Sia la matriz dada no es diagonalizable la descomposici´on espectral sigue siendo cierta en la forma

*AV* = *V D*

**Teorema 2.6.** *Sea dada una matriz cuadrada y sim´etrica. Entonces el es- pectro es real y los autovectores son ortonormales.*

**Ejemplo 2.4.** *Sea dada la matriz sim´etrica*

 10 *−*6 

*A* =

*La ecuaci´on caracter´ıstica es:*

*−*6 5

*pA*(*λ*) = (10 *− λ*)(5 *− λ*) *−* 36 = *λ*2 *−* 15*λ* + 14 = (*λ −* 1)(*λ −* 14)

*luego el espectro*

*σ*(*A*) = *{*1*,* 14*}*

*Resolviendo el sistema lineal homog´eneo*

(*A −* 14*I*)**v**1 = **0**

*se tiene el autovector normalizado*

**v**1 = (3*/√*13*, −*2*/√*13)

*Resolviendo el sistema lineal homog´eneo*

(*A − I*)**v**2 = **0**

*se tiene el autovector normalizado*

**v** = (2*/√*13*,* 3*/√*13)

2

*Autovectores que provienen de autovalores distintos son ortogonales:*

**v**1 *·* **v**2 = 0

*La matriz V que diagonaliza A es ortogonal,*

*y verifica*

*V T* = *V −*1

1 

3 2  *T*

 14 0 

*V* = *√*13  *−*2 3  *, V*

*AV* = *D* = 

0 1 

**Teorema 2.7.** *Dada una matriz cualquiera A ∈* R*m,n las matrices*

*AT A ∈* R*n,n, AAT ∈* R*m,m*

*son cuadradas y sim´etricas. Sus autovalores son reales y no negativos.*

Este resultado tiene importantes implicaciones al permitir realizar la Des- composici´on por Valores Singulares (SVD) de cualquier matriz. Adema´s se afirma que el espectro es no negativo luego *AT A* y *AAT* son SDP o DP. En cualquier caso las formas cuadr´aticas asociadas son Convexas y existe (al menos) un o´ptimo del problema de minimizaci´on.

## V Clase

Como desarrollo de los conceptos te´oricos vistos en las clases anteriores pasa- mos a describir dos t´ecnicas algebraicas que tienen importantes aplicaciones en el campo del procesamiento de im´agenes. La Descomposici´on por Valores Singulares (SVD) y la Pseudo Inversa de Moore-Penrose.

#### Descomposici´on por Valores Singulares (SVD)

Si una matriz no es cuadrada entonces no es posible calcular su espectro y la teor´ıa de los autovalores no se puede aplicar directamente sobre la matriz. Sin embargo una generalizaci´on al caso no cuadrado viene dada por la **Descom- posici´on por Valores Singulares** (SVD). Observa que cualquier matriz real admite una Descomposici´on en Valores Singulares tal y como afirma el siguiente teorema

**Teorema 3.1.** *Sea A ∈* R*m,n. Entonces existen*

*U ∈* R*m,m,* Σ *∈* R*m,n, V ∈* R*n,n*

*tales que*

*Am,n* = *Um,m*Σ*m,nV T*

*n,n*

*en donde U y V son matrices ortogonales y* Σ *es una matriz diagonal aunque no necesariamente cuadrada (ya que tiene las mismas dimensiones de A).*

**Definici´on 3.1.** *Los elementos de la diagonal de* Σ *son los* **Valores Singu- lares** *de la matriz A:*

*σ*1 *≥ ... ≥ σr ≥* 0

Los valores singulares son u´nicos pero las matrices *U* y *V* no.

**Definici´on 3.2.** *Las columnas de U son los* **vectores singulares por la izquierda** *y las columnas de V son los* **vectores singulares por la dere- cha***.*

En efecto se verifica que:

*Am,n***v***n,*1 = *σ***u***m,*1

para todo autovector **u** R*m* y **v** R*n*. Podemos interpretar esta descom- posici´on en valores singulares en t´erminos de una descomposici´on espectral.

*∈ ∈*

**Teorema 3.2.** *Sea A* R*m,n y sean U* R*m,m y V* R*n,n matrices verifi- cando la descomposici´on matricial*

*∈ ∈ ∈*

*Am,n* = *Um,m*Σ*m,nV T*

*n,n*

*Las columnas de U (vectores singulares por la izquierda) son los autovectores de la matriz*

*AAT*

*Las columnas de V (vectores singulares por la derecha) son los autovectores de la matriz*

*AT A*

Puesto que

*AT A* = (*U* Σ*V T* )*T U* Σ*V T* = *V* Σ*T UT U* Σ*V T* = *V* Σ*T* Σ*V T*

la matriz *V* diagonaliza la matriz *AT A* y por eso los vectores **v***j* son autovec- tores de *AT A*.

**Ejercicio 3.1.** *Demostrar que la matriz U diagonaliza la matriz AAT*

Los valores singulares no nulos de *A* son las ra´ız cuadrada de los autovalores de *AT A* (que coinciden con los de *AAT* ). Es decir

*σ*2 *≥ ... ≥ σ*2 *>* 0

1

*r*

son los autovalores (no nulos) de las matrices *AT A* y *AAT* .

**Ejercicio 3.2.** *Sea dada la matriz*

 1 *−*2 2 

*— −*



*A* =  

2 4 1

*−*1 *−*1 0*.* 

0 2 1 4*,*3

*Calcular los espectros de la smatrices sim´etricas*

*AT A, AAT*

**Ejemplo 3.1.** *Sea dada la matriz*

 1 1 

1 0

 *A* =

 0 1 

3*,*2

*Calcular su descomposici´on en Valores Singulares.*

Se tiene

cuyos autovalores son

2 1

*A A* =  1 2 

 

*T*

2*,*2

*λ*1 = 3*, λ*2 = 1

Los valores singulares de *A* son

*σ* = *√*3*, σ* = 1

1 2

y se tiene (observa la fila de ceros que da consistencia dimensional)

 *σ*1 0 

 *√*3 0 

Σ = 

0 *σ*2

0 0

3*,*2

= 0 1

0 0

3*,*2

Los correspondientes autovectores de *AT A* son

1  1  1  *−*1 





**v**1 = *√*2  1  *,* **v**2 = *√*2  1 

Ya podemos construir la matriz *V* de la descomposici´on

1  1 *−*1 

Usando las relaciones

*V* = *√*2  1 1 

*A***v**1 = *σ*1**u**1*, A***v**2 = *σ*2**u**2

se calculan los vectores ortogonales

 2 

 0 

 1   *−*1 

**u**1 = *√*6 

 *,* **u**2 = *√*2 



1

1

1

1

Se verifica la descomposici´on en valores singulares

*A* = *σ*1**u**1**v***T* + *σ*2**u**2**v***T*

1 2

M´as en general se tiene:

**Teorema 3.3.** *Sea A ∈* R*m,n y sean*

*σ*1 *≥ σ*2 *≥ ... ≥ σr >* 0

*sus valores singulares (los no nulos son r). Sean* **u**1*, ...,* **u***r sus vectores sin- gulares por la izquierda y* **v**1*, ...,* **v***r sus vectores singulares por la derecha. Entonces la factorizaci´on*

*se puede escribir en la forma:*

*A* = *UDV T*

*A* = *σ*1**u**1**v***T* + *...* + *σr***u***r***v***T*

1 *r*

*Adem´as se tiene:*

1. *rank*(*A*) = *r ≤ n.*
2. *La norma* 2 *de la matriz nos da el valor singular m´aximo*

*||A||*2 = *σ*1

*Si A es cuadrada (m* = *n) y no singular su condicionamiento es*

**cond**(*A*) = *||A|| ||A−*1*||*

= *σ*1

2 2

*σ*

*n*

1. *Los vectores {***u**1*, ...,* **u***r} forman una base ortonormal de R*(*A*)*.*
2. *Los vectores {***u***r*+1*, ...,* **u***n} forman una base ortonormal de N* (*AT* )*.*
3. *Los vectores {***v**1*, ...,* **v***r} forman una base ortonormal de R*(*AT* )*.*
4. *Si r < n los vectores {***v***r*+1*, ...,* **v***n} forman una base ortonormal de*

*N* (*A*)*.*

**Ejercicio 3.3.** *Sea dada la matriz del ejemplo*

 

1 1

1 0

 *A* =

 0 1 

3*,*2

*Calcular la dimensi´on y una base de los espacios*

*N* (*A*) *⊂* R2*, R*(*A*) *⊂* R3*, N* (*AT* ) *⊂* R3*, R*(*AT* ) *⊂* R2

*Clasificar las aplicaciones asociadas fA, fAT en inyectivas, suprayectivas y biyectivas. Determinar una base de estos subespacios vectoriales.*

#### La Pseudo Inversa de Moore-Penrose

Si bien no es siempre posible calcular la inversa de una matriz cuadrada y no es nunca posible calcular la inversa de una matriz no cuadrada existe la posibilidad de calcular, para cualquier matriz, su **Pseudo-Inversa de Moore-Penrose** que se define por

**Definici´on 3.3.** *Sea A* R*m,n. Entonces existe la matriz Pseudo-Inversa de Moore-Penrose, denotada por A† y definida por*

*∈*

*A†* = l´ım(*AT A* + *ϵIn*)*−*1*AT* = l´ım *AT* (*AAT* + *ϵIn*)*−*1 (4)

*ϵ→*0 *ϵ→*0

Un algoritmo muy pr´actico para calcular la Pseudo-Inversa se basa en la f´ormula

*A†* = *V* Σ*†UT* (5)

en donde *U* , Σ y *V* son la matrices proporcionadas por la SVD de *A* y Σ*†* es la pseudo-inversa de la matriz diagonal Σ que se construye tomando el rec´ıproco de los elementos no nulos de Σ y luego la traspuesta de la matriz resultante. Por ejemplo, considerando la matriz Σ de la SVD del ejemplo (**??**):

 *σ*1 0 

 *√*3 0 

se tiene



Σ = 

0 *σ*2

0 0

3*,*2

= 0 1

0 0

3*,*2

 1*/σ*1



0 *T*

1*/√*

3 0 0



Σ*†* = 

0 1*/σ*2 =

0 0

0 1 0

2*,*3

Si *A* es cuadrada y no singular (luego invertible) se recupera la matriz inversa: *A−*1 = *A†*. Para las matrices que no son de rango m´aximo el paso al l´ımite es necesario para el c´alculo de la pseudo-inversa. Para las matrices de rango m´aximo se tiene:

**Teorema 3.4.** *Sea A* R*m,n una matriz de rango m´aximo, es decir rank*(*A*) = m´ın *m, n . Si rank*(*A*) = *n podemos calcular la Pseudo-Inversa mediante la f´ormula*

*{ }*

*∈*

*A†* = (*AT A*)*−*1*AT*

*Si rank*(*A*) = *m podemos calcular la Pseudo-Inversa mediante la f´ormula*

*A†* = *AT* (*AAT* )*−*1

Si *A* R*m,n* no es una matriz de rango m´aximo, es decir rank(*A*) *<* m´ın *m, n* entonces hay que usar la f´ormula [4](#_bookmark63) de la definici´on. En todos los casos se pue- de utilizar el algoritmo [5](#_bookmark64) basado en la SVD y que est´a implementado en el comando **pinv.m** de matlab. El inter´es de la definici´on radica en que intro- duce claramente la t´ecnica de regularizaci´on de problemas mal planteados.

*∈ { }*

**Ejercicio 3.4.** *Sea dada la matriz*

*A* = 1 1

1 2

*Calcular su pseudo-inversa.*

Respuesta: La psuedo inversa es la matriz

*A†* = 2 *−*1

*−*1 1

**Ejercicio 3.5.** *Sea dada la matriz*

*A* = 1 1

1 1

*Calcular su pseudo-inversa.*

Respuesta: La psuedo inversa es la matriz

*A†* = 1*/*4 1*/*4 1*/*4 1*/*4

**Ejercicio 3.6.** *Sean dadas las matrices*

*A* = 



*Calcular su pseudo-inversa.*

2 1 

*−*1 0

0 1  *, B* = *AT*

Respuesta: Las psuedo inversas son

 1*/*3 1*/*6 

*A†* =

 

1*/*3 *−*1*/*3 *−*1*/*3

1*/*6 5*/*6 1*/*3

*B†* = (*AT* )*†* = *−*1*/*3 5*/*6 = (*A†*)*T*

*−*1*/*3 1*/*3

y se observa que la pseudo inversa de la traspuesta es la traspuesta de la pseudo inversa.

Cuando un sistema *A***x** = **b** viene descrito por una matriz rectangular *A* R*m,n* con *m* = *n* la Pseudo-Inversa de *A* permite resolver el sistema en el sentido de soluci´on de m´ınima energ´ıa (cuando hay infinitas soluciones) o en el sentido de los m´ınimos cuadrados (cuando no existe soluci´on).

*̸*

*∈*

#### Aplicaciones del

**A´lgebra Lineal**

Como resumen y aplicaci´on de los conceptos vistos en clase veremos dos apli- caciones b´asicas de la descomposici´on en valores singulares de una matriz: la Compresi´on de Im´agenes Digitales que es un problema fundamental en Visio´n Artificial y el Ana´lisis de Componentes Principales que aparece en Machine Learning para hacer m´as eficiente computacionalmente los algorit- mos de aprendizaje de caracter´ısticas no supervisados (unsupervised feature learning algorithm). La implementaci´on en Matlab se deja como ejercicio tras la lectura de las dos siguientes subsecciones.

##### Compresi´on de Im´agenes Digitales

La compresi´on de im´agenes digitales es un problema fundamental cuya re- soluci´on permite almacenar y transmitir los datos eficientemente as´ı como ahorrar memoria, ancho de banda y costes. B´asicamente hay una redundan- cia en la codificaci´on de los datos, en los valores entre p´ıxeles y tambi´en visual. Veremos un caso de **lossy compression** (compresi´on con p´erdida) en donde el enfasis est´a en la observaci´on de que en una imagen hay infor- maci´on redundante que puede no ser almacenada para permitir una tasa de transmisi´on de datos m´as elevada. Un simple ejemplo sobre Compresi´on de Im´agenes Digitales se puede encontrar en la pag 188 del libro de Quarte- roni. Tras leer una imagen mediante el comando **imread** y almacenarla en una variable, digamos *A* R*m,n* se pasan los datos enteros a datos de co- ma flotante, algo t´ıpico en Matlab, mediante el comando *A* = **double**(*A*). Se realiza la Descomposici´on *SV D* mediante el comando [U,S,V]=**svd**(A). Fijado el nu´mero de valores singulares, digamos *r*, se construye una versi´on comprimida *X* de la imagen original *A* mediante el comando:

*∈*

*Ared* = *U* (:*,* 1 : *r*) *∗ S*(1 : *r,* 1 : *r*) *∗ V* (:*,* 1 : *r*)*T*

Observa que el rango de la matriz reducida que hemos reconstruido es exac- tamente *r*, el nu´mero de valores singulares retenidos. Una cuantificaci´on de la compresi´on obtenida se puede dar definiendo el **ratio de compresi´on** *cr* que nos da el porcentaje de elementos retenidos en la versi´on reducida de la imagen:

*cr* =

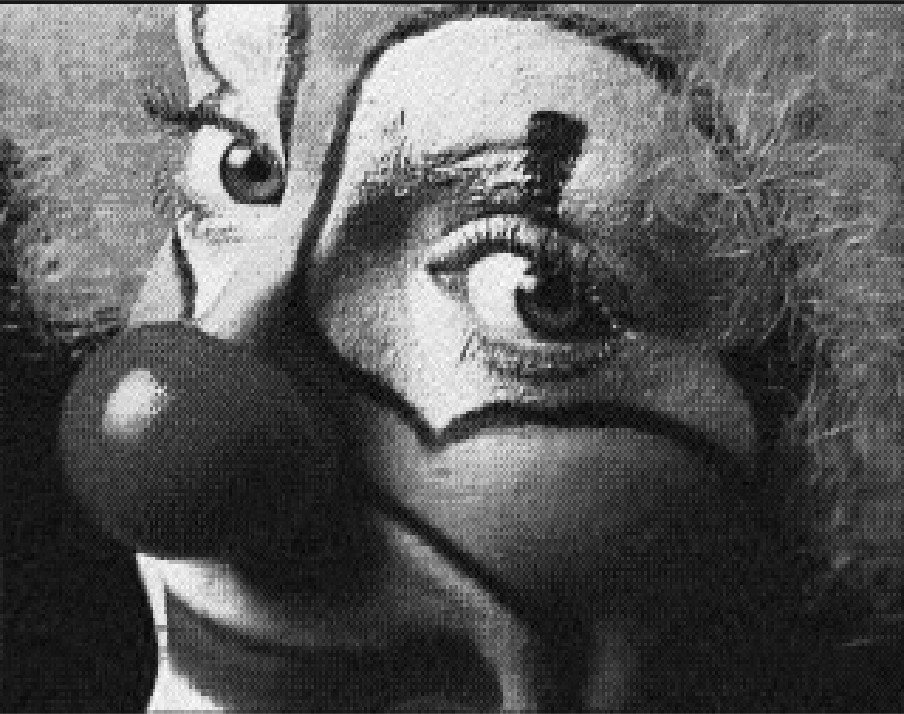
*r*(*m* + *n* + 1)

*mn*

**Ejercicio 3.7.** *Sea dada la imagen del clown que es posible cargar utilizando en matlab el comando*

load(’clown.mat’);

*Se pide calcular un descomposici´on SVD de la imagen y una versi´on reducida de la misma utilizando r* = 80 *modos de la descomposici´on. Comprobar que se retiene el* 65*.*12 % *de los valores de los p´ıxeles. Representar las im´agenes valorando visualmente la calidad de la reducci´on dimensional.*



* + - 1. Imagen original (b) Imagen reducida de rango *r* = 80.

Figura 12: Compresi´on de una imagen mediante una descomposici´on SVD.

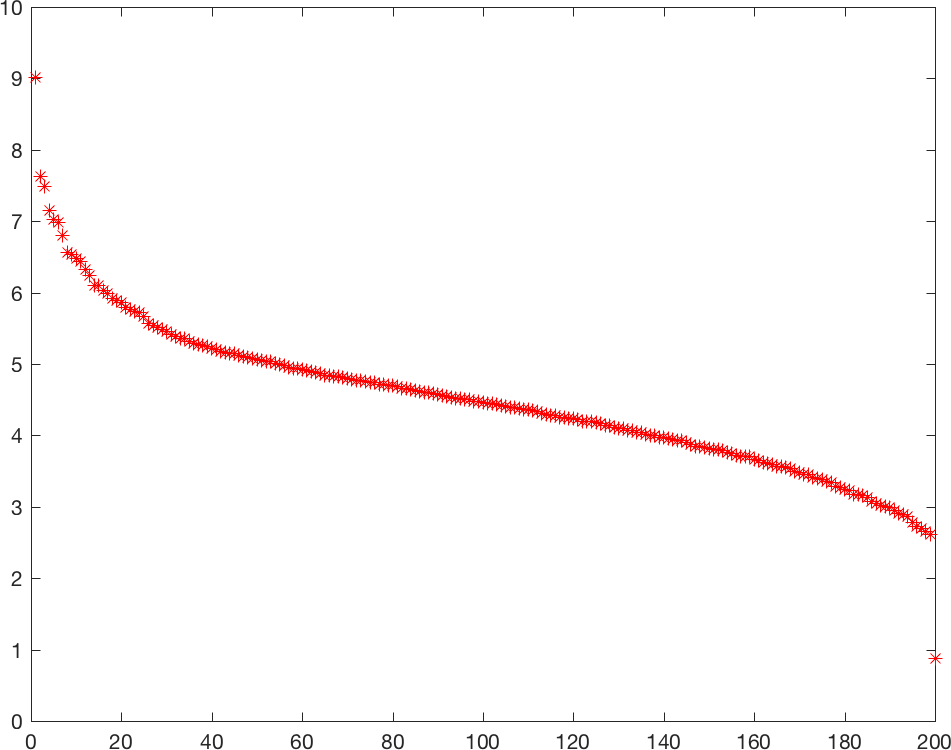


Figura 13: Gr´afica en escala logaritmica de los valores singulares de la imagen del clown.

##### An´alisis de Componentes Principales (PCA)

El Ana´lisis de Componentes Principales (PCA en ingl´es) es una de las herra- mientas m´as importantes del ´algebra aplicada y se utiliza en **aprendizaje autom´atico no supervisado** para el an´alisis de grandes cantidades de da- tos reduciendo su dimensionalidad. Permite entender tendencias, patrones y relaciones entre los datos que est´an obscurecidos en su representaci´on origi- nal.

Se trata de un algoritmo de reducci´on dimensional que puede ser utilizado para reducir de manera significante el tiempo de ejecuci´on de los algorit- mos de aprendizaje de caracter´ısticas no supervisados (unsupervised feature learning algorithm). Adema´s el entendimiento de la t´ecnica de PCA permi-

tira´ implementar un algoritmo de blanqueo (*PCA whitening*) que es un paso

de pre-procesado importante para muchos algoritmos de visi´on artificial y aprendizaje autom´atico no supervisado al reducir la redundancia entre los datos, algo inevitable cuando se procesan im´agenes. La aplicaci´on paso a pa- so de un algoritmo de PCA whitening se encuentra en el campus virtual de la Universidad de Stanford, UFLDL Tutorial,

<http://ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening/>

La base matem´atica del proceso es como sigue. Utilizaremos la notaci´on del tutorial. Sea dado un conjunto de datos **x**(1)*, ...,* **x**(*m*) en donde cada ejemplo (dato) tiene *n* componentes describiendo as´ı *n* caracter´ısticas, es decir **x**(*i*) R*n*. Queremos reducir la dimensionalidad del dataset realizando una PCA pasando a una descripci´on *k* dimensional, digamos **x**(*i*) R*k*. Para preparar los datos calculamos la media por filas

*∈*

*∈*

*{ }*

*n*

*n j*

*µ*(*i*) = 1 *x*(*i*)

*j*=1

para cada ejemplo del dataset:

**x**(*i*) = *x*(*i*)*, ...., x*(*i*)*, ...x*(*i*) *′ ∈* R*n*

1

*j*

*n*

y la restamos al dataset para normalizarlo

*x*(*i*) = *x*(*i*) *− µ*(*i*)

*j*

*j*

y obtener un nuevo dataset de media cero. Calculamos la matriz de Cova- rianza del dataset mediante la f´ormula

1 *m*

Σ = (**x**(*i*))(**x**(*i*))*T*

*m*

*i*=1

v´alida para datos de media cero. Observa la operaci´on columnas por filas que genera la matriz Σ. La matriz de Covarianza es sim´etrica y semi-definida positiva.

Mediante algoritmos de ´algebra lineal num´erica calculamos los autovalores

*{λ*1*, ..., λn}* y los autovectores *{***u**1*, ...,* **u***n}* de la matriz cuadrada Σ *∈* R*n,n*. Observa que el espectro es no negativo, *λi* 0 y que los autovalores est´an ordenados da mayor a menor:

*≥*

*λ*1 *≥ λ*2 *≥ λn ≥* 0

El primer autovalor, *λ*1 es el **autovalor principal** y corresponde a **u**1, el **autovector principal**. Con los autovectores construimos (ordenandolos por columnas) la matriz ortogonal *U* de la Factorizaci´on SVD de Σ.

Los autovectores constituyen una nueva base en la cual representar los datos. Sea ahora dado un ejemplo de entrenamiento **x**. Proyectamos en la direcci´on de los autovectores mediante el producto matriz-vector

**x***rot* = *UT* **x**

lo que equivale a una rotaci´on de los ejes cartesianos y tiene el efecto de decorrelar las caracter´ısticas. Las caracter´ısticas en esta nueva base son **x***rot,i*. La matriz de covarianza asociada a la representaci´on **x***rot* es la matriz diagonal *D* de la descomposici´on SVD con los autovalores *λi* en la diagonal. Repetimos la proyecci´on en todo el dataset obteniendo

(*i*)

**x**

*rot*

= *UT* **x**(*i*)

Al ser *U* ortogonal se tiene que la operaci´on (cambio de base) inversa es simplemente

**x** = *U* **x***rot*

En general si **x**(*i*) *∈* R*n* es un ejemplo y lo queremos pasar a dimensi´on *k < n*,

**x**(*i*) *∈* R*k* retenemos la primeras *k* componentes de **x**(*i*) que definen las direc-

*rot*

ciones de m´axima variancia y ponemos a cero las dem´as *n k* componentes obteniendo una representaci´on

*−*

**x˜**(*i*) *≈* **x**(*i*)

que se codifica eliminando los ceros en **x˜**(*i*) R*k* y en donde s´olo las primeras *k* componentes coinciden. Para estimar un valor u´til de *k* (si es muy grande no comprimo y si es muy pequen˜o tengo una mala aproximaci´on) utilizamos el concepto de *Porcentaje de variaci´on retenido* o porcentaje de varianza explicada de los datos que se define por

*∈*

*PVk* =

*k*

*λi*

*i*=1

*n*

*λi*

*i*=1

Pasamos ahora al PCA whitening imponiendo que todas las caracter´ısticas

tengas varianza unitaria. E*√*sto se puede hacer rescalando cada caracter´ısticas

**x***rot,i* mediante el factor 1*/*

*λi*. Se tiene

**x***rot,i,white* =

**x***rot,i*

*√λ*

*i*

La matriz de covarianza asociada a la representaci´on **x***rot,white* es la matriz identidad *In*: las caracter´ısticas est´an decorreladas y tienen varianza unitaria. La presencia de autovalores pr´oximos a cero puede causar explosi´on de los datos e inestabilidad num´erica. Por ello es t´ıpico usar una regularizaci´on de la forma

*ϵ* **x***rot,i*

**x***rot,i,white* = *√λ* + *ϵ*

*i*

en donde 0 *< ϵ ≪* 1.

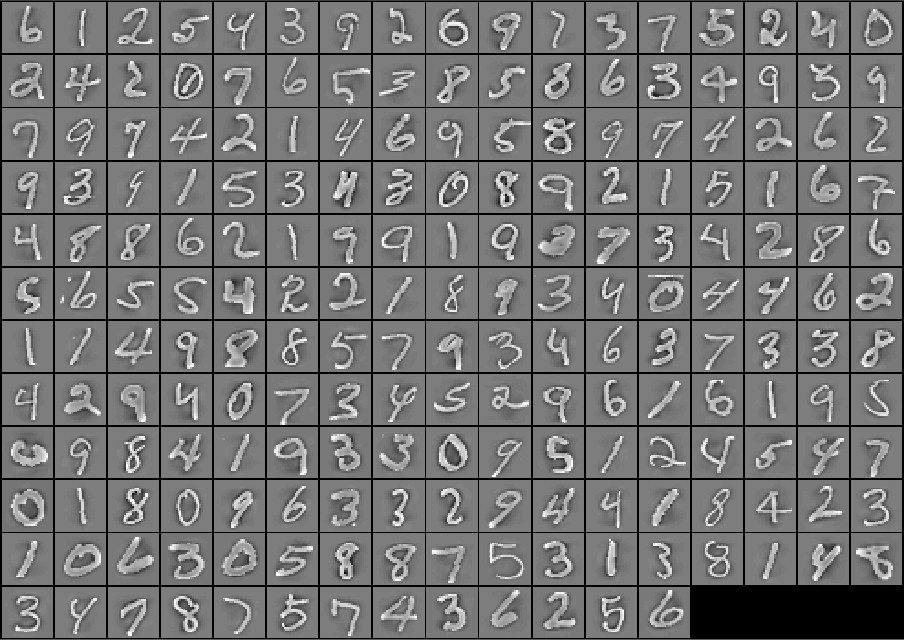


Figura 14: Imagen resultante del algoritmo de PCA con blanqueo y regula- rizaci´on *ϵ* = 0*.*5

## VI Clase

#### A´lgebra Lineal: Teor´ıa de los Sistemas Lineales

Los sistemas lineales y su resoluci´on eficiente representan el problema m´as comu´n al cual enfrentarse en el contexto del procesado de im´agenes y, en general, en la ingenier´ıa. Como libro de texto b´asico seguiremos el de Convex Optimization de S. Boyd y L. Vanderberghe cuyo pdf se encuentra en el enlace de la Universidad de Stanford del campus virtual. En particular, la resoluci´on de un sistema lineal permitir´a resolver problemas de Optimizaci´on Convexa Cuadr´atica y Lineal, lo que haremos en la siguiente secci´on en donde los conceptos b´asico de dualidad ser´an introducidos para la comprensi´on de los distintos algoritmos de optimizaci´on para la resoluci´on de problemas de Optimizaci´on Convexa. La materia desarrollada permitir´a adem´as tener una base matem´atica para el entendimiento de las t´ecnicas propias de Machine Learning y SVM (Support Vector Machine).

En este primer ejemplo, completamente desarrollado, se considera la teor´ıa de sistemas lineales y su relaci´on con la minimizaci´on de formas cuadr´aticas. A partir de un sistema incompatible se considera el sistema normal asociado al problema. Se introduce el concepto de regularizaci´on mediante problemas aproximantes bien planteados y se pasa al l´ımite obteniendo la expresi´on

de la pseudo-inversa de la matriz del sistema lo que permitir´a obtener la expresi´on expl´ıcita de la soluci´on del problema de m´ınimos cuadrados de m´ınima energ´ıa.

##### Principio de Hadamard y Dependencia Continua

El Principio de Jacques Hadamard define las 3 condiciones que un proble- ma tiene que verificar para ser definido de bien planteado o *well posed*. En concreto un problema est´a bien planteado s´ı y s´olo s´ı:

* + - 1. Existe soluci´on al problema (Existencia)
      2. La soluci´on es u´nica (Unicidad)
      3. La soluci´on depende de manera continua de los datos del problema (Dependencia Continua)

en donde la u´ltima condici´on viene a decir que si perturbamos un poco los datos la soluci´on ser´a un poco distinta de la soluci´on del problema no per- turbado. Para cuantificar (medir) exactamente el tipo de dependencia nece- sitaremos el concepto de Nu´mero de Condicionamiento de una matriz que se define a trav´es de la resoluci´on del problema de autovalores asociado a la ma- triz del sistema, v´ease la subsecci´on ([2.2.3](#_bookmark56)). Notamos que cuando queremos resolver nu´mericamente un sistema

*A***x** = **b**

en realidad estamos resolviendo el sistema perturbado

(*A* + *δA*)**x**ˆ = **b** + *δ***b**

donde *δA* es una matriz y *δ***b** un vector que estiman el error del modelo lineal y el ruido presente en los datos respectivamente. La soluci´on del problema perturbado **x**ˆ ser´a una buena aproximaci´on de la soluci´on exacta **x**, es decir

**x**ˆ *≈* **x** s´olo si el sistema tiene condicionamiento *K*(*A*) pequen˜o. Vamos a

suponer, por simplicidad, que tenemos incertidumbre s´olo en los datos **b** y no en la matriz del sistema. El caso general se puede encontrar en el libro de Quarteroni, pg 144, desigualdad 5.27 Sean entonces *δA* = 0, *δ***b** = **0** y consideremos el sistema perturbado

*̸*

*A***x**ˆ = **b** + *δ***b**

Se puede entonces demostrar que

*||***x** *−* **x**ˆ*|| ≤ λ*m´ax *||δ***b***||* = *K*(*A*) *||δ***b***||*

(6)

*||***x***||*

*λ*m´ın

*||***b***||*

*||***b***||*

Definimos el residuo del sistema

**r** = **b** *− A***x**ˆ

que se puede considerar como un estimador del error **x** *−* **x**ˆ. Observando que

*δ***b** = *A*(**x**ˆ *−* **x**) = *A***x**ˆ *−* **b** = *−***r**

podemos escribir ([6](#_bookmark72)) en la forma

*||***x** *−* **x**ˆ*|| ≤ λ*m´ax *||***r***||*

= *K*(*A*) *||***r***||*

*||***x***|| λ*m´ın *||***b***||*

*||***b***||*

Esta desigualdad implica que, aun teniendo residuo **r** pequen˜o podemos estar muy lejos de la soluci´on, es decir **x x**ˆ puede ser grande si el condiciona- miento *K*(*A*) es grande.

*|| − ||*

##### Teorema de Rouch´e-Frobenius

El Teorema de Rouch´e-Frobenius afirma:

**Teorema 4.1.** *Sea dada una matriz A ∈* R*m,n que define el sistema*

*A***x** = **b**

*en donde* **x** R**n** *es el vector de inc´ognitas y* **b** R**m** *es el vector de datos. Sea Aamp* = (*A* **b**) *la matriz ampliada del sistema, Aamp* R*m,n*+1*.*

*| ∈*

*∈ ∈*

*Entonces el sistema es Compatible (admite soluci´on) s´ı y s´olo s´ı los rangos de la matriz de coeficientes A y de la matriz ampliada Aamp son iguales:*

*rank*(*A*) = *rank*(*Aamp*)

*Si existen soluciones ´estas forman un sub-espacio af´ın de* R*n de dimensi´on*

*n − rank*(*A*)*. En particular si*

*rank*(*A*) = *n*

*el sistema es Compatible Determinado y la soluci´on es u´nica. Si*

*rank*(*A*) *< n*

*el sistema es Compatible Indeterminado y las soluciones son infinitas.*

##### F´ormula de la Dimensi´on

La Fo´rmula de la Dimensi´on relaciona el nu´mero de columnas de una matriz con las dimensiones de sus espacios imagen y nu´cleo. En concreto si *A* R*m,n* se tiene

*∈*

*n* = *dim*(*R*(*A*)) + *dim*(*N* (*A*)) (7)

Mediante el uso de la f´ormula es posible analizar la estructura del sistema.

##### Sistemas Sobredeterminados e Indeterminados

Sea dada una matriz *A ∈* R*m,n* que define el sistema

*A***x** = **b**

en donde **x** R**n** es el vector de inc´ognitas y **b** R**m** es el vector de datos. Supongamos que *m > n* luego el sistema es sobredeterminado (tenemos m´as restricciones que grados de libertad). En este caso el sistema puede no tener soluci´on. La soluci´on existe s´olo si **b** (*A*) R*m* (pertenece al sub-espacio imagen). En general, para **b** arbitrario podemos buscar un **x***∗* R*n* que minimice (el cuadrado de) la norma eucl´ıdea del residuo. Es decir

*∈*

*∈ ∈*

*∈ R ⊂*

Φ(**x***∗*) = *||A***x***∗ −* **b***||*2 = m´ın *||A***x** *−* **b***||*2 = m´ın Φ(**x**)

2

2

**x***∈*R*n* **x***∈*R*n*

Supongamos ahora que *m < n* luego el sistema es indeterminado (tenemos menos restricciones que grados de libertad). En este caso el sistema, de tener soluci´on, no puede ser u´nica. Se deduce a partir de la f´ormula de la dimensi´on ([7](#_bookmark75)). Las infinitas soluciones, cuando existen, deben de ser parametrizadas y el nu´mero de par´ametros necesarios viene dado por *dim*(*N* (*A*)).

#### El Problema de M´ınimos cuadrados

La soluci´on **x***∗* R*n* del sistema sobredeterminado se llama **Soluci´on de M´ınimo Cuadrado**. Se puede encontrar imponiendo que el gradiente de la funci´on (campo escalar)

*∈*

sea nulo, es decir

2

Φ(**x**) = *||A***x** *−* **b***||*2

*∇*Φ(**x***∗*) = **0**

##### Ecuaciones Normales

Tr´as su c´alculo (veremos m´as adelante su deducci´on) se tiene que **x***∗* R*n*

*∈*

tiene que verificar el sistema lineal cuadrado

*AT A***x** = *AT* **b**

que se obtienen de manera pr´actica al multiplicar el sistema original por la matriz *AT* . Si *B* = *AT A*, sim´etrica, tiene rango m´aximo, es decir

*rank*(*B*) = m´ın(*m, n*)

la matriz *B* es definida positiva. Este sistema es **no singular** (es decir la matriz *AT A* es invertible) y existe una u´nica soluci´on del problema de mini- mizaci´on. Para calcular la soluci´on se podr´ıa usar la descomposici´on de Cho- lesky pero debido a los errores de redondeo el c´alculo de la matriz *B* = *AT A* podr´ıa verse afectado por errores de redondeo con la consecuente p´erdida del car´acter definido positivo de la matriz. T´ıpicamente se opta por una Facto- rizaci´on QR. Recordamos para ello que dada *A ∈* R*m,n* de rango m´aximo existe una u´nica **Factorizaci´on QR**, es decir existen *Q ∈* R*m,m* ortogonal y una matriz *R ∈* R*m,n* trapezoidal superior tales que

*Am,n* = *Qm,mRm,n*

Usando esta descomposici´on el sistema *A***x** = **b** se transforma en

*QR***x** = **b** *∈* R*m*

Multiplicamos por *QT* para obtener las ecuaciones normales:

*m,m*

*QT QR***x** = *R***x** = *QT* **b** *∈ R*(*A*) *⊂* R*m*

siendo *QT* **b** la proyecci´on ortogonal de **b** en el espacio imagen. Para problema sobredeterminados *m > n* y la matriz *Rm,n* no es cuadrada luego no es inver- tible. Eliminando informaci´on redundante definimos la matriz *R*¯ R*n,n* que es la restricci´on cuadrada de *R* definida por *R*¯ = *R*(1 : *n,* 1 : *n*) (eliminando sus u´ltimas *m n* filas) y la matriz *Q*¯ R*m,n* definida por *Q*¯ = *Q*(1 : *m,* 1 : *n*) (eliminando sus u´ltimas *m n* columnas). La u´nica soluci´on de m´ınimos cua- drados viene dada por

*∈*

*−*

*— ∈*

**x***∗* = (*R*¯*−*1)*n,nQ*¯*T*

*n,m*

**b***m,*1 *∈* R*n*

##### Aplicaci´on de la Pseudo Inversa de Moore-Penrose

**Ejemplo 4.1.** *Sea dada la matriz A y el vector* **b** = [2*,* 0]*T que definen el sistema A***x** = **b**

 1 1   *x*1   2 

 1 1   *x*2  =  0 

*siendo* **x***′* = (*x*1*, x*2)*′ el vector de inc´ognitas. Discutir la existencia y unicidad de soluciones del sistema A***x** = *b mediante el Teorema de Rouche-Frobenius. Se pide:*

* + - 1. *Calcular una soluci´on de m´ınimos cuadrados.*
      2. *¿Es tambi´en soluci´on del sistema original A***x** = **b***?*
      3. *¿Es la u´nica soluci´on de m´ınimos cuadrados ?*
      4. *Calcular, si existe, la soluci´on de m´ınimos cuadrados de m´ınima energ´ıa.*
      5. *Calcular la energ´ıa m´ınima asociada.*

Respuesta: Aplicamos el Teorema de Rouche-Frobenius puesto que el pro- blema es lineal. Al ser la matriz cuadrada podemos calcular su determinante (*det*(*A*) = 0) y deducir

*rg*(*A*) = 1 *<* 2 = *rg*(*A|***b**)

de donde la incompatibilidad (no existencia de soluciones). Esto se resume diciendo que **b** *∈/ Imf* siendo *fA* = *f* : R2 *→* R2 la aplicaci´on lineal asociada

a la matriz segu´n las bases can´onicas.

Sea *r* = (*r*1*, r*2)*T* , **b** = (*b*1*, b*2)*T* . Al no haber soluci´on intentamos reducir el residuo del sistema, *r* = *A***x b** minimizando su norma (al cuadrado para dar diferenciabilidad al problema), es decir calculando la soluci´on del problema de minimizaci´on cuadr´atico

*−*

arg m´ın 1 *||A***x** *−* **b***||*2

**x***∈*R*n* 2 2

El factor 1*/*2 se introduce por comodidad ya que al derivar se simplificar´a con el cuadrado de la norma. Por supuesto el valor de la energ´ıa se reduce a la mitad pero estamos interesados en el punto **x** en donde se alcanza el m´ınimo y no en el valor m´ınimo alcanzado por la energ´ıa.

Al utilizar la notaci´on de argumento m´ınimo arg m´ın se tiene arg m´ın 1 *||A***x** *−* **b***||*2 = arg m´ın *||A***x** *−* **b***||*2

2

2

*x∈*R*n* 2 **x***∈*R*n*

Vamos a analizar el problema desarrollando la expresi´on de la norma cuadr´ati- ca en la forma

*||A***x** *−* **b***||*2 = *⟨A***x** *−* **b***, A***x** *−* **b***⟩* = (*A***x** *−* **b**) (*A***x** *−* **b**) =

2

*T*

= (**x***T AT −* **b***T* )(*A***x** *−* **b**) = **x***T AT A***x** *−* **x***T AT* **b** *−* **b***T A***x** + **b***T* **b** =

= **x***T AT A***x** *−* 2**x***T AT* **b** + **b***T* **b**

porque

**b***T A***x** = *⟨***b***, A***x***⟩* = *⟨A***x***,* **b***⟩* = (*A***x**)*T* **b** = **x***T AT* **b***.*

Tenemos por tanto el problema de minimizaci´on

arg m´ın 1 *||A***x** *−* **b***||*2 = arg m´ın (1 **x***T AT A***x** *−* **x***T AT* **b** + 1 **b***T* **b**

2

**x***∈*R*n* 2 **x***∈*R*n* 2 2

que podemos escribir en la forma estandar (v´ease el ejemplo 4.5 pg 140 del Boyd)

arg m´ın (1 **x***T P* **x** + **q***T* **x** + *r*

**x***∈*R*n*

2

siendo

 2 

 2 

*||***b***||*2

*P* = *AT A* =   *,* **q** = *−AT* **b** = *−*   *, r* = = 2

|  |  |  |  |
| --- | --- | --- | --- |
| 2 |  | | |
|  |  |  | 2 |
| 2 | 2 | 2 | 2 |

Definimos y calculamos expl´ıcitamente la forma cuadr´atica asociada

1 *T T*

1  2 2   *x*1   *x*1 

*f*0(**x**) = 2 **x** *P* **x**+*q* **x**+*r* = 2 (*x*1*, x*2)  2 2   *x*2 *−*(2*,* 2)  *x*2



+2 =

1

= 2 (*x*1*, x*2)

2*x*1 + 2*x*2

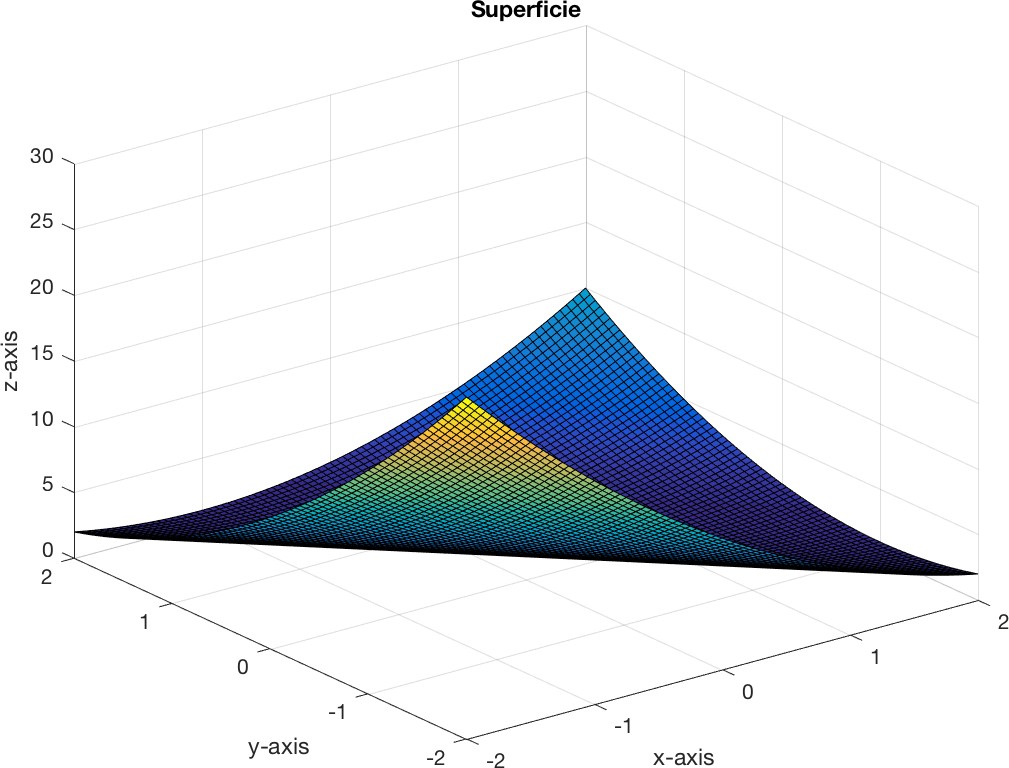
2*x*1 + 2*x*2

 *−* 2*x*1 *−* 2*x*2 + 2 =

= *x*2 + *x*2 + 2*x*1*x*2 *−* 2*x*1 *−* 2*x*2 + 2

1

2



Utilizando MATLAB vemos por la gr´afica (superficie) de la forma cuadr´atica, la existencia de infinitos puntos de m´ınimos (globales ya que es un problema convexo). Calculamos los autovalores de *P* considerando el sistema *P* **x** = *λ***x**. Se tiene la ecuaci´on caracter´ıstica

2 2

*p*(*λ*) = (2 *− λ*) *−* 4 = *λ −* 4*λ* = *λ*(*λ −* 4)

luego *P ∈ S*2 porqu´e es sim´etrica y semidefinida positiva al tener autovalores

+

no negativos: *λ*1 = 0, *λ*2 = 4. Al aparecer el autovalor nulo *λ* = 0 la matriz sigue siendo singular (*A* y *AT* son singulares). Derivando la forma cuadr´atica convexa se tiene la condici´on necesaria y suficiente (por la convexidad)

*P* **x** + **q** = 0

que es equivalente al sistema de ecuaciones normal

*AT A***x** = *AT* **b**

##### Derivaci´on de la forma cuadr´atica

La derivaci´on de

*f*0(**x**) = *x*2 + *x*2 + 2*x*1*x*2 *−* 2*x*1 *−* 2*x*2 + 2

1

2

es como sigue. Puesto que *f*0(**x**) es una funci´on multivariable calcularemos su gradiente.

Utilizando la definici´on de derivadas direccional perturbamos la funci´on a trav´es de un par´ametro 0 *< t ≪* 1 y en todas las posibles direcciones dadas por un gen´erico vector **v**. Pasando al l´ımite para *t →* 0 se tiene

*⟨∇f* (**x**)*,* **v***⟩* = l´ım *f*0(**x** + *t***v**) *− f*0(**x**) =

0

*t→*0 *t*

= l´ım

*t→*0

(1*/*2)(**x** + *t***v**)*T P* (**x** + *t***v**) + **q***T* (**x** + *t***v**) + *r −* (1*/*2)**x***T P* **x** *−* **q***T* **x** *− r t*

(1*/*2)(**x***T* + *t***v***T* )(*P* **x** + *tP* **v**) + *t***q***T* **v** *−* (1*/*2)**x***T P* **x**

=

= l´ım

*t→*0

= l´ım

*t→*0

=

*t*

(1*/*2)**x***T* (*tP* **v**) + (1*/*2)(*t***v***T* )(*P* **x** + *tP* **v**) + *t***q***T* **v**

=

*t*

= (1*/*2)**x***T P* **v** + (1*/*2)**v***T P* **x** + **q***T* **v** =

= (1*/*2) *PT* **x***,* **v**) + (1*/*2) *⟨***v***, P* **x***⟩* + *⟨***q***,* **v***⟩* =

= (1*/*2) *PT* **x***,* **v**) + (1*/*2) *⟨P* **x***,* **v***⟩* + *⟨***q***,* **v***⟩* =

= (1*/*2) (*PT* + *P* )**x***,* **v** + *⟨***q***,* **v***⟩* = (puesto que *P* es sim´etrica)

)

= *⟨P* **x***,* **v***⟩* + *⟨***q***,* **v***⟩* = *⟨P* **x** + **q***,* **v***⟩*

es decir

*⟨∇f*0(**x**)*,* **v***⟩* = *⟨P* **x** + **q***,* **v***⟩ , ∀* **v** *∈ V*

de donde se deduce la **condici´on necesaria de optimalidad**

ya que

*P* **x** + **q** = 0

*⟨P* **x** + **q***,* **v***⟩* = 0*, ∀* **v** *∈ V.*

Observa que, en general *P* = *A′A* es S.D.P luego la forma cuadr´atica *f*0(**x**) es convexa y la condici´on es suficiente para la existencia de almenos un punto de m´ınimo.

**Ejercicio 4.1.** *Sea dada la matriz A* R*n,n, el vector de datos* **b** R*n y la forma cuadr´atica*

*∈ ∈*

*f* 1 *′ ′*

(**x**) = 2 **x** *A***x** *−* **b x**

*Suponiendo que A* **no** *es sim´etrica calcular el gradiente de la forma cuadr´ati- ca.*

##### An´alisis del sistema

Analicemos el sistema normal

*AT A***x** = *P* **x** = *−***q** = *AT* **b**

es decir

 2 2   *x*1   2 

siendo

 2 2   *x*2  =  2 

*T*  1 1   2   2 

**q** = *−A*

**b** = *−*  1 1   0  = *−*  2 

El Teorema de Rouche-Frobenius nos dice

*rg*(*A*) = 1 = *rg*(*A|***b**) *< n* = 2

luego el sistema es compatible indeterminado. Simplificando, se deduce de la primera ecuaci´on del sistema normal

*x*1 + *x*2 = 1

que el conjunto de soluciones es del tipo

(*x*1*, x*2) = (1 *− x*2*, x*2) = (1*,* 0) *− x*2(1*, −*1)

lo que representa un espacio lineal (af´ın) de dimensi´on 1. En concreto

*kerfP* = *L*[(1*, −*1)]*.*

Existen por tanto infinitas soluciones del problema de m´ınimos cuadrados.

Para determinar una u´nica soluci´on vamos a buscar aquella de norma cuadr´ati- ca (energ´ıa) m´ınima. Sea *Id* la matriz identidad. Esto se corresponde a una regularizaci´on del tipo

*Bϵ* = *AT*

*A* + *ϵId* = *P* + *ϵId* =

2 + *ϵ* 2

2 2 + *ϵ* 





que es ahora invertible para todo *ϵ >* 0 porqu´e

*det*(*Bϵ*) = (2 + *ϵ*)2 *−* 4 = *ϵ*2 + 4*ϵ >* 0*, ∀ ϵ >* 0

Calculamos los autovalores de *Bϵ* resolviendo *Bϵ***x** = *λ***x**. Se tiene la ecuaci´on caracter´ıstica

*Pϵ*(*λ*) = (2 + *ϵ − λ*)2 *−* 4 = *λ*2 *−* 2(2 + *ϵ*)*λ* + *ϵ*2 + 4*ϵ*

de donde los autovalores positivos *λ*1 = *ϵ*, *λ*2 = 4 + *ϵ*. El sistema asociado a la matriz *Bϵ* es

*Bϵ***x** = (*AT A* + *ϵId*)**x** = *AT* **b**

es decir

 2 + *ϵ* 2

  *x*1

 2 

Se tiene

2 2 + *ϵ*   *x*2

 =  2 

 2 + *ϵ* 2

*Bϵ−*1 = (*AT A* + *ϵId*)*−*1 = (*P* + *ϵId*)*−*1 = 

2 2 + *ϵ*

*−*1



=

=

(

Su (u´nica) soluci´on es

1

(2 + *ϵ*)2 *−* 4

2 + *ϵ −*2

*−*2 2 + *ϵ*





*−*1 *T* ( 1

**x***ϵ* = *Bϵ A*

**b** =

 2 + *ϵ −*2

  2 

(  2*ϵ*   2*ϵ* 

(2 + *ϵ*)2 *−* 4

*−*2 2 + *ϵ*   2  =

1

*−*

(2 + *ϵ*)2 *−* 4 

=

(2 + *ϵ*)2 *−* 4

 2*ϵ*

 = 

(2 + *ϵ*)2 4

2*ϵ* 

Podemos finalmente calcular la matriz regularizante

*Rϵ* = *AT Bϵ−*1 = *Bϵ−*1*AT* = *AT* (*AT A* + *ϵId*)*−*1 = (*AT A* + *ϵId*)*−*1*AT* =

( 1  2 + *ϵ −*2

  1 1 

=

(2 + *ϵ*)2 *−* 4

*−*2 2 + *ϵ*   1 1  =

1

(

=

(2 + *ϵ*)2 *−* 4

*ϵ ϵ*

 *ϵ ϵ*  =

 

(

*ϵ*

(2 + *ϵ*)2 *−* 4

1 1

 1 1  =

 



*ϵ*



(2 + *ϵ*)2 4



*−*

= *ϵ*



(2 + *ϵ*)2 *−* 4

*ϵ*

(2 + *ϵ*)2 4

*−*





*ϵ*

(2 + *ϵ*)2 *−* 4

Pasando al l´ımite se tiene (utilizando la regla del Hopital debido a la forma indeterminada 0*/*0 que aparece en el l´ımite) el valor de la matriz pseudo- inversa de Moore-Penrose

 1 1

  1*/*4 1*/*4 

*A†* = l´ım *Rϵ*



*ϵ→*0

= l´ım

*ϵ→*0

2(2 + *ϵ*)

1

2(2 + *ϵ*)

2(2 + *ϵ*)

1

2(2 + *ϵ*)

 = 

1*/*4 1*/*4 

y la u´nica soluci´on de norma m´ınima viene dada por

*\* †*  1*/*4 1*/*4   2   1*/*2 

**x**

= *A* **b** =

1*/*4 1*/*4   0  =  1*/*2 

Pasando al l´ımite en la sucesi´on *{***x***ϵ}* se tiene (nuevamente usamos Hopital)

 

Observar que

l´ım **x***ϵ* = **x***∗*

*ϵ→*0

1*/*2

=  1*/*2 

l´ım **x***ϵ* = l´ım *Bϵ−*1*AT* **b** = l´ım *Rϵ***b** = l´ım(*AT A* + *ϵId*)*−*1*AT* **b** = *A†***b** = **x***∗*

*ϵ→*0

*ϵ→*0

*ϵ→*0

*ϵ→*0

Volviendo al conjunto de soluciones del problema de m´ınimos cuadrados, parametrizadas en la forma

(*x*1*, x*2) = (1 *− x*2*, x*2) = (1 *− α, α*)

calculamos su energ´ıa y minimizamos. Se tiene

*E*(*α*) = *||*(1 *− α, α*)*||*2 = (1 *− α*)2 + *α*2 = 1 *−* 2*α* + 2*α*2

2

luego

*E′*(*α*) = *−*2 + 4*α* = 0

de donde *α* = 1*/*2, *E*(1*/*2) = 1*/*2 y reencontramos la soluci´on o´ptima

**x***∗* = (1*/*2*,* 1*/*2)

Claramente no es soluci´on de *A***x** = **b** (que es incompatible), es soluci´on de *AT A***x** = *AT* **b** pero no es la u´nica (ya que es indeterminado) y es el l´ımite para *ϵ* 0 de la sucesi´on de soluciones (u´nicas) de los problemas compatibles determinados

*→*

(*AT A* + *ϵId*)**x** = *AT* **b**

de donde la f´ormula expl´ıcita

**x** = l´ım(*AT A* + *ϵId*)*−*1*AT* **b** = *A†***b**

*ϵ→*0

**Ejercicio 4.2.** *Sea dada la matriz A*

 2 *−*1 

0 1

*A* =

1 0





 *−*1 2 

*y el vector* **b***′* = [1*,* 0*,* 2*,* 4]*′. Se pide:*

1. *Clasificar el sistema asociado A***x** = **b** *en* indeterminado *o* sobretermi- nado*.*
2. *Discutir la existencia y unicidad de soluciones mediante el Teorema de Rouche-Frobenius.*
3. *Calcular una soluci´on de m´ınimos cuadrados. ¿Es tambi´en soluci´on del sistema ? Es u´nica ?*

Respuesta: 1) El sistema *A***x** = *b* es sobredeterminado ya que *m* = 4 *>* 2 = *n*.

1. Es sistema es incompatible. No hay existencia de soluciones ya que

*rank*(*A*) = 2 *̸*= 3 = *rank*(*A|b*)*.*

1. El problema de m´ınimos cuadrados est´a bien planteado ya que el rango de *A′A* es m´aximo

*rank*(*A′A*) = 2

luego *A′A* es sim´etrica y definida positiva. [1](#_bookmark81) La soluci´on de m´ınimos cuadra- dos es **x** = (1*.*4*,* 2*.*1)*′* y es la u´nica. No es soluci´on del sistema original que es incompatible.

1Observa que *A′A* siempre es sim´etrica. Si adem´as es de rango m´aximo entonces es definida positiva. En general es semi-definida positiva.

**Ejercicio 4.3.** *Sea dada la matriz Q*

 

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | 2 | 3 | 13 |
| 5 | 11 | 10 | 8 |
| 9 | 7 | 6 | 12 |
| 4 | 14 | 15 | 1 |

*Q* =

 

*predefinida en matlab mediante el comando Q* = *magic*(4) *y el vector* **q** = [34*,* 34*,* 34*,* 34]*′. Se pide:*

* 1. *Discutir el sistema Q***x** = **q***.*
  2. *Calcular la soluci´on de m´ınimos cuadrados de norma m´ınima.*

Respuesta: 1) Para el sistema *Q***x** = **q** se tiene que hay existencia de solucio- nes ya que *rank*(*Q*) = 3 = *rank*(*Q|***q**).

1. El problema de m´ınimos cuadrados no est´a bien planteado ya que *rank*(*A*) = 3 *<* 4 (no es de rango m´aximo) luego *Q′Q* es sim´etrica y semi-definida posi- tiva. Una soluci´on de m´ınimos cuadrados se puede obtener mediante *Q* **q** y otra mediante el calculo de la pseudo-inversa de Moore-Penrose. Ambas son soluciones del sistema *Q***x** = **q** que no tiene unicidad. La soluci´on de m´ınimos cuadrados de norma m´ınima es **x** = (1*,* 1*,* 1*,* 1)*′*.

**Ejercicio 4.4.** *Se considera la teor´ıa de sistemas lineales. Para ello sea dada la matriz A*3*,*4

 2 *−*1 3 0 

*A* =  0 1 0 1 

1 0 *−*1 2

*y el vector b* = [1*,* 0*, −*1]*′. Se pide:*

* 1. *Discutir la existencia y unicidad de soluciones del sistema mediante el Teorema de Rouche-Frobenius. ¿Se puede decir que el problema aso- ciado a la resoluci´on del sistema est´a bien planteado ? Calcular una soluci´on del sistema*

*A***x** = *b*

*mediante el operador backslash y mediante la Pseudo-Inversa de Moore- Penrose. Discutir el resultado. [1 punto]*

* 1. *Calcular una soluci´on del problema de m´ınimos cuadrados asociado:*

m´ın *∥A***x** *− b∥*2

**x**

2

*lo que se corresponde a resolver el sistema*

*AT A***x** = *AT b*

*¿Es tambi´en soluci´on del sistema original A***x** = *b? ¿Se puede decir que el problema est´a bien planteado ? Calcular la soluci´on del problema de m´ınimos cuadrados que tenga m´ınima energ´ıa:*

*¿Existe soluci´on ? Es tambi´en soluci´on del sistema original A***x** = *b ?*

Respuesta Para el sistema

*A***x** = *b*

se tiene que es compatible indeterminado y existen infinitas soluciones ya que *rank*(*A*) = 3 = *rank*(*A b*) = *m < n* = 4 siendo *m* = 3 el nu´mero de ecuaciones y *n* = 4 el nu´mero de inc´ognitas. El problema no est´a bien planteado. La soluci´on calculada con el operador backslash coincide en Octave con la calculada con la Pseudo-Inversa de Moore-Penrose:

*|*

**x** = (*−*0*.*1019*,* 0*.*2130*,* 0*.*4722*, −*0*.*2130)*T*

y tiene residuo del orden de 1*e* 16. Si utilizamos matlab, versi´on 2016b, la soluci´on calculada con la Pseudo-Inversa de Moore-Penrose es la misma anterior pero la soluci´on calculada con el operador backslash es

*−*

**x** = (0*,* 0*.*2857*,* 0*.*4286*, −*0*.*2857)*T*

Esto se debe a que matlab ha especializado el operador backslash para en- contrar soluciones *sparse*, es decir huecas, con muchos ceros y se aprecia que esta u´ltima soluci´on tiene la primera componente nula. De manera anal´ıtica

calculamos la dimensi´on del nu´cleo, dimker(*f* ) = 1 y deducimos que todas las soluciones se pueden escribir en la forma

( 2 7 3 3

*−* 5 *−* 5 *t, −t,* 5 + 5 *t, t*

( 2 3

( 7 3

Tampoco el problema de m´ınimos cuadrados, asociado al sistema

=

*−* 5 *,* 0*,* 5 *,* 0

+ *t*

*−* 5 *, −*1*,* 5 *,* 1

*A′T A***x** = *AT b*

est´a bien planteado. Sea *B*4*,*4 = *AT A*. Se determina facilmente *B* es singular ya que *rank*(*B*) = 3 *<* 4 luego *B* es sim´etrica y semi-definida positiva. Adema´s *rank*(*B*) = *rank*(*B AT b*) = 3 *<* 4 luego el sistema asociado al problema de m´ınimos cuadrados admite infinitas soluciones. Si utilizamos matlab, versi´on 2016b, la soluci´on calculada con el operador backslash es

*|*

**x** = (1*,* 1*,* 0*, −*1)*T*

y vemos que tiene una coordenada nula (es lo m´as *sparse* posible). Utilizando Octave la soluci´on calculada con el operador slash coincide nuevamente con la soluci´on calculada con la Pseudo-Inversa de Moore-Penrose y es

**x** = (*−*0*.*1019*,* 0*.*2130*,* 0*.*4722*, −*0*.*2130)*T*

y tiene residuo del orden de 1*.e −* 16. Es soluci´on del sistema. Tiene norma m´ıınima *||***x***||*2 = 0*.*5693 y coincide con la soluci´on calculada anteriormente.

## VII Clase

#### Aplicaciones

En esta clase consideraremos algunas aplicaciones de la teor´ıa de m´ınimos cuadrados.

##### Regresi´on Lineal

Una primera aplicaci´on de la teor´ıa de m´ınimos cuadrados consiste en cal- cular una recta de regresi´on, es decir una recta que pase por una nube de puntos minimizando la distancia eucl´ıdea entre todos los puntos y la recta. Un ejemplo interesante se encuentra en el libro de Quarteroni, problema 3.3 pg 74, en donde las mediciones de un test de biomec´anica para caracteri- zar el comportamineto de un tejido biol´ogico frente a distintas tensiones es analizado.

**Ejercicio 5.1.** *Sea dada una matriz A en donde registramos las tensiones σ aplicadas en un ensayo y un vector b en donde ponemos las deformaciones ϵ medidas. En concreto*

 *σ*1 1 

 0 1 

 *ϵ*1   0 

*σ*2 1

 

*σ* 1

 3

*σ*4 1

 

0*.*06 1

0*.*14 1

 

 0*.*25 1 

 



*ϵ*2 0*.*08

*ϵ*3 0*.*14



*ϵ*4 0*.*20



 

=



*A* = = *, b* =

   

*σ*5 1 0*.*31 1  *ϵ*

  0*.*23 

   



 

 7

 

 5   

*σ*6 1



*σ* 1

 *σ*8 1 

0*.*47 1

0*.*60 1

0*.*70 1

*ϵ*6

*ϵ*8 

 *ϵ*7

 0*.*28 

 0*.*25 

0*.*29

*Se pide encontrar la recta de regresi´on lineal*

*ϵ* = *x*1*σ* + *x*2

*y estimar as´ı las deformaciones para una tensi´on de* 0*.*5 *MPa siendo MPa*=

*N/cm*2*.*

Respuesta: Las inc´ognitas de la recta de regresi´on lineal son las soluciones del sistema *A***x** = **b**. Se trata de un sistema sobredeterminado ya que *m > n* siendo *m* = 8 ecuaciones y *n* = 2 inc´ognitas. No hay existencia ya que

*rg*(*A*) = 2 *<* 3 = *rg*([*Ab*])

y tiene sentido buscar una soluci´on de m´ınimos cuadrados. Multiplicando por la matriz traspuesta se tiene:

*A′A* =

1*.*2527 2*.*53

2*.*53 8

 

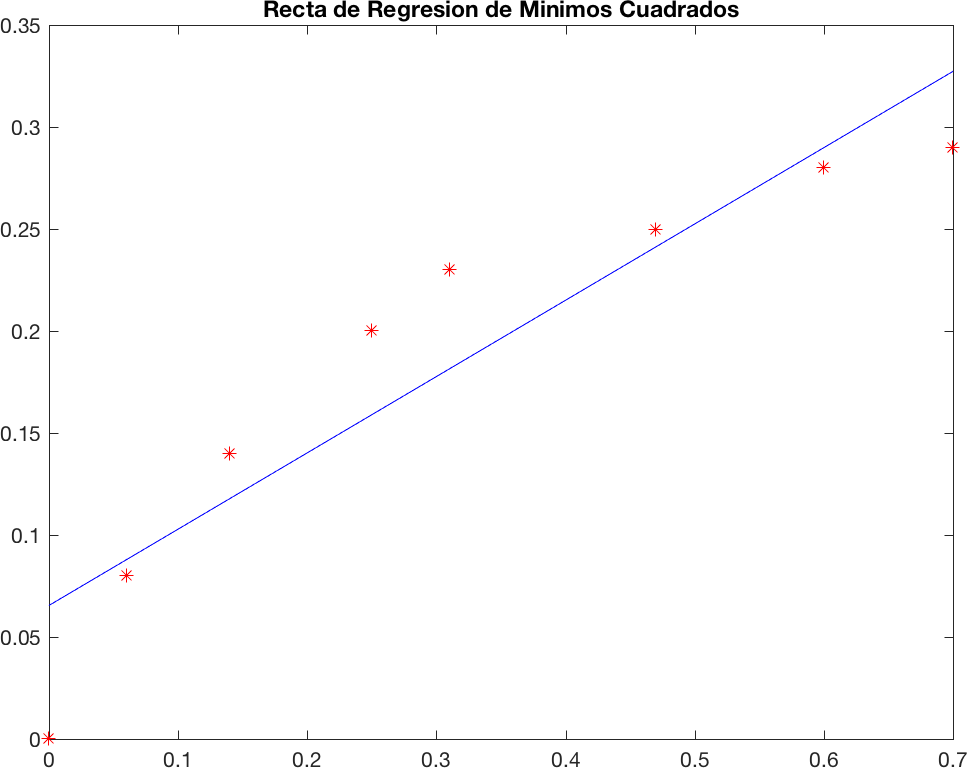
que es cuadrada, sim´etrica y no singular ya que *det*(*A′A*) = 3*.*6207 luego es Definida Positiva (DP) y el problema de m´ınimos cuadrados est´a bien planteado y tiene una u´nica soluci´on que podemos calcular con el operador pinv[2](#_bookmark85) obteniendo

**x***∗* = (*x*1*, x*2)*T* = (0*.*3741*,* 0*.*0654)*T*

Utilizando estos valores la recta de regresi´on es

*ϵ* = *x*1*σ* + *x*2 = 0*.*3741*σ* + 0*.*0654

y se estima el valor de deformaci´on *ϵ* = 0*.*2525 para una tensi´on *σ* = 0*.*5.



2No es necesario utilizar el operador pinv. Otros m´etodos directos podr´ıan serm´as r´apidos. En los ficheros de matlab se comparan varios m´etodos directos en donde destaca la factorizaci´on QR ya que el problema de m´ınimos cuadrados est´a bien planteado.

##### Aplicaci´on: Regresio´n Lineal in machine learning

Utilizando los conceptos y definiciones introducidos anteriormente presenta- mos en esta secci´on un simple primer algoritmo de aprendizaje autom´atico (machine learning): el de Regresi´on Lineal. Este tratamiento se puede encon- trar en el cap 5, Machine learning Basics del libro online de deeplearning.

El problema consiste en construir un sistema (un modelo) que dado un vector de input **x** R*n* sea capaz de predecir un valor escalar *y* R como output. Definimos como *y*ˆ R el valor de output de nuestro sistema, es decir nuestra predicci´on:

*∈*

*∈ ∈*

*y*ˆ = **w***T* **x**

siendo **w** R*n* un vector de par´ametros que controla el comportamiento (output) del sistema ponderando la importancia de cada caracter´ıstica (fea- ture) del sistema representada por las componentes *xi* del input **x**.

*∈*

Para medir la capacidad de nuestro modelo de predecir, dado un conjunto de datos (ejemplos) **X***test* R*m,n* y su valor correcto (etiqueta) **y***test* R*m* calculamos el Error Medio Cuadr´atico (o MSE, Mean Square Error) entre la predicci´on del modelo (vector de regresi´on) **yˆ***test ∈* R*m* y su etiqueta

*∈ ∈*

*m*

*m*

2

*m*

*i*

*i*

MSE

*test*

= 1 *||***yˆ***test −* **y***test||*2 = 1

(*y*ˆ*test − ytest*)2

*i*=1

y consideramos el problema de optimizaci´on dado por la minimizaci´on de la forma cuadr´atica convexa (es una suma de cuadrados luego no negativa)

m´ın MSE

*m*

*∈*

**w***∈*R*n*

*test*

*m*

*i*

= 1 (*y*ˆ*test − ytest*)2

*i*=1

Para construir un algoritmo de aprendizaje autom´atico necesitamos que los pesos **w** R*n* vayan cambiando segu´n vayamos ganando experiencia sobre un conjunto de datos de entrenamiento (**X***train,* **y***train*) siendo nuestras pre- dicciones

*i*

**yˆ***train* = **X***train***w**

de tal forma que se minimice el error MSE*test*. Una manera de hacerlo consiste en minimizar el error MSE*train*:

MSE

*train*

*m*

= 1 *||***yˆ***train −* **y***train||*2 = 1

*m*

(*y*ˆ*train − ytrain*)2

2

*i*

*i*

*i*=1

es decir considerar el problema

m´ın MSE*train*

*m*

**w***∈*R*n*

Para ello es necesario (y suficiente) resolver la ecuaci´on

*∇***w**MSE*train* = 0

lo que nos conduce a las ecuaciones normales y su soluci´on anal´ıtica mediante la pseudo inversa de Moore-Penrose:

**w** = ((**X***train*)*T* **X***train*)*−*1(**X***train*)*T* **y***train*

Una vez obtenidos los pesos podemos predecir la etiqueta *ypred* = *y*ˆ de un nuevo ejemplo **x***new* = **x** mediante un simple producto escalar:

*n*

*y*ˆ = *⟨***w***,* **x***⟩* = **w***T* **x** = **w** *·* **x** = *wixi*

*i*=1

**Ejercicio 5.2.** *Escribir un c´odigo en matlab que genere una design ma- trix (dataset) de m ejemplos de dimensi´on n y un conjunto de etiquetas de entrenamiento. Resolver el problema de regresi´on lineal mediante la pseudo- inversa de More-Penrose. Predecir la etiqueta de un nuevo ejemplo mediante los pesos ´optimos aprendidos.*

En general se entiende por modelo de regresi´on lineal a una ecuaci´on del tipo

*y*ˆ = **w***T* **x** + *b*

en donde *b* R es la *bias* o desviaci´on en referencia a que es la respuesta del sistema frente a un input nulo **x** = **0**

*∈*

##### Aplicaci´on: Calibrado de C´amara

La resoluci´on de un sistema lineal homog´eneo *A***x** = **0** es un problema apa- rentemente m´as sencillo que la resoluci´on de un sistema lineal no homog´eneo del tipo *A***x** = **b**. De hecho podr´ıa considerarse un caso particular del caso no homog´eneo en donde **b** = **0**. Sin embargo la estrategia de resoluci´on del caso homog´eneo es distinta de la que hemos aprendido en el caso no homog´eneo en donde la t´ıpica dificultad es la no existencia de soluciones del sistema.

En el caso homog´eneo siempre existe al menos una soluci´on, la trivial **x** = **0**. Si el sistema est´a bien planteado es la u´nica y, claramente no tiene inter´es (in- formaci´on). Frente a un problema modelado por un sistema lineal homog´eneo el caso interesante es cuando el sistema admite infinitas soluciones. Luego, para convertir el problema en bien planteado habr´a que imponer una restric- ci´on que seleccione, entre las infinitas soluciones la u´nica que cumple con la restricci´on impuesta.

Este camino es an´alogo al aplicado en el caso de sistemas no homog´eneos lo que nos ha llevado al c´alculo de la soluci´on de m´ınimos cuadrados de m´ınima energ´ıa mediante la pseudo-inversa de Moore-Penrose. En el caso homog´eneo la soluci´on trivial **x** = **0** es la que tiene menos energ´ıa por lo cual necesitamos imponer una condici´on que la soluci´on trivial no pueda cumplir.

En este marco, lo que se conoce como el problema de **m´ınimos cuadrados homog´eneo** se reformula como un problema de minimizaci´on con restriccio- nes que se modela con la teor´ıa de los **multiplicadores de Lagrange**[3](#_bookmark88).

Una vez formulado propiamente el problema de minimizaci´on veremos como obtener, mediante un m´etodo algebraico basado en la descomposici´on SVD de una matriz, la soluci´on no trivial del sistema homog´eneo inicial.

Este problema aparece en el problema del calibrado de c´amaras mediante correspondencias de puntos que se introduce, modela y resuelve en la asig- natura de Visio´n Tridimensional.

3Esta teor´ıa no ha sido considerada durante el curso ya que es una extensi´on no trivial de la teor´ıa de la optimizaci´on sin restricciones y requiere, por lo tanto de alguna semana de clases para introducir la programaci´on lineal, la programaci´on cuadr´atica y la teor´ıa de la dualidad que son las herramientas fundamentales para resolver problemas de optimizaci´on con restricciones.

#### El problema modelo

El problema modelo se puede escribir en la forma: Resolver el sistema ho- mog´eneo

*A***x** = **0** (8)

siendo *A* = *B*2*n,*12, **x** = **x**12*,*1, **0** = **0**2*n,*1. El par´ametro *n* indica el nu´mero de correspondencias de (pares) de puntos en la matriz del sistema. El sistema

1. es t´ıpicamente sobredeterminado, es decir 2*n >* 12 y hay m´as ecuaciones (correspondencias entre puntos) que inc´ognitas (grados de libertad). Supon- dremos entonces 2*n >* 12, es decir *n ≥* 7 ya que es el caso relevante.

Observamos que el conjunto de soluciones del sistema homog´eneo ([8](#_bookmark89)) es un espacio vectorial y viene dado por el nu´cleo de la aplicaci´on lineal

*f* : R12 *→* R2*n*

definida por *f* (**x**) = *A***x**. Matem´aticamente hay 2 problemas o escenarios posibles.

* 1. El problema aproximado[4](#_bookmark90). El nu´cleo ker(*fA*) tiene dimensi´on 0, es decir dim(ker(*fA*)) = 0. Entonces la soluci´on trivial es la u´nica soluci´on y la matriz tiene todos sus valores singulares positivos[5](#_bookmark91):

*σ*1 *≥ σ*2 *≥ ... ≥ σ*12 *>* 0

Esto ocurre cuando el rango de *A* es m´aximo, es decir

rank(*A*) = 12 = dim(Im(*A*))

lo que pasa a menudo en problemas pr´acticos. Por la f´ormula de la dimensi´on ([7](#_bookmark75)) se tiene

dim(Ker(*f* )) = 0*.*

Como la u´nica soluci´on es la nula tenemos que resolver el sistema de forma aproximada:

4Se trata del caso m´as comu´n ya que las mediciones (correspondencias) que se recogen en la matriz est´an sujetas a errores (ruido). Si el sistema original es *A***x** = 0 podemos pensar que existe *B* (la matriz exacta) tal que *A B* por lo cual el sistema correcto a resolver es *B***x 0**.

*≈*

*≈*

5El nu´mero de valores singulares positivos es igual al rango de la matriz *A*.

*A***x** *≈* **0***.*

* 1. El problema exacto. El nu´cleo ker(*fA*) tiene dimensi´on positiva, diga- mos

*k* = dim(ker(*fA*)) *≥* 1

Se identifica entonces el nu´cleo con el autoespacio asociado a los valores singulares nulos. Es posible encontrar infinitas soluciones no nulas que verifican exactamente el sistema homog´eneo original *A***x** = **0**. En este caso los u´nicos errores son los de redondeo generados por el algoritmo.

##### El problema aproximado

Normalmente los sistemas sobredeterminados no tienen soluci´on pero al ser homog´eneo la soluci´on existe siempre. Necesitamos caracterizar otra solu- ci´on **x**, no trivial, en el sentido de m´ınimos cuadrados para la cual se tendr´a *A***x 0**. Esta soluci´on se conoce en la literatura como **Least-Squares So- lution of a Homogeneous Linear Systems.**

*≈*

##### El problema de m´ınimos cuadrados

Se considera entonces el problema de m´ınimos cuadrados

arg m´ın *E*(**x**) = arg m´ın

1 2

*||A***x***||* = arg m´ın

1 **x***′A′A***x** (9)

**x***∈*R12

**x***∈*R12 2 2

**x***∈*R12 2

siendo *E*(**x**) la funci´on (cuadr´atica) de la energ´ıa a minimizar.

Buscamos soluciones tales que *||A***x***||*2 es pequen˜a, es decir *||A***x***||*2 *≈* 0 de donde[6](#_bookmark94) se deduce *A***x** *≈* 0 que es la caracterizaci´on de nuestra soluci´on[7](#_bookmark95).

Para resolver ([9](#_bookmark93)) calculamos el gradiente *∇E*(**x**) e imponemos

*∇E*(**x**) = **0**

obteniendo el sistema de ecuaciones normales

6Aqui utilizamos la (tercera) propiedad de la norma de un vector que se anula s´olo si el vector es nulo.

7Si consider´aramos el caso 2) esta estrateg´ıa, es decir buscar soluciones no triviales que verifiquen *A***x** *≈* 0 es sub-´optima. Razonar el porqu´e.

siendo

*A′A***x** = **0** (10)

*P* = *P*12*,*12 = *A′A* = *B′*

12*,*2*n*

*B*2*n,*12*,* **0** = **0**12*,*1

Observa que la matriz *P* = *A′A* es cuadrada y sim´etrica y se corresponde con la matriz hessiana de la forma cuadr´atica que ser´a por tanto convexa o estrictamente convexa (es decir la hessiana es SDP o DP).

Para caracterizar *P* utilizamos el siguiente resultado:

**Lema** Sea *A* una matriz rectangular y sea *A′* la matriz transpuesta. Entonces el determinante de la matriz *A′A* es un nu´mero real no negativo y es igual a 0, si y s´olo si, el rango de *A* es menor que el nm´ ero de columnas.

Puesto que rank(*A*) = 12 podemos concluir que det(*P* ) *>* 0, *P* es DP, la forma cuadr´atica es estrictamente convexa y existe una u´nica soluci´on.

La deducci´on de las conocidas ecuaciones normales se ha explicado en el ejemplo [4.1](#_bookmark80) en donde hemos aprendido a derivar una forma cuadr´atica escri- ta en forma matricial. Se ha utilizado la propiedad de que *P* es sim´etrica.

Las ecuaciones normales son las condiciones necesarias para que el problema de m´ınimos cuadrados tenga un m´ınimo ya que se corresponden a la condi- ci´on necesaria de anulaci´on del gradiente de la funci´on *E*(**x**).

El problema de m´ınimos cuadrados ([9](#_bookmark93)) nuevamente tiene la soluci´on trivial. En los supuestos del caso 1) tenemos unicidad de la soluci´on trivial. La for- ma cuadr´atica asociada a *P* es estrictamente convexa y hay un u´nico m´ınimo trivial.

La estrategia no ha tenido ´exito aparentemente. Sin embargo una simple con- dici´on sobre la norma (longitud) de la soluci´on nos permitir´a *alejarnos* de la soluci´on trivial.

Vamos entonces a imponer que la soluci´on verifique tener norma unitaria, **x** 2 = 1. De esta manera buscaremos una soluci´on que est´a en la (hyper)- superficie de la bola unitaria resolviendo el problema de minimizaci´on

*|| ||*

arg m´ın

**x***∈*R12: *||***x***||*2=1

*||A***x***||*2

(11)

##### El m´etodo algebraico directo

2

El problema de minimizaci´on ([11](#_bookmark96)) se puede resolver de una forma directa utilizando la t´ecnica algebraica de Descomposici´on en Valores Singulares de una matriz (ver la secci´on [3.1](#_bookmark60)). Para calcular una descomposici´on en valo- res singulares de la matriz rectangular *A*2*n,*12 podemos aplicar el algoritmo de matlab mediante el comando [U,S,V]=**svd**(A) y obtener, en output, las matrices

tales que

*U ∈* R2*n,*2*n,* Σ *∈* R2*n,*12*, V ∈* R12*,*12

*A*2*n,*12 = *U*2*n,*2*n*Σ2*n,*12*V*1*′*2*,*12

en donde *U* y *V* son matrices ortogonales y Σ es una matriz diagonal aunque no necesariamente cuadrada (ya que tiene las mismas dimensiones de *A*). Por la definici´on [3.2](#_bookmark61) las columnas de *U* son los **vectores singulares por la izquierda** y las columnas de *V* son los **vectores singulares por la derecha**. El problema a resolver se puede plantear de la forma

arg m´ın

12: *||***x***||*2

*∈*R

**x**

=1 *||A***x***||*2

Aunque no necesario para la aplicaci´on del m´etodo es conveniente considerar una minimizaci´on cuadr´atica

arg m´ın *||A***x***||* = arg m´ın

2

2

**x***′A′A***x** (12)

**x***∈*R12:*||***x***||*2=1 **x***∈*R12: *||***x***||*2=1

y hablar de soluci´on de m´ınimos cuadrados de norma unitaria (**least-square unit-norm solution**).

Antes de resolver el problema observamos que tenemos garantizada la exis- tencia (por la convexidad de la forma cuadr´atica asociada *A***x** 2) pero no la unicidad de soluciones. Matem´aticamente hay unicidad s´olo si la matriz *A* tiene rango m´aximo lo que implica que la forma cuadr´atica asociada *A***x** 2 es estrictamente convexa. Si el rango no es m´aximo (rank deficient) entonces hay infinitas soluciones. Todas son m´ınimos globales.

2

2

*|| ||*

*|| ||*

##### Resoluci´on mediante SVD

En caso de unicidad se calcula la descomposici´on SVD de la matriz del siste- ma *A* y se selecciona la u´ltima columna de la matriz ortogonal *V* . Es decir, la soluci´on se corresponde al vector singular asociado al valor singular menor. En caso de tener infinitas soluciones ´estas se definen a partir de las combi- naciones lineales de las u´ltimas columnas de la matriz ortogonal *V* .

##### Demostraci´on

Empezamos calculando una SVD de la matriz del sistema *A* y definiendo el cambio de coordenadas **y** = *V ′***x**. Este cambio permitir´a *diagonalizar* el sistema en el sentido de la SVD[8](#_bookmark99) y obtener una forma can´onica de la for- ma cuadr´atica asociada. Utilizamos tambi´en la propiedad de las matrices ortogonales de preservar la norma: *||U* **x***||* = *||***x***||*, *∀***x**. Se obtiene

**x***′A′A***x** = *||A***x***||*2 = *||U* Σ*V ′***x***||*2 = *||*Σ*V ′***x***||*2 = *||*Σ**y***||*2 = **y***′*Σ*′*Σ**y**

2

2

2

2

La restricci´on **x** = 1 se transforma en **y** = *V ′***x** = **x** = 1 por la ortogonalidad de *V* . Realizando el producto Σ*′*Σ se obtiene la forma can´onica

*|| || || || || || || ||*

*||*Σ**y***||*2 = **y***′*Σ*′*Σ**y** = *σ*2*y*2 + + *σ*2*y*2

2

1

1

*n*

*n*

El problema ([12](#_bookmark97)) es, en las nuevas coordenadas,

arg m´ın *||*Σ**y***||* = arg m´ın

2

2

*σ*2*y*2 + + *σ*2*y*2

(13)

**y***∈*R12:*||***y***||*2=1 **y***∈*R12:*||***y***||*2=1

1 1

*n n*

La resoluci´on de este problema es puramente intuitiva aunque se pueda for- malizar con las herramientas del c´alculo multivariable.

El arg m´ın de esta cantidad se obtiene concentrando toda la masa **y** 2 (unitaria) donde los valores singulares son m´as pequen˜os. Suponiendo una multiplicidad *k* para el valor singular m´ınimo *σn*, *σn* 0, anulamos las con- tribuciones asociadas a todos los dem´as valores singulares:

*≥*

*|| ||*

*y*1 = *....* = *yn−k* = 0 (14)

8La matriz *A* es rectangular, no cuadrada, luego no se puede hablar de diagonal de la matriz.

Considerando el cambio **y** = *V ′***x**, la ortogonalidad de *V* y las condiciones

([14](#_bookmark98)) se obtiene que la soluci´on buscada **x** es una combinaci´on lineal de las *k*- u´ltimas columnas de la matriz *V* que son los vectores singulares **v***i*, *i* = 1*...k*:

*n*

**x** = *V* **y** = *yi***vi** = *y*1**v**1 + + *yn***v***n* =

*i*=1

*k*

= *αi***vi** = *α*1**v***n−k*+1 + + *αk***v***n*

*i*=1

Los coeficientes *αi*, *i* = 1*...k* verifican

*α*1 = *yn−k*+1*, ...., αk* = *yn, α*2 + *....* + *α*2 = 1*.*

1

*k*

Existen dos casos posibles: unicidad o multiplicidad de soluciones. En este segundo caso el problema est´a mal planteado.

Cuando el rango de la matriz *A* es m´aximo se verifica que *σn >* 0 y el caso m´as comu´n en las aplicaciones es *k* = 1. Por ([14](#_bookmark98)) podemos escribir *y*1 = *...* = *yn−*1 = 0, *α*1 = *yn* = 1, *α*2 = *y*2 = 1. La u´nica soluci´on es, simple-

1

*n*

mente, **x** = **v***n*, la u´ltima columna de la matriz *V* .

Para *k >* 1 hay infinitas soluciones dadas por las combinaciones lineales de las u´ltimas *k* columnas de la matriz *V* :

*k*

**x** = *αi***vi** = *α*1**v***n−k*+1 + + *αk***v***n*

*i*=1

siendo *α*2 + *....* + *α*2 = 1.

1 *k*

##### El Principio de Rayleight

El Principio de Rayleight es una potente herramienta de an´alisis funcional para calcular los autovalores (m´ınimo y m´aximo) de una matriz sim´etrica definida positiva. Se aplica por ejemplo en los problemas en donde se utilizan t´ecnicas del tipo PCA y LDA.

**Teorema 5.1.** *Sea A una matriz sim´etrica definida positiva y sea R*(**x**) *el cociente de Rayleight definido por la funci´on (escalar)*

*Se verifica entonces*

*R*(**x**) =

**x***′A***x**

**x***′***x** *, ∀* **x** *̸*= **0** (15)

m´ın *R*(**x**) = *λ*1*,* m´ax *R*(**x**) = *λn,*

**x x**

*siendo λ*1 *el autovalor m´ınimo y λn el autovalor m´aximo de la matriz A:*

0 *< λ*1 *≤ λ*2 *≤ ... ≤ λn*

Una propiedad importante del cociente de Rayleight *R*(**x**) es su invariancia frente a cambios de escala. Sea **x** un m´ınimo de *R*(**x**) de norma **x** 2 = *c*. Entonces **y** = **x***/c* alcanza el mismo valor m´ınimo y tiene norma unitaria:

*|| ||*

*||***y***||* = 1. En efecto se tiene

*R*(**x**) =

**x***′A***x x***′***x**

(1*/c*2)**x***′A***x**

=

(1*/c*2)**x***′***x**

(**x***′/c*)*A*(**x***/c*)

=

(**x***′/c*)(**x***/c*)

= *R*(**y**)

La consecuencia importante es que podemos buscar, sin perdida de gernera- lidad el m´ınimo de *R*(**x**) en la superficie de la bola unitaria.

##### Resoluci´on

Para aplicar este resultado volvemos a considerar el problema de minimiza- ci´on ([12](#_bookmark97))

arg m´ın *||A***x***||* = arg m´ın

2

2

**x***′A′A***x**

**x***∈*R12:*||***x***||*2=1 **x***∈*R12: *||***x***||*2=1

y consideramos el cociente de Rayleight a la matriz sim´etrica *A′A*

**x***′A′A***x**

*R*(**x**) =

**x***′***x**

(16)

Por las hip´otesis del Principio de Rayleight supondremos *A′A* definida posi- tiva.

Al ser ([12](#_bookmark97)) un problema de minimizaci´on con restricciones definimos el con- junto de admisibilidad de soluciones del problema como

*A* = *{***x** *∈* R12 */ ||x||*2 = 1*}*

2

Por la propiedad de invariancia frente a cambio de escala escribimos entonces

([11](#_bookmark96)) en la forma equivalente

m´ın *||A***x***||*2 = m´ın

2

*||A***x***||*2

2

= m´ın

**x***′A′A***x**

= (17)

**x***∈A*

2

**x***∈A*

*||***x***||*2

**x***∈A*

**x***′***x**

= m´ın *R*(**x**) = m´ın *R*(**x**) = *λ*1

**x***∈A* **x**

Una vez determinado el valor m´ınimo del problema vamos a calcular el **x** = arg m´ın donde se alcanza. Utilizaremos el teorema ([2.5](#_bookmark54)) que nos garantiza que *Toda matriz sim´etrica de coeficientes reales es ortogonalmente diagonalizable y sus valores propios son reales*. El estudio de la diagonalizabilidad se ha realizado en la secci´on [2.2.1](#_bookmark52). Existe por tanto una matriz ortogonal *Q*12*,*12 tal que diagonaliza la matriz *A′A*:

Escribimos

*Q′A′AQ* = *D*

*A′A* = (*Q′*)*−*1*DQ−*1 = *QDQ′*

y calculamos la forma cuadr´atica

**x***′A′A***x** = **x***′QDQ′***x** = (*Q′***x**)*′D*(*Q′***x**)

Observamos tambi´en que

(*Q′***x**)*′*(*Q′***x**) = *||Q′***x***||*2 = *||***x***||*2 = **x***′***x**

2 2

luego el cociente de Rayleight en ([16](#_bookmark101)) es

*R*(**x**) =

**x***′A′A***x x***′***x**

(*Q′***x**)*′D*(*Q′***x**)

=

(*Q′***x**)*′*(*Q′***x**)

Definimos **y** = *Q′***x**, siendo *||***y***||*2 = *||Q′***x***||*2 = *||***x***||*2 = 1. Se tiene

2

2

2

*R*(**y**) = **y***′D***y** = *λ y*2 + *.....* + *λ y*2

**y***′***y** 1 1 *n n*

El rango del cociente es [*λ*1*, λn*] lo que se puede demostrar concentrando toda la masa de la soluci´on en *y*1 = 1 para obtener el valor m´ınimo y en *yn* = 1 para obtener el valor m´aximo. La soluci´on es por tanto **y** = **e**1 = (1*,* 0*,* 0*,* 0*, ...,* 0)*′*, el primer vector de la base can´onica. Volviendo a las variables originales **x** = *Q***y** = *Q***e**1 que es la primera columna de la matriz ortogonal *Q*. Viene dada por el primer autovector de *Q* (asociado al autovalor m´ınimo de *A′A*).

##### Optimizaci´on con restricciones

El problema ([11](#_bookmark96)) es de minimizaci´on con restricci´on. Se puede pasar a un pro- blema sin restricciones introduciendo una nueva inc´ognita *λ* (**multiplicador de Lagrange**) y considerar el problema de minimizaci´on del lagrangiano

arg m´ın *L*(**x***, λ*) = arg m´ın

*||A***x***||* + *λ*(1 *− ||***x***||* ) (18)

(**x***,λ*)*∈*R12*×*R (**x***,λ*)*∈*R12*×*R

2 2

2

2

Observa que la restricci´on *||***x***||*2 = 1 se puede (y debe) escribir en la forma

(diferenciable) *||***x***||*2 = 1. La diferenciabilidad de la energ´ıa (lagrangiano)

2

asegura que los u´nicos puntos cr´ıticos son los de anulaci´on de su gradiente.

Con la restricci´on de norma unitaria ya no hay soluci´on trivial al proble- ma. Adem´as la restricci´on se ha incluido en la funci´on a minimizar luego es un problema de minimizaci´on sin restricciones. El precio a pagar ha sido introducir una nueva inc´ognita *λ* que se tiene que calcular como parte de la soluci´on del problema (**x***, λ*).

Calculamos las condiciones necesarias de I orden dadas por la anulaci´on del gradiente:

*∇ L*(**x***, λ*) = (*∂L*(**x***, λ*)*, ∂L*(**x***, λ*) = **0***.*

(**x***,λ*)

*∂***x**

*∂λ*

Simplificando la constante multiplicativa que se obtiene al calcular el gra- diente del lagrangiano definido en ([65](#_bookmark217)) se obtiene

*∂L*(**x***, λ*) = *A′A***x** *− λ***x** = **0***, ∂L*(**x***, λ*) = 1 *− ||***x***||*2 = 0 (19)

*∂***x**

*∂λ*

2

y la soluci´on no trivial (**x***, λ*) se caracteriza por resolver, necesariamente:

*A′A***x** = *λ***x***, ||***x***||*2 = 1*.*

2

El problema *A′A***x** = *λ***x** es un problema de autovalores para la matriz hessia- na *P* = *A′A* cuyas soluciones son autovectores normalizados (por la restric- ci´on). El multiplicador de Lagrange buscado es un autovalor de la (matriz) hessiana *P* de la forma cuadr´atica. Puesto que *P* = *A′A* se trata de una matriz (al menos) SDP y de una forma cuadr´atica convexa.

Evaluamos la expresi´on del lagrangiano

*L*(**x***, λ*) = *||A***x***||*2 + *λ*(1 *− ||***x***||*2) =

2

2

= **x***′A′A***x** + *λ*(1 *−* **x***′***x**) = **x***′*(*A′A − λI*)**x** + *λ*

a lo largo de las direcciones dadas por los autovectores, es decir **x** tales que existe *λ* tal que *P* **x** = *λ***x**. Se verifica que

*||A***x***||*2 = **x***′A′A***x** = **x***′P* **x** = *λ***x***′***x** = *λ||***x***||*2

luego

2 2

*L*(**x***, λ*) = *λ||***x***||*2 + *λ*(1 *− ||***x***||*2) = *λ*

2 2

por tanto

*L*(**x***, λ*1) = *λ*1 *< L*(**x***, λ*2) = *λ*2 *... < L*(**x***, λ*12) = *λ*12

y el m´ınimo del lagrangiano se alcanza a lo largo de la direcci´on dada por el autovector unitario **x***m* asociados al autovalor m´ınimo *λm* = *λ*1:

**x***m* = argm´ın *L*(**x***, λ*)*, λm* = m´ın *L*(**x***, λ*)

**x***∈*R12 **x***∈*R12

En el caso 1) hemos supuesto el rank(*A*) = 12 luego dim(Ker(*A*)) = 0 (por la f´ormula de la dimensi´on) luego *λ* = 0 no es autovalor[9](#_bookmark102) y *λm >* 0.

9Si lo fuese tendr´ıamos dim(Ker(*A*)) *≥* 1.

## VIII Clase

Una vez aclarada la formulaci´on de un sistema lineal y las herramientas necesarias para establecer que el problem definido est´a bien planteado (v´ease secci´on [4.1.1](#_bookmark71)) pasamos a estudiar los m´etodos num´ericos para su resoluci´on. Veremos los m´etodos directos y los iterativos cuya elecci´on adecuada depende del problema en consideraci´on y su patolog´ıa. Seguiremos en particular el libro de Quarteroni, cap 5. Sistemas Lineales.

#### A´lgebra Num´erica

Existen dos grandes familias de m´etodos y algoritmos para resolver un sis- tema de ecuaciones lineales. Los m´etodos directos, que en un nu´mero finito de operaciones nos dan la soluci´on exacta del sistema (a menos de errores de redondeo) y los m´etodos iterativos que convergen, en un nu´mero infinito de iteraciones a la soluci´on del sistema. Al tener que imponer un criterio de parada del algoritmo se obtendr´a una soluci´on aproximada del sistema. Em- pezamos describiendo los m´etodos directos. El m´as utilizado de los m´etodos directos es el de la factorizaci´on *LU* que describimos a continuaci´on y que se basa en la formulaci´on algoritmica del proceso de eliminaci´on de Gauss.

##### M´etodos Directos: Factorizaci´on LU

Sea dado un sistema lineal de la forma

*A***x** = **b**

siendo *A Mn* = R*n,n* (cuadrada) y vectores de entrada (input) **b** y salida (output) **x** en R*n*. Vamos a suponer la existencia de matrices triangulares inferiores *L* y superior *U* tales que

*∈*

*A* = *LU*

a lo que llamaremos la **factorizaci´on** o descomposici´on *LU* de *A*. Si *A* es no singular lo mismo son *L* y *U* . Mediante esta factorizaci´on para resolver el sistema cuadrado *A***x** = **b** nos reconducimos a la resoluci´on de los 2 sistemas triangulares

*L***y** = **b***, U* **x** = **y**

que son mucho m´as f´aciles de resolver. Ve´amos un simple ejemplo.

**Ejemplo 6.1.** *Sea dada la matriz*

 1 *−*1 2 

*A* =  2 3 3 

0 2 5

*Realizar una Factorizaci´on de tipo LU.*

El proceso se realiza realizando los siguientes pasos. Primera etapa: *k* = 1. Siguiendo el proceso de eliminaci´on de Gauss generamos ceros por debajo del elemento *a*1*,*1 = 1 (al que queremos convertir en cabecera de fila[10](#_bookmark106)). Calculamos los multiplicadores

*l*2*,*1

= *a*2*,*1

*a*1*,*1

= 2*, l*

3*,*1

= *a*3*,*1

*a*1*,*1

= 0*,*

y realizamos las operaciones necesarias entre filas construyendo la matriz

 1 0 0   1 0 0 

*M*1 =  *−l*2*,*1 1 0  =  *−*2 1 0 

*−l*3*,*1 0 1

y aplic´andola (multiplicando) a la matriz *A*

0 0 1

 1 0 0   1 *−*1 2   1 *−*1 2 

*A*1 = *M*1*A* =  *−*2 1 0   2 3 3  =  0 5 1 

0 0 1

0 2 5

0 2 5

Pasamos a la segunda etapa, *k* = 2. Calculamos el u´nico multiplicador

y construimos la matriz

*l*3*,*2

= *a*3*,*2 = 2

*a*2*,*2 5

10Definimos a una cabecera (por filas o columnas) a la primera componente no nula de cada fila o columna. Las cabeceras tienen que estar ordenadas en sentido descreciente, es decir, la cabecera de una fila nunca puede estar en la misma columna de la cabecera anterior.

 1 0 0   1 0 0 

*M*2 =  0 1 0  =  0 1 0 

0 *−l*3*,*2 1

0 *−*2*/*5 1

y multiplicamos por la matriz *A*1 obteniendo la matriz triangular superior *U*

de la factorizaci´on:

1 *−*1 2





*U* = *M*2*A*1 = *M*2*M*1*A* = 0 5 *−*1







0 0 27*/*5

La matriz *L* de la factorizaci´on es simplemente la matriz de los multiplica- dores

 1 0 0   1 0 0 

*L* =  *l*2*,*1 1 0  =  2 1 0 

*l*3*,*1 *l*3*,*2 1

0 2*/*5 1

Observamos que no siempre es posible realizar una factorizaci´on del tipo LU con L triangular inferior y U triangular superior. Dos clases de matrices para las cuales siempre es posible son las: **Sim´etricas Definidas Positivas** y las **Estrictamente Diagonalmente Dominantes por filas** definidas por:

*|ai,i| >*

*n*

*j*= 1*, j̸*=*i*

*|ai,j|, i* = 1*...n*

Si la desigualdad no es estricta tenemos las **Diagonalmente Dominantes por filas** definidas por:

*|ai,i| ≥*

*n*

*j*= 1*, j̸*=*i*

*|ai,j|, i* = 1*...n*

Las matrices tridiagonales o pentadiagonales que se deducen al aplicar la discretizaci´on num´erica en diferencias finitas de las ecuaciones en derivadas parciales que vamos a deducir m´as adelante en problemas de procesamiento de im´agenes digitales, son t´ıpicas matrices Diagonalmente Dominantes por filas.

**Ejercicio 6.1.** *Sea dada la matriz*

 1 1 3 

*A* =  2 2 2 

3 6 4

*Explicar porqu´e no es posible realizar una Factorizaci´on de tipo LU sin per- mutaciones. Verificar la respuesta en Matlab.*

##### La t´ecnica del Pivoteo

Introducimos la t´ecnica del Pivoteo que permite obtener una factorizaci´on del tipo *LU* para cualquier matriz no singular *A*. Siempre basada en el proceso de eliminaci´on de Gauss consiste en permutar las filas para obtener elementos pivotes grandes y multiplicadores pequen˜os (*<* 1) que permitan no propagar excesivamente el error de redondeo. Para ello, en cada etapa, se realiza un **pivoteo parcial por filas** que consiste en poner en primera fila la cabecera de fila mayor. Esta t´ecnica se implementa en matlab con el comando

[*L, U, P* ] = **lu**(*A*)

Se aplica tambi´en en los comandos **det.m**, **inv.m** para el c´alculo de deter- minantes y de inversas.

**Ejercicio 6.2.** *Sea dada la imagen de Barbara que podeis descargar del cam- pus virtual. Convertir la imagen a doble precisi´on y realizar una factoriza- ci´on LU con pivoteo parcial de la imagen. Representar en una u´nica gr´afica y 2 figuras la factorizaci´on LU y la reconstrucci´on de la imagen original A* = *P−*1*LU .*



* + - 1. Reconstrucci´on LU (b) Reconstrucci´on *A* = *P −*1*LU*

Figura 15: Reconstrucci´on de la imagen. A la izquierda reconstruimos me- diante LU y a la derecha considerando tambi´en las permutaciones debidas al pivoteo parcial.

#### A´lgebra Num´erica: M´etodos Iterativos

Tras ver la teor´ıa de las factorizaciones del tipo Choleski, QR, LU que origi- nan los m´etodos de resoluci´on num´ericos directos de resoluci´on de sistemas de ecuaciones lineales pasamos a describir a los m´etodos iterativos.

Empezaremos con la deducci´on del esquema num´erico general a partir del cual podemos deducir los m´etodos m´as conocidos (Jacobi, Gauss-Seidel, Gra- diente, etc) que se pueden clasificar con el nombre de m´etodos de Richardson y que, a la vez, se dividen en Estacionarios y Din´amicos. En particular el m´etodo de Jacobi, el de Gauss-Seidel, la Sobre-relajaci´on (SOR), el m´eto- do del Gradiente, el del Gradiente Precondicionado (PG), el del Gradiente Conjugado (CG), el del Gradiente Conjugado Precondicionado (PCG) y el m´etodo de los residuos m´ınimos generalizado, Gmres.

Finalizaremos resolviendo un sistemas de ecuaciones de Hilbert viendo como los m´etodos iterativos pueden superar las dificultades encontradas con los m´etodos directos.

##### M´etodos de Resolucio´n Iterativos

Sea dado el sistema de Ecuaciones Lineales definido por

*A***x** = **b**

en donde *A* R*n,n* es la matriz cuadrada del sistema, **x** R**n** es el vector de inc´ognitas y **b** R**n** es el vector de datos.

*∈*

*∈ ∈*

Un **m´etodo iterativo** se construye calculando de manera recursiva la ex- presi´on

**x**(*k*+1) = *B***x**(*k*) + **g***, k ≥* 0 (20)

siendo **x**(0) *∈* R*n* un vector inicial. Este procedimiento genera una sucesi´on de vectores *{***x**(*k*)*, k ≥* 0*}* que queremos que converja

l´ım **x**(*k*) = **x**

*k→∞*

a la soluci´on exacta **x** = *A−*1**b** satisfaciendo el sistema

**x** = *B***x** + **g** (21)

Sustituyendo la expresi´on **x** = *A−*1**b** en el esquema num´erico ([21](#_bookmark111)) se tiene

de donde

*A−*1**b** = *BA−*1**b** + **g***,*

**g** = (*I − B*)*A−*1**b***,*

Definimos el **vector de error e**(*k*) cometido en cada etapa como la diferencia entre la soluci´on exacta y la soluci´on del problema iterativo correspondiente:

**e**(*k*) = **x** *−* **x**(*k*)*, k ≥* 0 Restando los sistemas verificados ([21](#_bookmark111)) - ([20](#_bookmark110)) se tiene

**e**(*k*+1) = *B***e**(*k*)

siendo *B* la **matriz de iteraci´on**. Se tiene el siguiente resultado (prop 5.2 pg 150 del libro de Quarteroni) que nos da condiciones necesarias y suficientes para la convergencia del m´etodo iterativo.

**Teorema 6.1.** *Sea dado un m´etodo iterativo del tipo (*[*20*](#_bookmark110)*) cuya matriz de iteraci´on B satisface (*[*21*](#_bookmark111)*). Entonces el m´etodo es convergente s´ı y s´olo s´ı el radio espectral verifica ρ*(*B*) *<* 1*.*

##### Precondicionadores

Dada la matriz *A* siempre la podemos descomponer en la forma

*A* = *P −* (*P − A*)

siendo *P* una matriz no singular que llamaremos el **Precondicionador de**

*A*. Entonces si *A***x** = **b** se tiene

*P* **x** = (*P − A*)**x** + **b**

Multiplicando por *P−*1 se tiene

**x** = *P−*1(*P − A*)**x** + *P−*1**b**

que es de la forma ([21](#_bookmark111)) si

*B* = *P−*1(*P − A*) = *I − P−*1*A,* **g** = *P−*1**b**

Sustituyendo en ([20](#_bookmark110)) se tiene

de donde

**x**(*k*+1) = (*I − P−*1*A*)**x**(*k*) + *P−*1**b**

**x**(*k*+1) *−* **x**(*k*) = *P−*1(**b** *− A***x**(*k*))

luego el esquema iterativo

*P* (**x**(*k*+1) *−* **x**(*k*)) = **b** *− A***x**(*k*) *.*

= **r**

(*k*) (22)

donde **r**(*k*) es el **vector residuo** en la iteraci´on *k*-´esima. Una generalizaci´on natural es el esquema

*P* (**x**(*k*+1) *−* **x**(*k*)) = *αk***r**(*k*)*, k ≥* 0 (23)

donde *αk* = 0 es un par´ametro que podemos utilizar para mejorar las pro- piedades de convergencia del m´etodo. Sea

*̸*

**z**(*k*) = **x**(*k*+1) *−* **x**(*k*)

el vector **residuo precondicionado**. Entonces en cada etapa tenemos que resolver el sistema lineal

y la nueva iteraci´on es

*P* **z**(*k*) = *αk***r**(*k*)*, k ≥* 0

**x**(*k*+1) = **x**(*k*) + *αk***z**(*k*)

Para que el m´etodo sea eficiente es necesario que la matriz *P* sea de un tipo simple, barato computacionalmente, del tipo matriz diagonal, triangu- lar o tridiagonal. En base a esta filosof´ıa podemos definir varios esquemas iterativos.

##### El M´etodo de Jacobi

El M´etodo de Jacobi puede converjer si se dan las dos siguientes condiciones:

1. El sistema a resolver tiene una u´nica soluci´on.
2. Los elementos de la diagonal de la matriz del sistema son no nulos:

*ai,i ̸*= 0.

En estos supuestos definimos el precondicionador *P* dado por la matriz

*P* = *D* = *diag*(*A*) que se obtiene con el comando

D=diag(diag(A));

P=D;

El m´etodo de Jacobi corresponde a la elecci´on *αk* = 1, *k*. Sustituyendo en

*∀*

1. se tiene

de donde

es decir

*D*(**x**(*k*+1) *−* **x**(*k*)) = **r**(*k*) = (**b** *− A***x***k*)*, k ≥* 0

*D*(**x**(*k*+1) *−* **x**(*k*)) = **b** *− A***x***k, k ≥* 0

*D***x**(*k*+1) = **b** *−* (*A − D*)**x***k, k ≥* 0

que es de la forma ([20](#_bookmark110))

**x**(*k*+1) = *D−*1(*D − A*)**x***k* + *D−*1**b***, k ≥* 0

con matriz de iteraci´on

*B* = *D−*1(*D − A*)

El algoritmo de **itermeth** recoge esta formulaci´on definiendo *P* = *J*.

**Teorema 6.2.** *Se tiene que una condici´on suficiente para que el m´etodo de Jacobi converja es que A sea* **estrictamente diagonalmente dominante por filas***.*

**Ejercicio 6.3.** *Sea dado el sistema A***x** = **b** *definido por*

   4 

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 0 | 1 | 1 |
| *−*2 | 4 | 0 | 1 |
| *−*1 | *−*1 | 3 | 1 |
| 0 | 3 | *−*1 | 4 |

*A* = 

 *,* **b** =  3 

   2 

6

*cuya soluci´on exacta es* **x** = (1*,* 1*,* 1*,* 1)*T . Se pide:*

* 1. *Comprobar que la matriz A es diagonalmente dominante por filas. ¿Se puede deducir convergencia mediante el teorema (*[*6.2*](#_bookmark117)*) ?.*
  2. *Calcular la matriz de iteraci´on B del m´etodo de Jacobi y su radio es- pectral. Deducir convergencia mediante el teorema (*[*6.1*](#_bookmark112)*).*
  3. *Resolver el sistema utilizando el m´etodo de Jacobi con semilla inicial* **x**0 = (0*,* 0*,* 0*,* 0)*T , tolerancia* 10*−*12 *y un nu´mero m´aximo de* 100 *itera- ciones.*
  4. *Calcular el nu´mero de iteraciones realizadas, el error absoluto cometido y el tiempo de ejecuci´on del algoritmo.*
  5. *Comparar en t´erminos de error y de rapid´ez con una resoluci´on me- diante m´etodos directos y concluir el mejor m´etodo para este sistema.*

##### El M´etodo de Gauss-Seidel

Puesto que durante la ejecuci´on del algoritmo de Jacobi cada componente del nuevo vector **x**(*k*+1) se calcula independientemente de las otras (y de forma sequencial) es posible aprovechar (actualizar) los nuevos valores segu´n se vayan calculando. Esta idea define el **M´etodo de Gauss-Seidel**. En concreto se considera ([23](#_bookmark115)) con precondicionador

*P* = *D − E, αk* = 1*, k ≥* 0

donde *E* es una matriz triangular inferior cuyos elementos no nulos son

*eij* = *−ai,j, i* = 2*..n, j* = 1*..n −* 1*, j < i.*

Se verifica que es equivalente a definir *P* =tril(*A*) y as´ı se recoge en el c´odigo. La matriz de iteraci´on es

*B* = (*D − E*)*−*1(*D − E − A*)

El par´ametro de entrada en el algoritmo **itermeth** es *P* = *G*. Tambi´en para el m´etodo de Gauss-Seidel existen matrices especiales que garantizan su convergencia.

**Teorema 6.3.** *Las matrices* **estrictamente diagonalmente dominantes por filas** *y las* **sim´etricas definidas positivas** *originan iteraciones del M´etodo de Gauss-Seidel que son convergentes.*

**Ejercicio 6.4.** *Analizar las propiedades de convergencia de los esquemas iterativos de Jacobi y Gauss-Seidel aplicados a la resoluci´on de un sistema lineal definido por la matriz*

 *α* 0 1 

*A* = 

1. *α* 0
2. 0 *α*

 *, α ∈* R

Respuesta: Calculando el determinante de *A* se tiene

*|A|* = *α − α* = *α*(*α −* 1)

3

2

luego buscaremos *α ̸*= *{−*1*,* 0*,* 1*}* para que el sistema gobernado por *A* sea compatible determinado. Ambos m´etodos son convergentes para *|α| >* 1, es decir *α < −*1 y *α >* 1.

**Ejercicio 6.5.** *Analizar las propiedades de convergencia de los esquemas iterativos de Jacobi y Gauss-Seidel aplicados a la resoluci´on de un sistema lineal definido por la matriz*

*A* = *−*10 2 *, β ∈* R

*β* 5

Respuesta: Calculando el determinante de *A* se tiene

*|A|* = *−*2*β −* 50

luego buscaremos *β* = 25 para que el sistema gobernado por *A* sea compa- tible determinado.

*̸ −*

Una condici´on suficiente para la convergencia de los m´etodos de Jacobi y Gauss-Seidel es que *A* sea estrictamente diagonalmente dominante. La se- gunda fila de *A* satisface la condicion de dominancia de la diagonal con tal de que *β <* 5. Esta condici´on no es necesaria. N´otese que si requerimos directamente que los radios espectrales de las matrices de iteraci´on sean me- nores que 1 (que es una condici´on necesaria y suficiente para la convergencia), encontramos la limitaci o´n (menos restrictiva) *|β| <* 25 para ambos m´etodos.

*| |*

**Ejercicio 6.6.** *Sea dado el sistema A***x** = **b** *definido por*

 1 1 0   2 

*A* =  1 6 2  *,* **b** =  9 

0 2 1

3

*cuya soluci´on exacta es* **x** = (1*,* 1*,* 1)*T . Se pide:*

* 1. *Comprobar que la matriz A no es diagonalmente dominante por filas.*
  2. *Deducir convergencia del m´etodo de Gauss-Seidel mediante el teorema (*[*6.3*](#_bookmark119)*).*
  3. *Calcular la matriz de iteraci´on B del m´etodo de Jacobi y su radio es- pectral. Deducir convergencia mediante el teorema (*[*6.1*](#_bookmark112)*).*
  4. *Resolver el sistema utilizando los m´etodos de Jacobi y de Gauss-Seidel con semilla inicial* **x**0 = (0*,* 0*,* 0)*T , tolerancia* 10*−*12 *y un nu´mero m´axi- mo de* 100 *iteraciones.*
  5. *Calcular el nu´mero de iteraciones realizadas, el error absoluto cometido y el tiempo de ejecuci´on de los algoritmos.*
  6. *Comparar en t´erminos de error y de rapid´ez con una resoluci´on me- diante m´etodos directos y concluir el mejor m´etodo para este sistema.*

##### M´etodos de Relajaci´on

Si consideramos ([23](#_bookmark115)) con

1

*P* = *w D − E, αk* = 1*, k ≥* 0

donde *E* es una matriz triangular inferior anterior tenemos un m´etodo de Realajaci´on siendo *w* un **par´ametro de relajaci´on**. La matriz de iteraci´on es

*B* = ( 1 *D − E*

*w*

*−*1 1

*w D − E − A*

(

Se puede deducir que una condici´on necesaria para la convergencia del M´eto- do de Relajaci´on es 0 *< w <* 2. Para *w* = 1 se recupera el M´etodo de Gauss-Seidel. Para 1 *< w <* 2 se tiene un **M´etodo de Sobre-relajaci´on** sucesiva (SOR) del m´etodo de Gauss. El efecto es acelerar la convergencia del m´etodo de Gauss. Para 0 *< w <* 1 se tiene un **M´etodo de Sub-relajaci´on** sucesiva del m´etodo de Gauss. El efecto es favorecer la convergencia de un m´etodo de Gauss no convergente.

**Ejercicio 6.7.** *Se considera el sistema lineal A***x** = **b** *donde* **b** *se elige de tal forma que la soluci´on del sistema es el vector unitario* **x** = (1*,* 1*, ...,* 1)*T y A es la matriz tridiagonal* 10 *×* 10 *definida por*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 3 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | -1 | 3 | -2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -1 | 3 | -2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -1 | 3 | -2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | -1 | 3 | -2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | -1 | 3 | -2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | -1 | 3 | -2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 3 | -2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 3 |

*que se puede obtener utilizando los comandos*

n=10

A=3\*eye(n)-2\*diag(ones(n-1,1),1)-diag(ones(n-1,1),-1)

*Se pide*

1. *Calcular los radios espectrales de las matrices de iteraci´on de los m´eto- dos de de Jacobi y Gauss-Seidel y concluir la convergencia de ambos.*
2. *Resolver el sistema mediante los algoritmos de Jacobi y Gauss-Seidel con semilla inicial* **x**0 = (0*,* 0*,* 0*,* 0)*T , tolerancia* 10*−*12 *y un nu´mero m´aximo de* 1000 *iteraciones.*
3. *Comparar la velocidad de convergencia en t´erminos del nu´mero de ite- raciones.*
4. *Modificando adecuadamente el algoritmo* **itermeth.m** *escribir un pro- grama que calcule una relajaci´on del m´etodo de Gauss-Seidel siendo el par´ametro de relajaci´on w dado en input. Resolver el problema anterior para w* = 1*.*5 *determinando el nu´mero de iteraciones necesarias.*

##### M´etodos de Richardson y del Gradiente

Consideremos ahora m´etodos iterativos de la forma general ([23](#_bookmark115)) (a los que llamaremos m´etodos de Richardson) y para los cuales los par´ametros de aceleraci´on no son constantes. Se trata de m´etodos **din´amicos** si comparado con el de Jacobi o Gauss-Seidel en donde *αk* = 1, *∀ k ≥* 0 que son por tanto m´etodos de Richardson **estacionarios**. En ambos casos el valor de *αk* no puede ser cualquier cosa si queremos convergencia. En concreto se tiene:

**Teorema 6.4.** *Sean P y A matrices* **sim´etricas y definidas positivas***. Entonces el m´etodo de* **Richardson estacionario** *converge s´ı y s´olo s´ı*

2

*en donde*

0 *< α <*

*λ*m´ax

*λmax* = *ρ*(*P−*1*A*) *>* 0

*es el radio espectral de P−*1*A. El valor ´optimo de α que hace que el radio espectral de*

*sea m´ınimo es*

*Bα* = *I − αP−*1*A*

2

*αopt* = *λ*

m´ax

+ *λ*m´ın

*siendo λ*m´ın *el autovalor m´ınimo de P−*1*A.*

**Teorema 6.5.** *las mismas hip´otesis sobre las matrices P y A (***sim´etricas y definidas positivas***) el m´etodo de* **Richardson din´amico** *converge (con- dici´on suficiente) si los αk se eligen de la forma*

(**z**(*k*))*T* **r**(*k*)

*αk* = (**z**(*k*))*T A***z**(*k*) *, k ≥* 0

*siendo* **z**(*k*) = *P−*1**r**(*k*) *el residuo precondicionado.*

El m´etodo de Richardson din´amico se llama **M´etodo del Gradiente Pre- condicionado** si *P* = *I* y **M´etodo del Gradiente** si *P* = *I*. La versi´on din´amica no requiere el c´alculo de los autovalores extremos de *P−*1*A* y utiliza s´olo cantidades ya disponibles resultando conveniente computacionalmente.

*̸*

Utilizando el algoritmo **itermeth.m** si *P* no se prescribe el programa asume *P* = *I* y se tiene el m´etodo del Gradiente. Si se prescribe *P* = *I* se tiene el m´etodo del Gradiente Precondicionado.

*̸*

##### Algoritmo del Gradiente

Sea *A* una matriz cuadrada, sim´etrica y definida positiva. El **m´etodo del gradiente** se puede entender convirtiendo el problema lineal

*A***x** = **b**

en el problema de minimizaci´on cuadr´atico

m´ın *f* (**x**) = m´ın (1 **x***T A***x** *−* **b***T* **x** + *c*

ya que

**x x** 2

1 *T*

*∇f* (**x**) = 2 (*A* + *A* )**x** *−* **b** = *A***x** *−* **b**

luego si resolvemos *f* (**x**) = **0** se tiene una soluci´on del sistema *A***x** = **b** puesto que *A* es sim´etrica por hip´otesis. Por hip´otesis la forma cuadr´atica es estrictamente convexa (*A* es DP) luego existe u´nico **x***∗* tal que

*∇*

*∇f* (**x***∗*) = **0**

luego

siendo

*A***x***∗* = **b**

**x***∗* = arg m´ın *f* (**x**)

**x**

Veamos que pasa si las hip´otesis sobre *A* no se cumplen. Si *A* no es sim´etrica entonces no se tiene una soluci´on del problema original

sino del sistema

*A***x** = **b**

1 (*A* + *AT* )**x** = **b**

2

Si *A* es sim´etrica pero definida negativa tenemos un m´aximo en lugar de un m´ınimo. Si *A* es sim´etrica pero indefinida tenemos un punto de silla. Veamos come se explica el algoritmo del gradiente, en t´erminos de la iteraci´on

**x**(*k*+1) = **x**(*k*) + *αk***z**(*k*)

Si definimos **z**(*k*) = *f* (**x**(*k*)) (vamos en la direcci´on contraria al gradiente) se tiene la iteraci´on

*−∇*

**x**(*k*+1) = **x**(*k*) *− αk∇f* (**x**(*k*))

en donde podemos calcular expl´ıcitamente los coeficientes *αk* que aseguran el descenso m´as r´apido conocido como **steepest descent**. Para ello definimos el error entre cada iterada **x***k* y la soluci´on **x***∗* y el residuo en cada iteraci´on en la forma

Se tiene

**e***k* = **x** *−* **x***k,* **r***k* = **b** *− A***x***k* = *A***e***k*

*−∇f* (**x** ) = **b** *− A***x** = **r**

(*k*+1) (*k*+1) *k*+1

Imponiendo la condic´on de ortogonalidad entre residuales sucesivos que se corresponde a la ortogonalidad del gradiente con las curvas de nivel se tiene

*⟨***r** *,* **r** *⟩* = **r** *·* **r** = (**r** ) **r** = **0**

*k*+1 *k k*+1 *k k*+1 *T k*

Desarrollando la definici´on

y escribiendo la iteraci´on

**r***k*+1 = **b** *− A***x**(*k*+1)

se tiene

**x**(*k*+1) = **x**(*k*) *− αk∇f* (**x**(*k*+1)) = **x**(*k*) + *αk***r***k*

**r***k*+1 = **b** *− A***x**(*k*+1) = **b** *− A*(**x**(*k*) + *αk***r***k*) = **r***k − αkA***r***k*

Sustituimos esta expresi´on en la condici´on de ortogonalidad

*⟨***r***k*+1*,* **r***k⟩* = *⟨***r***k − αkA***r***k,* **r***k⟩* = *⟨***r***k,* **r***k⟩ − αk⟨A***r***k,* **r***k⟩*

de donde el valor o´ptimo

**r***k,* **r***k*

*⟨ ⟩*

*αk* = *⟨A***r***k,* **r***k⟩*

Confronta este valor con el valor ´optimo que aparece en el teorema ([6.5](#_bookmark122)) para *P* = *I*. Una de las limitaciones del m´etodo del gradiente radica en su comportamiento bastante lento ya que zizagea entre las curvas de nivel. Esta limitaci´on se supera con los m´etodos de gradiente conjugado.

##### El M´etodo del Gradiente Conjugado: CG y PCG

Para matrices *A* **sim´etricas y definidas positivas** el m´etodo del Gradiente Conjugado Precondicionado (PCG) es uno de los m´as efectivos para resolver sistemas lineales sim´etricos definidos positivos muy grandes y huecos (sparse). El algoritmo est´a implementado en Matlab mediante el comando **pcg.m**

##### Sistemas No Sim´etricos: M´etodos de Krylov. GMRES

Si el sistema es grande, hueco y no sim´etrico un m´etodo muy efectivo es el de Residuos M´ınimos Generalizados, conocido como Gmres. Es parte de los algoritmos de Matlab y se implementa mediante el comando **gmres.m**

**Ejercicio 6.8.** *Sea dada la matriz de hilbert H* = *H*(16) *(de orden* 16*) definida mediante el comando* **hilb(16)***. Sea* **x***∗* = **1** *la soluci´on del sistema asociada a los datos* **b** = *H***x***∗. Se pide*

1. *Calcular su condicionamiento*
2. *Resolver el sistema asociado H***x** = **b** *mediante un m´etodo directo uti- lizando el operador slash (caja negra).*
3. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo de Jacobi.*
4. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo de Gauss- Seidel.*
5. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo de Gauss- Seidel relajado. ¿Te conviene hacer sobre-relajaci´on o sub-relajaci´on*

*?*

1. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente.*
2. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente Precondicionado con precondicionador de Jacobi*
3. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente Conjugado.*
4. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente Conjugado Precondicionado con precondicionador de Cholesky.*
5. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente Conjugado Precondicionado con precondicionador de Cholesky incom- pleto.*
6. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gradiente Conjugado Precondicionado con precondicionador de Cholesky incom- pleto modificado.*
7. *Resolver el sistema asociado H***x** = **b** *mediante el m´etodo del Gmres.*
8. *Comentar los resultados.*

## IX Clase

#### C´alculo y Optimizaci´on Multi-variable

Entramos ahora a definir los fundamentos matem´aticos de c´alculo de una y varias variables que nos proporcionan las herramientas necesarias para la op- timizaci´on de funciones, un problema b´asico en Visi´on artificial y Aprendiza- je Automa´tico. Empezaremos extendiendo al c´alculo multivariable conceptos b´asicos del an´alisis real de funciones de una variable.

##### Introduccio´n y Resultados B´asicos

Junto con el A´lgebra, el C´alculo multi-variable es la otra herramienta funda- mental para trabajar en Visi´on Artificial y Aprendizaje Autom´atico.

Una lista de Conceptos b´asicos de C´alculo multi-variable puede ser como sigue:

* + - 1. Dominios y Continuidad. Curvas de nivel.
      2. Diferenciabilidad, Derivadas Direccionales y Gradiente.
      3. Desarrollos de Taylor. Aproximaci´on Lineal y Cuadr´atica.
      4. Matriz Jacobiana y Matriz Hessiana.
      5. Representaci´on Gr´afica utilizando Matlab.
      6. Optimizaci´on sin Restricciones: Condiciones Necesarias de I orden.
      7. Optimizaci´on sin Restricciones: Condiciones Suficientes de I orden.
      8. Optimizaci´on con Restricciones de igualdad: Multiplicadores de La- grange.
      9. Optimizaci´on con Restricciones de igualdad y desigualdad: Multiplica- dores de KKT.

Durante la clase revisaremos u´nicamente aquellos conceptos que se utilizan en aprendizaje autom´atico, tal y como aparece en el cap´ıtulo 4, Numerical Computation, secci´on 4.3 Gradient Based Optimization del libro Online de Deep Learning de Ian Goodfellow, Yoshua Bengio and Aaron Courville que se puede encontrar en la direcci´on:

[http://www.deeplearningbook.org](http://www.deeplearningbook.org/)

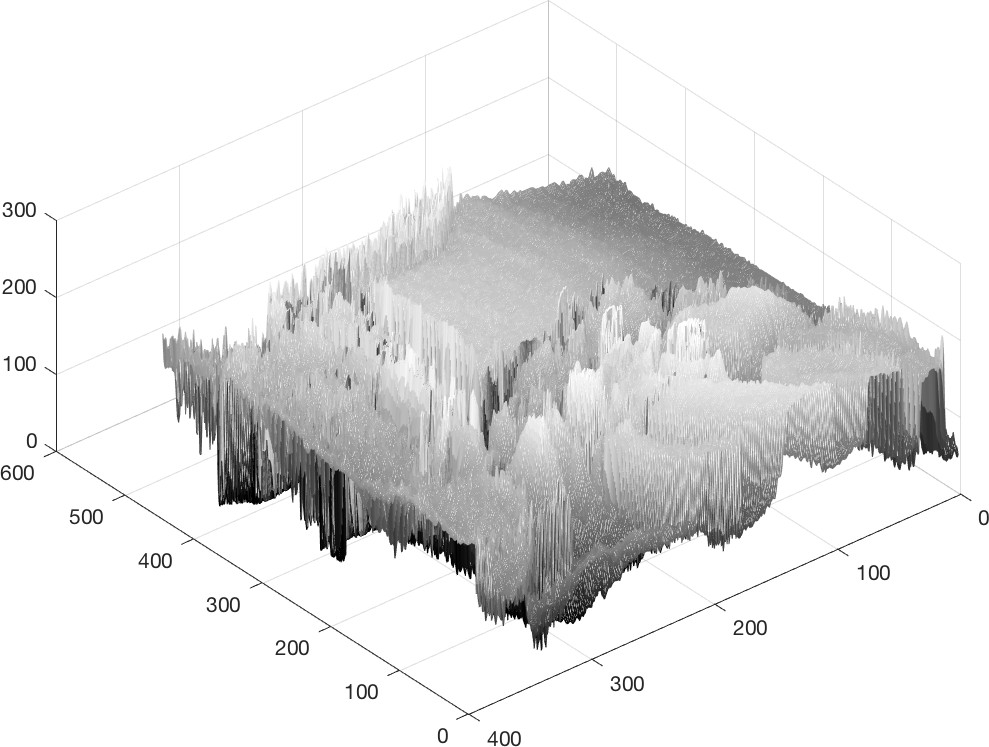


Figura 16: Gr´afica 3*D* de la imagen del cuadro de Dali. Se aprecian las discontinuidades en las zonas de gradiente elevado



Figura 17: Imagen del cuadro de Dali

##### Optimizaci´on basada en el c´alculo del gradiente

La mayor´ıa de los algoritmos de aprendizaje autom´atico tienen como objetivo realizar un proceso de optimizaci´on (minimizaci´on o maximizaci´on) de alguna funci´on escalar *f* (**x**) R o vectorial **f** (**x**) R*n* siendo **x** R*n* la variable vectorial a determinar. La notaci´on *f* define campos escalares mientras que **f** denota campos vectoriales. T´ıpicamente los algoritmos est´a disen˜ados para minimizar una funci´on, escalar o vectorial. La maximizaci´on se puede realizar a partir de un algoritmo para minimizaci´on minimizando el valor de *f* (**x**) R o, en el caso vectorial, **f** (**x**) R*n*. A la funci´on que queremos optimizar se le llamar´a **funci´on objetivo** o criterio, funci´on de coste, funci´on de p´erdida o funci´on de error. El uso de una u otra terminolog´ıa suele depender del enfoque

*— ∈*

*— ∈*

*∈ ∈ ∈*

(analisis de funciones, an´alisis funcional, an´alisis estad´ıstico o aprendizaje autom´atico) y ser´a aclarado en cada ejemplo. En general todos estos t´erminos definen el mismo problema de optimizaci´on. El valor de la variable **x** R*n* en la cual se encuentra el o´ptimo se suele denotar por **x***∗* R*n* y se define formalmente por:

*∈*

*∈*

siendo

el valor ´optimo.

**x***∗* = arg m´ın *f* (**x**) = arg m´ax(*−f* (**x**))

**y***∗* = *f* (**x***∗*) = m´ın *f* (**x**)

##### Optimizaci´on 1D

Empezamos con el an´alisis real de funciones reales, es decir consideremos *f* : R R tal que *y* = *f* (*x*) con *x* input e *y* output variables reales. Denotamos su derivada por *f′*(*x*) o *df /dx* lo que nos da la pendiente de la recta tangente a la gr´afica de *f* en el punto *x*. Usando una aproximaci´on lineal de Taylor:

*→*

*f* (*x* + *ϵ*) = *f* (*x*) + *ϵf′*(*x*)*,* 0 *< ϵ ≪* 1

podemos relacionar un pequen˜o cambio en el input con el correspondiente cambio en el output. La derivada nos permite por tanto entender como te- nemos que cambiar un poquito *x* si queremos minimizar *f* (*x*) obteniendo un output *y* menor.

Podemos observar estas propiedades en el siguiente ejemplo

**Ejemplo 7.1.** *Sea dada la funci´on*

*f* (*x*) = *−e−x*(*x*3 *− x*)

*Se pide:*

* + - 1. *Definir la funci´on en matlab usando c´alculo simb´olico,*
      2. *Calcular una aproximaci´on lineal de la funci´on centrada en el punto*

*x*0 = 0 *mediante el polinomio aproximante de taylor de primer grado.*

* + - 1. *Calcular una aproximaci´on cuadr´atica de la funci´on centrada en el pun- to x*0 = 0 *mediante el polinomio aproximante de taylor de segundo grado grado.*
      2. *Calcular el error relativo cometido con las dos aproximaciones estiman- do el valor de la funci´on en el punto x∗* = *x*0 + *ϵ* = 0*.*1*.*
      3. *Representar gr´aficamente la funci´on y sus aproximaciones.*

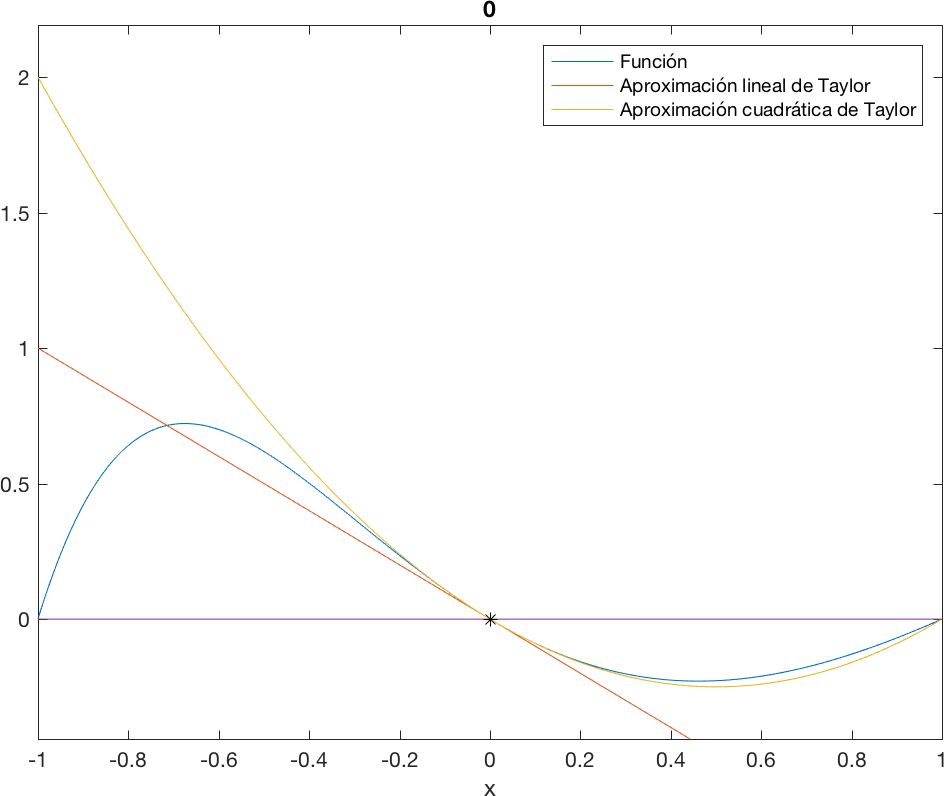


Figura 18: Gr´afica de la funci´on y sus aproximaciones lineal y cuadr´atica en el punto *x*0 = 0

El c´odigo matlab para producir las gr´aficas es el siguiente:

syms x

f=(x.^3-x).\*exp(-x) eje=0.\*x

p1=taylor(f,x,’ExpansionPoint’, 0,’Order’,2) p2=taylor(f,x,’ExpansionPoint’, 0,’Order’,3)

%% error en un punto x\_0=0.1

exacto=double(subs(f,x\_0)) L=double(subs(p1,x\_0)) C=double(subs(p2,x\_0)) error\_L=abs(exacto-L)/abs(exacto) error\_C=abs(exacto-C)/abs(exacto)

%% grafica a=1

figure ezplot(f,[-a,a]) hold on

ezplot(p1,[-a,a]) hold on ezplot(p2,[-a,a]) hold on ezplot(eje,[-a,a]) hold on plot(0,0,’k\*’) legend(’Funci´on’,

’Aproximacion lineal de Taylor’ , ’Aproximacion cuadratica de Taylor’)

Utilizando las propiedades de monoton´ıa asociadas al signo de una derivada (*f′*(*x*) *>* 0 indica una funci´on mon´otona estrictamente creciente y *f′*(*x*) *<* 0 indica una funci´on mon´otona estrictamente decreciente) es posible demostrar que

*f* (*x − ϵ* sign(*f′*(*x*))) *< f* (*x*)*,* 0 *< ϵ ≪* 1 En efecto, si *f′*(*x*) *<* 0 entonces lo anterior es

*f* (*x* + *ϵ*) *< f* (*x*)*,* 0 *< ϵ ≪* 1 Si *f′*(*x*) *>* 0 entonces lo anterior es

*f* (*x − ϵ*) *< f* (*x*)*,* 0 *< ϵ ≪* 1

Podemos por tanto reducir el valor de *f* movi´endonos unos pequen˜os pasos (la informaci´on de la derivada es local) en direcci´on opuesta a la derivada *f′*(*x*): si es positiva me voy a la izquierda y si es negativa me voy a la dere- cha. Esta t´ecnica, muy antigua (Cauchy, 1847) se conoce con el nombre de **descenso gradiente**. Cuando *f′*(*x*) = 0 el criterio de la derivada nos dice de no movernos: tenemos **puntos cr´ıticos o estacionarios**. Un **m´ınimo local** es un punto tal que no es posible minimizar ulteriormente la funci´on movi´endonos en un entorno del punto. Un **m´aximo local** es un punto tal que no es posible maximizar ulteriormente la funci´on movi´endonos en un entorno del punto. Los puntos cr´ıticos que no son extremos se conocen co- mo **puntos de silla**. Un punto en donde se alcanza el valor m´ınimo entre todos los puntos de m´ınimo local se define como **m´ınimo global**. El punto en donde se alcanza el m´ınimo global puede ser u´nico, puede haber varios o

pueden ser infinitos. Obviamente puede haber m´ınimos locales en donde no se alcanza un m´ınimo global y esto es lo que suele pasar en deep learning en donde aparecen m´ınimos locales y tambi´en puntos de silla.

Veamos un ejemplo cuyos resultados se visualizan en la figura de abajo

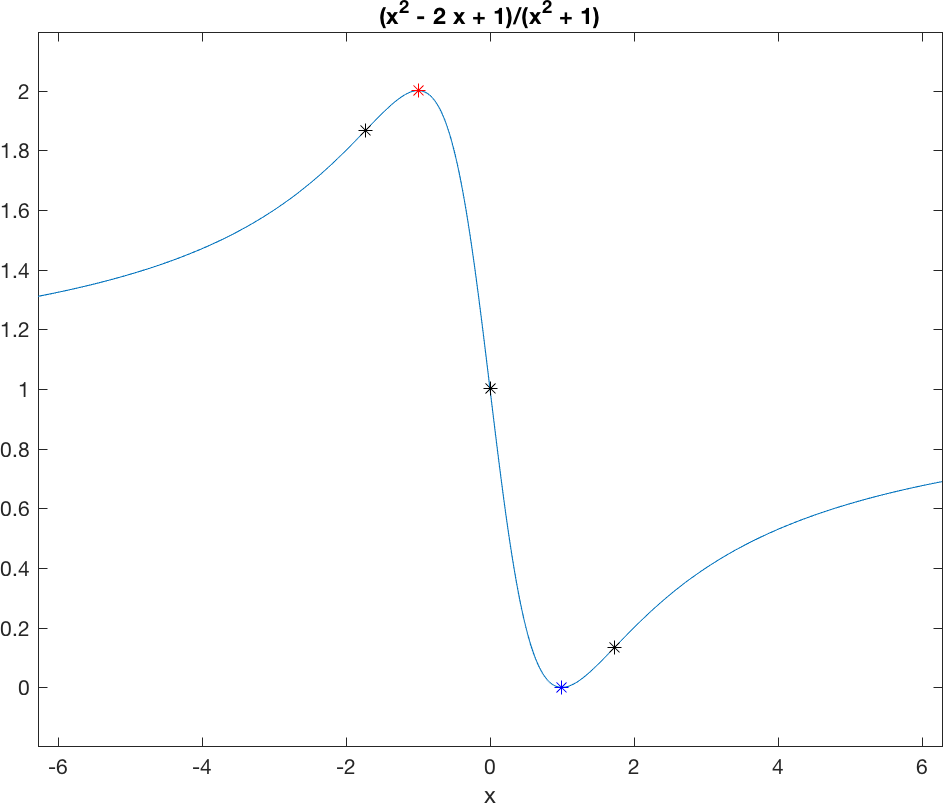


Figura 19: Gr´afica de la funci´on y clasificaci´on de sus puntos cr´ıticos. En rojo el punto de m´aximo absoluto en *x* = 1, en azul el punto de m´ınimo absoluto en *x* = 1 y en color negro los puntos de silla que no son extremos. Observa que la funci´on no es convexa sin embargo tiene un u´nico punto de m´ınimo y de m´aximo y los extremos son absolutos, globales.

*−*

**Ejemplo 7.2.** *Sea dada la funci´on*

*f* (*x*) =

*x*2 *−* 2*x* + 1

*x*2 + 1

*Se pide:*

1. *Calcular sus puntos cr´ıticos.*
2. *Clasificarlos en puntos de m´aximo, m´ınimo (extremos relativos) e in- flexi´on.*
3. *Representar la gr´afica de la funci´on y sus puntos cr´ıticos.*

Para la clasificaci´on de los puntos cr´ıticos de una funci´on real de variable real es posible utilizar la informaci´on de la segunda derivada (cuando existe) *f ′′* (*x*) que define la **curvatura** de la gr´afica de una funci´on.

Sea *x*0 un punto cr´ıtico de una funci´on diferenciable. Entonces

*f′* (*x*0) = 0

es decir tenemos la anulaci´on de la derivada en el punto. Si adem´as

*f ′′* (*x*0) *>* 0

la funci´on es **convexa** y su gr´afica tiene **curvatura positiva**: *x*0 es un punto de m´ınimo. Un simple ejemplo es la funci´on cuadr´atica (par´abola) *f* (*x*) = *x*2. Si por el contrario

*f ′′* (*x*0) *<* 0

la funci´on es **concava** y su gr´afica tiene **curvatura negativa**: *x*0 es un punto de m´aximo. Un simple ejemplo es la funci´on cuadr´atica (par´abola) *f* (*x*) = *−x*2.

##### Optimizaci´on multi-variable

La optimizaci´on en deep learning es algo m´as dificil considerando que el input *x* es multidimensional: **x** R*n*. Para entrar en materia consideremos por tanto funciones (campos) escalares de varias variables:

*∈*

*f* : R*n →* R

tal que *y* = *f* (**x**) con **x** R*n* input multidimensional e *y* output variable real. De manera an´aloga a lo hecho anteriormente introducimos por tanto el concepto de **derivada parcial** que mide las variaciones de la funci´on cuando nos movemos en la direcci´on definida por una componente *xi* del vector **x** = (*x*1*, .., xn*) *∈* R*n*. La derivada parcial en la direcci´on i-´esima se denota por *∂f/∂xi* y el conjunto de todas las derivadas parciales define el **vector gradiente** *f* (**x**) R*n*. Observa que el gradiente de una funci´on de varias variables es una funci´on (campo) vectorial:

*∈*

*∇ ∈*

*∇f* : R*n →* R*n.*

**Ejemplo 7.3.** *Sea dada la funci´on (campo escalar) f* : R2 *→* R *definida por*

*f* (*x, y*) = ln *x*2 + *y*2 + 1

*Calcular su gradiente f* (*x, y*)*. Para ello, observamos que la funci´on est´a definida y es diferenciable en el plano. El campo vectorial de gradientes*

*∇*

*∇f* : R2 *→* R2

*es*

*∇f* (*x, y*) = (*∂f , ∂f* = ( 2*x ,* 2*y ∈* R2

*∂x*

*∂y*

*x*2 + *y*2 + 1

*x*2 + *y*2 + 1

*La u´nica soluci´on del sistema de ecuaciones no lineales*

*∇f* (*x, y*) = (0*,* 0)*T*

*es el origen luego el u´nico punto cr´ıtico es* (*x*0*, y*0) = (0*,* 0)

Las gr´aficas anteriores se pueden obtener mediante el c´odigo propuesto a continuaci´on. El sistema de ecuaciones no lineales dado por la anulaci´on del campo de gradientes se resuelve mediante el comando **solve.m**

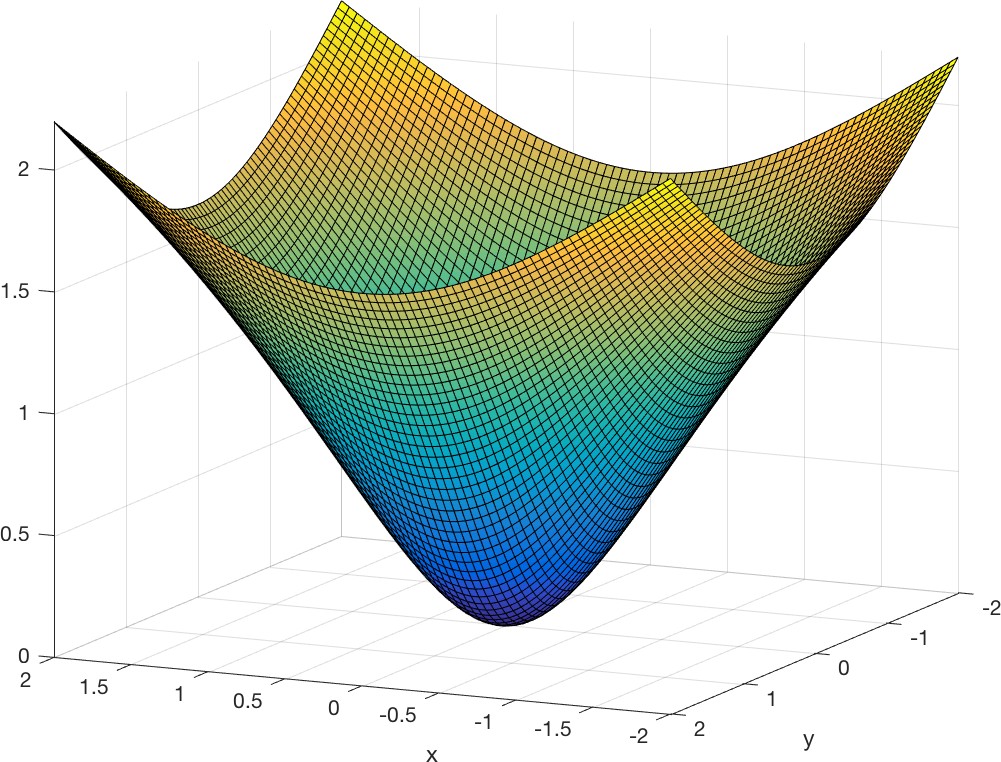
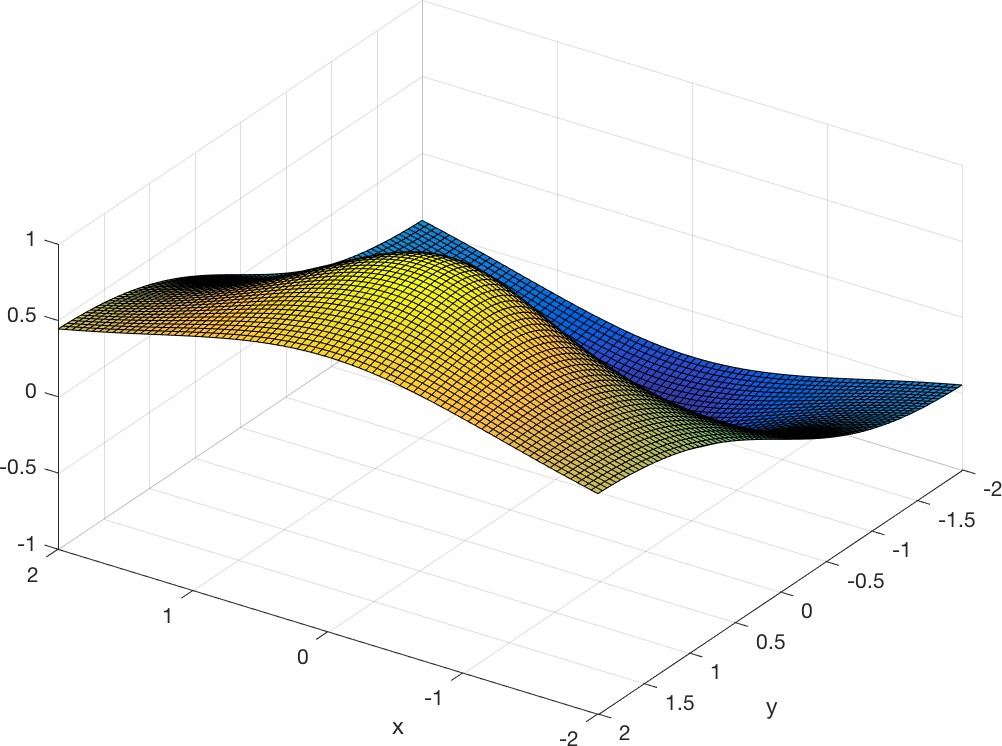
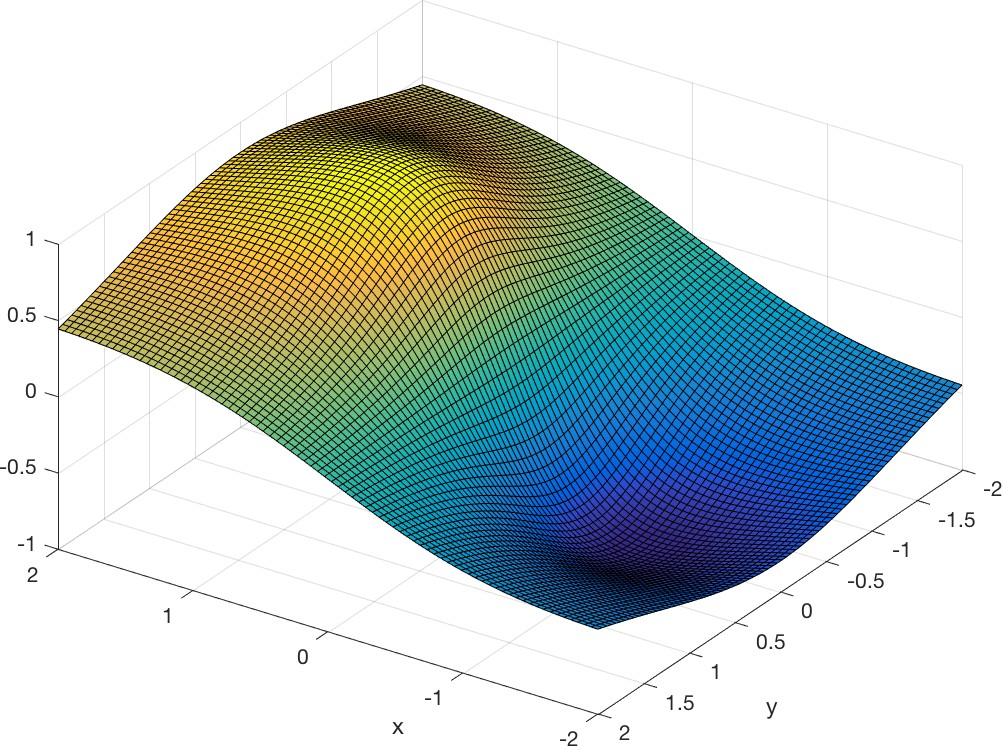


Figura 20: Gr´afica de la funci´on *f* (*x, y*). Es evidente la convexidad estricta de la funci´on y la existencia de un u´nico punto de m´ınimo global



* 1. Derivada parcial primera en *x* (b) Derivada parcial primera en *y*

Figura 21: Gr´aficas de las Derivadas Parciales. A la izquierda el campo

*fx*(*x, y*) y a la derecha el campo*fy*(*x, y*)

syms x y f=log(x^2+ y^2+1);

% gradiente fx=diff(f,x);

fy=diff(f,y);

G=[fx;fy];

% puntos criticos

[xcr,ycr]=solve(fx,fy); [xcr,ycr]

%%

a=2 figure

fsurf(f,[-a,a,-a,a],’MeshDensity’,80) xlabel(’x’);

ylabel(’y’);

view(-155,14)

%%

figure

fsurf(fx,[-a,a,-a,a],’MeshDensity’,80) xlabel(’x’);

ylabel(’y’);

view(-146,44)

%%

figure

fsurf(fy,[-a,a,-a,a],’MeshDensity’,80) xlabel(’x’);

ylabel(’y’);

view(-146,44)

%title(’ Parcial y’)

**Ejemplo 7.4.** *Sea dada la funci´on f* (*x, y*) = *xe−x*2*−y*2 *. Representamos su gr´afica que es una superficie en* R3 *y a continuaci´on el campo vectorial de gradientes*

En el caso multi-variable los puntos cr´ıticos son los puntos **x** R*n* en donde se satisface la ecuaci´on vectorial (que es un sistema t´ıpicamente no lineal):

*∈*

*∇f* (**x**) = **0** *∈* R*n* (24)

El sistema no lineales ([24](#_bookmark131)) se conoce como condici´on necesaria de primer or- den para funciones diferenciables. Si observamos que las derivadas parciales son derivadas en direcci´on de los ejes coordenados podemos considerar un concepto m´as general que incluye el de derivada parcial: el de **derivada di- reccional**. Dada una direcci´on cualquiera del espacio definida por un vector **u** *∈* R*n* podemos calcular la derivada direccional de una funci´on en el pun- to **x**0 *∈* R*n* y en la direcci´on del vector **u**, denotada por *f′*(**x**0*,* **u**). Se suele considerar s´olo vectores unitarios: *||***u***||* = 1. Si una funci´on es **diferenciable**

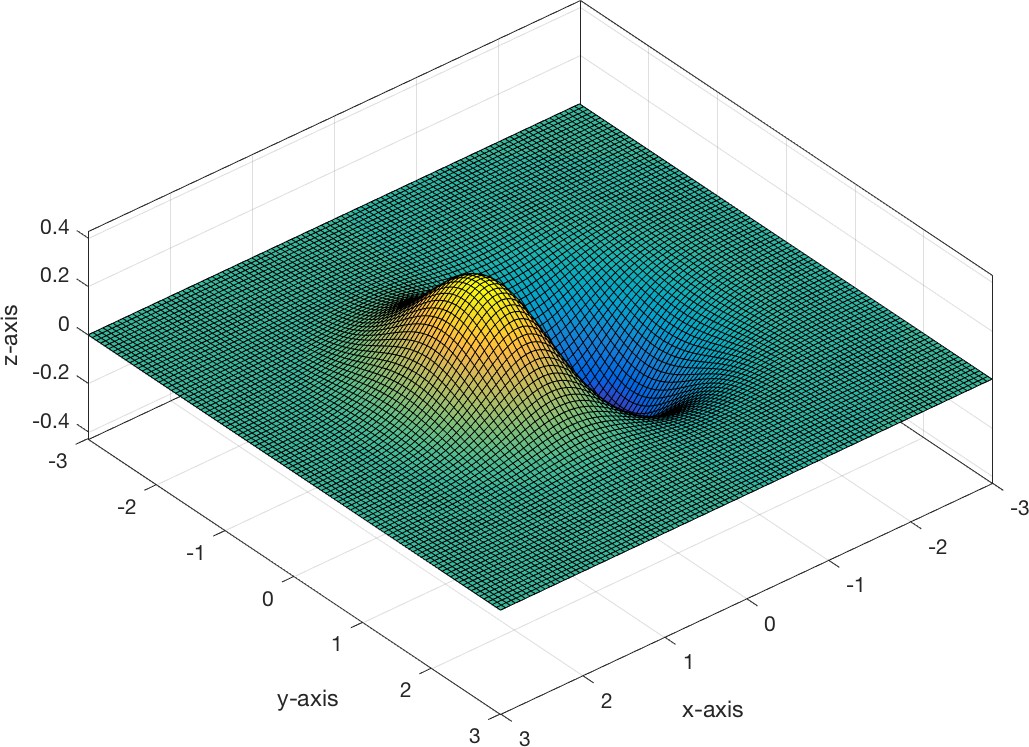


Figura 22: Gr´afica de la funci´on *f* (*x, y*).

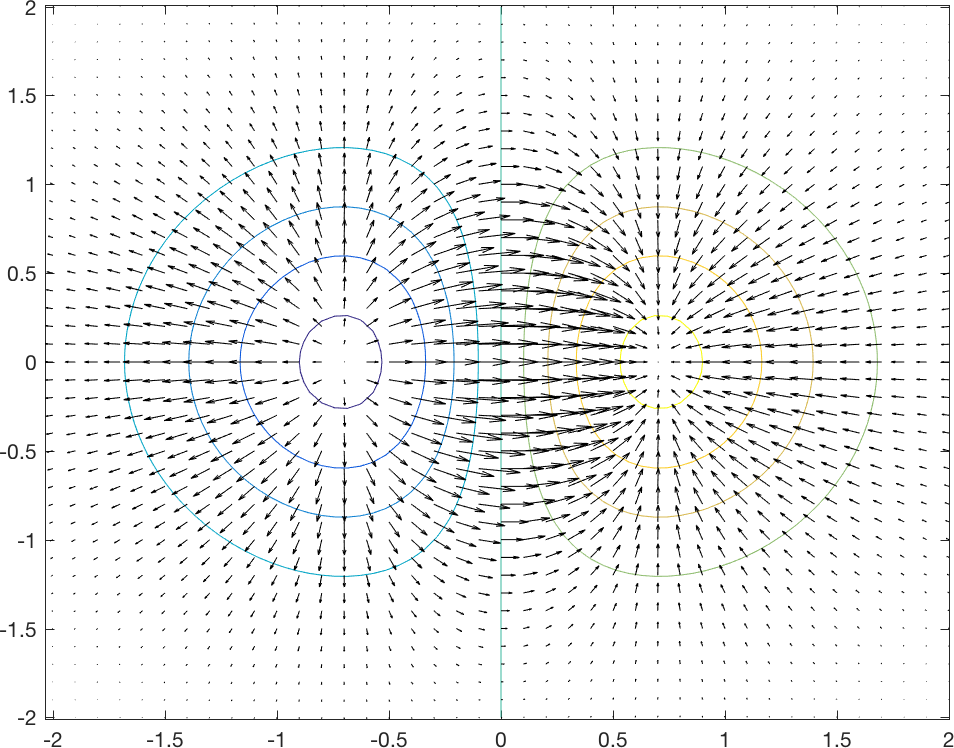


Figura 23: Gr´afica del campo vectorial de gradientes de la funci´on *f* (*x, y*). Los vectores definidos por el campo de gradientes salen del punto de m´ınimo y se dirigen al punto de m´aximo.

existen todas las derivadas direccionales pero la condici´on es suficiente y no necesaria. Pueden existir todas las derivadas direccionales sin que la funci´on sea diferenciable. Para conocer las condiciones que aseguran la diferenciabili- dad y eventualmente la existencia de las derivadas direccionales consultar las notas de c´alculo. Adema´s si una funci´on es **diferenciable** existe una simple forma de calcular la derivada direccional de una funci´on dada por el producto escalar:

*f′*(**x**0*,* **u**) = **u** *· ∇f* (**x**0) = **u***T ∇f* (**x**0) = *⟨***u***, ∇f* (**x**0)*⟩*

Para minimizar *f* , nuestro objetivo, vamos a movernos en la direcci´on de m´aximo decrecimiento de la funci´on. Para ello minimizamos:

m´ın

**u***∈*R*n, ||***u***||*=1

*f′*(**x**0*,* **u**) = m´ın

**u***∈*R*n, ||***u***||*=1

**u***T ∇f* (**x**0) =

= m´ın

**u***∈*R*n, ||***u***||*=1

*||***u***|| ||∇f* (**x**0)*||* cos(*θ*) = **u***∈* m´ın

= *−||∇f* (**x**0)*||*

R*n, ||***u***||*=1

*||∇f* (**x**0)*||* cos(*θ*) =

si tomamos cos(*θ*) = *−*1 lo que implica *θ* = *π*: es decir, la direcci´on de m´aximo decrecimiento es la tomada por un vector **u** = *f* (**x**0) que va en la misma direcci´on del gradiente pero en sentido contrario. Este m´etodo se conoce con el nombre de **steepest descent** (descensos m´as r´apidos) o simplemente **gradient descent** (descenso del gradiente).

*−∇*

##### M´etodo del Descenso

Utilizando los razonamientos anteriores que sugieren, dada una semilla ini- cial, movernos en la misma direcci´on pero en el sentido contrario al gradiente, disen˜aremos un m´etodo de descenso gradiente. Computacionalmente pode- mos por tanto definir un paso del algoritmo de **descenso gradiente** en la forma:

**x** = **x**0 *− ϵ∇f* (**x**0) (25)

en donde **x** es el nuevo punto que predice el algoritmo.

El escalar *ϵ >* 0 es la **learning rate** o tasa de aprendizaje y su correcta determinaci´on es uno de los puntos m´as complicados a la hora de aplicar un algoritmo de deep-learning. En primera opci´on se toma 0 *< ϵ* 1 constante. Actualmente se opta por una elecci´on adaptativa. Otra estrategia es realizar

*≪*

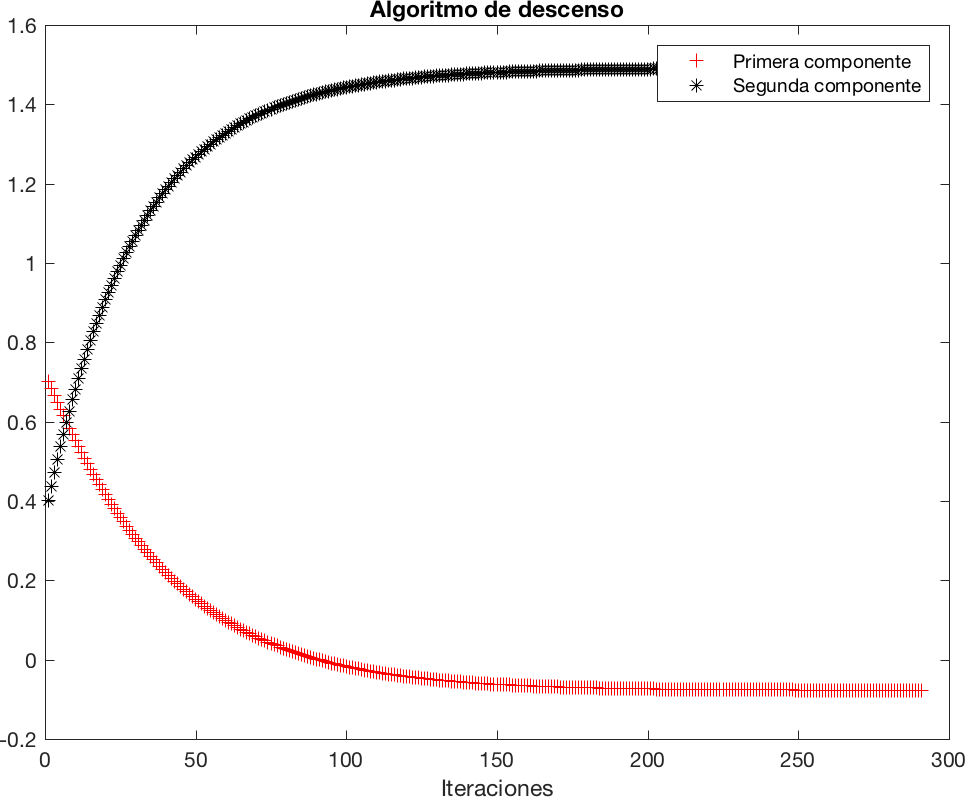


Figura 24: Convergencia del algoritmo de descenso en 292 iteraciones para

la funci´on *f* (**x**) = (1*/*2)*||A***x** *−* **b***||*2 siendo *A* definida positiva. La tasa de

2

aprendizaje o paso de discretizaci´on *ϵ* = 10*−*1.

una **line search** (es decir cuanto me muevo a lo largo de la direcci´on opuesta al gradiente) evaluando

*f* (**x**0 *− ϵ∇f* (**x**0))

para distintos valores de *ϵ* y eligiendo el que m´as minimiza la funci´on objetivo.

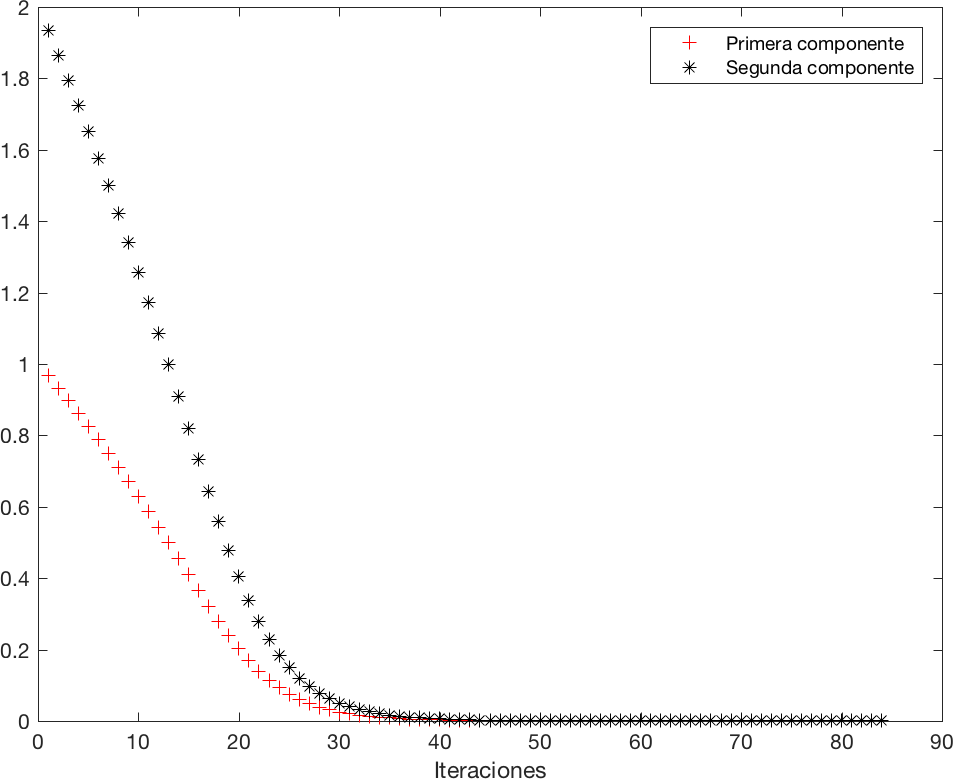


Figura 25: Convergencia del algoritmo de descenso en 85 iteraciones para la funci´on *f* (*x, y*). La tasa de aprendizaje o paso de discretizaci´on *ϵ* = 10*−*1.

**Ejercicio 7.1.** *Escribir un algoritmo de descenso gradiente que calcule el punto de m´ınimo absoluto de la funci´on del ejemplo (*[*7.3*](#_bookmark130)*):*

*f* (*x, y*) = ln *x*2 + *y*2 + 1

*Utilizar una semilla inicial* (*x*0*, y*0)*T* = (1*,* 2)*T , una tolerancia δ* = 1*.e −* 6

*para la norma del gradiente y un ϵ* = 1*e −* 1*.*

##### Matriz Jacobiana y Hessiana

Empezamos considerando la generalizaci´on del concepto de derivada para campos vectoriales **f** : R*m* R*n*. Se tiene **y** = **f** (**x**) R*n*, es decir input y output son ambos vectores. En este caso calculadas todas las posibles deriva- das parciales *∂fi/∂xj* se construye la **matriz jacobiana** *J ∈* R*n,m*, conocida con el nombre de matriz de gradientes en donde cada elemento *Ji,j* = *∂fi/∂xj*.

*→ ∈*

**Ejemplo 7.5.** *Sea dado el campo vectorial* **f** : R3 *→* R2 *definido por*

*f* (*x, y, z*) = (*f*1(*x, y, z*)*, f*2(*x, y, z*)) = (*x*2 *− y* + *z*3*, y*2 + *z*) *∈* R2

*Calculamos la matriz jacobiana J ∈* R2*,*3*. Se tiene*

*J***f** (*x, y, z*) =

(*∇f*1(*x, y, z*))*T*

(*∇f*2(*x, y, z*))*T*

2*x* 1 3*z*2

=

*−*

0 2*y* 1

*y se evalu´a en los puntos en la forma, por ejemplo en el punto* (*x, y, z*) = (1*,* 1*,* 1) *se tiene*

*J***f** (1*,* 1*,* 1) =

(*∇f*1(1*,* 1*,* 1))*T*

(*∇f*2(1*,* 1*,* 1))*T*

= 2 *−*1 3

0 2 1

Puesto que el gradiente de un campo escalar es un campo vectorial para calcular las derivadas parciales segundas de un campo escalar podemos cal- cular el jacobiano del gradiente lo que resulta en la **matriz Hessiana** cuyo gen´erico elemento se define

*∂*2

*H*(*f* )(**x**)*i,j* = *∂x ∂x*

*i*

*j*

*f* (**x**)

**Ejemplo 7.6.** *Sea dado el campo escalar*

*f* (*x, y*) = *e−x*2*−y*2

*Calculamos su gradiente en la forma*

*∇f* (*x, y*) = (*−*2*xe−x*2*−y*2 *, −*2*ye−x*2*−y*2 ) = (*f* (*x, y*)*, f* (*x, y*))

1 2

*y hay un u´nico punto cr´ıtico, en el origen* (*x*0*, y*0) = (0*,* 0) *Calculamos su*

*hessiana aplicando el operador jacobiano*

*J∇***f** (*x, y*) =

( *f*1(*x, y*))*T*

=

*∇*

(*∇f*2(*x, y*))*T*

2*e−x*2*−y*2 + 4*x*2*e−x*2*−y*2 4*xye−x*2*−y*2

*−*

= =

4*xye−x*2*−y*2 *−*2*e−x*2*−y*2 + 4*y*2*e−x*2*−y*2

= *e−x*2*−y*2

*−*2 + 4*x*2 4*xy* 4*xy −*2 + 4*y*2

Evaluando en el origen se tiene

*J∇***f**

(0*,* 0) = *Hf*

(0*,* 0) = *−*2 0

0 *−*2

y la forma cuadr´atica en el origen es definida negativa luego concava. Hay un punto de m´aximo en donde *f* (0*,* 0) = 1.

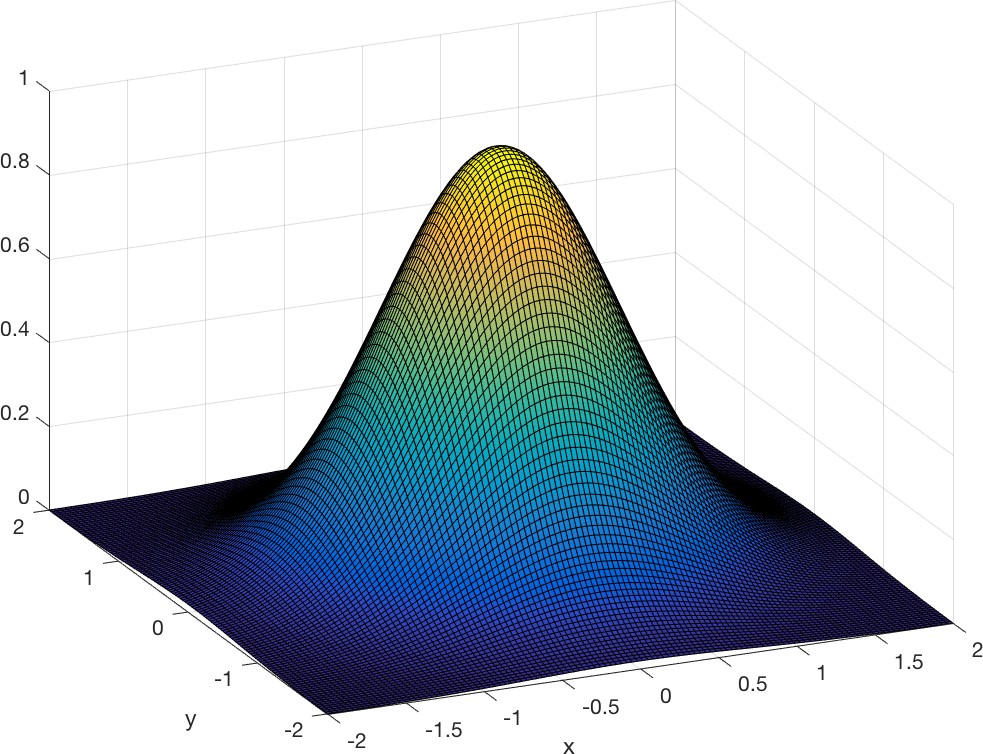


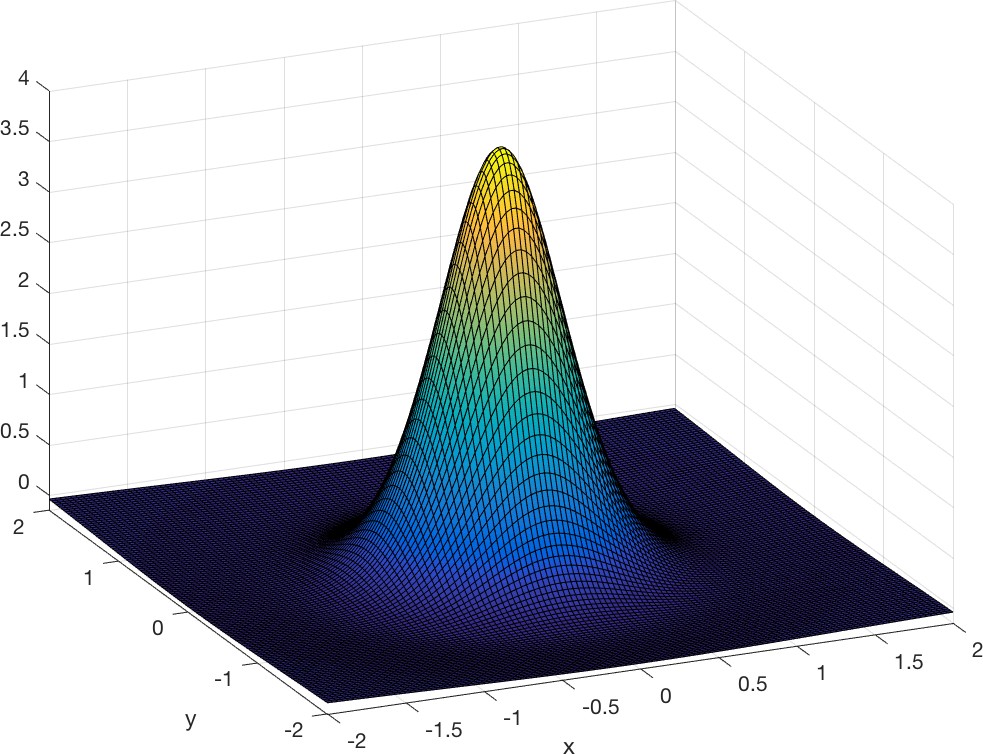
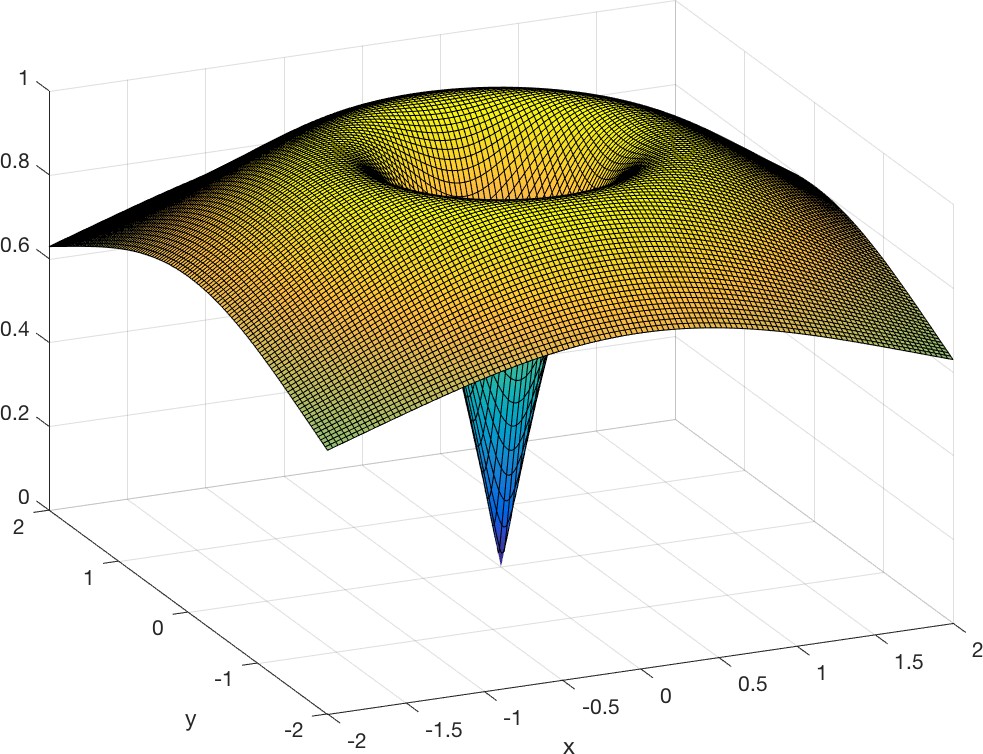
Figura 26: Punto de m´aximo de la funci´on

**Ejemplo 7.7.** *Sea dada la funci´on*

*Se pide:*

*f* (*x, y*) = ln(*x*2 + *y*2 + 1)

* + - 1. *Calcular su gradiente utilizando matlab.*
      2. *Calcular sus derivadas de orden superior y definir su matriz hessiana.*
      3. *Calcular el campo escalar dado por la norma del gradiente.*
      4. *Calcular el campo escalar dado por el determinante de la hessiana.*
      5. *Representar en una u´nica figura y dos gr´afica la norma del gradiente el determinante de la hessiana.*



* + - * 1. norma del gradiente (b) determinante de la hessiana

Figura 27: Gr´aficas de la norma del gradiente (izquierda) y del determinante de la hessiana (derecha)

Un simple c´odigo que permite el c´alculo de los operadores gradiente y hessia- na viene a continuaci´on. Observa que calculamos los operadores gradiente y hessiana en 3 formas distintas: mediante el c´alculo de las derivadas parciales como componentes del vector gradiente, mediante los comandos **gradient.m** y **hessian.m** o mediante la aplicaci´on iterada del operador **jacobian.m** pri- mero al campo escalar (la funci´on) y luego a su gradiente que es un campo vectorial. Finalizamos calculando dos campos escalares asociados a tales ope- radores. La norma del gradiente y el determinante de la hessiana.

%% Parciales de orden superior clear all

close all syms x y

f = x^2-y^2

%%

a=2 figure

fsurf(f,[-a,a,-a,a],’MeshDensity’,80) xlabel(’x’);

ylabel(’y’);

view(-158,40)

print -dpng ./imagenes/clase\_V\_1.png

%% Calculo parciales fx=diff(f); fy=diff(f, y);

fyy=diff(f, y, 2);

fxx=diff(f, x, 2);

fxy=diff(diff(f, y), x);

fyx=diff(diff(f, x), y);

%% definicion campo de gradientes y hessiana G1=[fx,fy];

H1=[fxx, fxy; fyx, fyy];

% operadores gradient y hessian G2=gradient(f,[x,y]);

H2=hessian(f,[x,y])

% operador jacobiano

G3 = jacobian(f, [x, y]); H3 = jacobian(G1, [x, y])

% Calculo norma y determinante norma=norm(G1); determinante=fxx\*fyy-fxy^2;

Si las derivadas parciales son continuas (**Teorema de Schwartz**) entonces las derivadas segunda cruzadas son iguales

*∂*2

*∂xi∂xj*

*f* (**x**) =

*∂*2

*∂xj∂xi*

*f* (**x**)

y la matriz Hessiana es real y **sim´etrica**. Los autovalores son reales y exis- te una descomposici´on en una base de autovectores ortonormales. V´ease la secci´on ([2.2.5](#_bookmark58)). Gracias a la Hessiana podemos tambi´en calcular la derivada segunda direccional de una funci´on en direcci´on del vector unitario **u**:

*H*(**x**0*,* **u**) = **u***T H*(**x**0)**u**

Cuando **u** es un autovector de *H* se obtiene el autovalor asociado:

*H*(**x**0*,* **u**) = **u***T H*(**x**0)**u** = *λ***u***T* **u** = *λ*

El valor m´aximo (m´ınimo) entre todas las derivadas segundas direccionales viene dado por *λ*m´ax (*λ*m´ın).

##### M´etodo del Descenso revisitado

A partir del conocimiento de las derivadas segundas direccionales es posible inferir informaci´on sobre el comportamiento de un algoritmo de descenso. Definimos por **g** = **g**(**x**0) y *H* = *H*(**x**0) al vector gradiente y la matriz Hes- siana de una funci´on escalar de varias variables. Utilizando una aproximaci´on de Taylor cuadr´atica (de II orden) centrada en **x**0 se tiene

*f* (**x**) *≈ f*ˆ(**x***,* **x** ) = *f* (**x** ) + (**x** *− T*

0

0

**x**0)

1

*—* **x** ) *H*(**x** *−* **x** ) (26)

*T*

**g** + (**x** 2

0

0

en donde recordamos que los operadores gradiente y hessiana se evaluan en el punto en donde se centra la aproximaci´on. Consideremos un paso del descenso gradiente ([25](#_bookmark133))

**x** = **x**0 *− ϵ***g** = **x**0 *− ϵ∇f* (**x**0)

con learning rate *ϵ*. Sustituyendo en ([26](#_bookmark136)) la expresi´on del paso de correcci´on

∆**x** = **x** *−* **x**0 = *−ϵ***g**

se tiene

*f* (**x**) *≈ f*ˆ(**x***,* **x** ) = *f* (**x** ) *− ϵ***g***T* **g** + 1 *ϵ*2**g***T H***g** (27)

0

0

2

Hay 3 t´erminos en ([27](#_bookmark137)): el valor de la funci´on, la mejora debida a la pen- diente y la correcci´on debida a la curvatura. Cuando este t´ermino es muy grande el paso de descenso es en realidad de ascenso! Cuando **g***T H***g** 0 la aproximaci´on de Taylor nos dice que hay descenso para cualquier valor de *ϵ >* 0. Sin embargo la aproximaci´on de Taylor es precisa s´olo para 0 *< ϵ* 1 luego hay que acudir a alguna elecci´on heur´ıstica de *ϵ* en este caso. Cuando **g***T H***g** *>* 0 el paso o´ptimo que m´as disminuye la aproximaci´on es

*≤*

*≪*

**g***T* **g**

*ϵ∗* = **g***T H***g**

El caso peor es cuando **g** se alinea con el autovector correspondiente al au- tovalor m´aximo *λ*m´ax en cuyo caso

*ϵ∗* = 1

*λ*m´ax

Si la funci´on que estamos minimizando se aproxima bien por una funci´on cuadr´atica el autovalor m´aximo *λ*m´ax (correspondiente a la derivada segunda direccional m´axima) nos da la escala de la tasa de aprendizaje.

La consideraci´on de la descomposici´on espectral de la Hessiana (es decir el estudio de sus autovalores) permite generalizar el criterio suficiente de las derivadas segundas a mu´ltiples dimensiones.

En un punto cr´ıtico, definido como una soluci´on del sistema ([24](#_bookmark131)) dado por la condici´on necesaria de anulaci´on del gradiente

*∇f* (**x**) = **0**

podemos examinar el espectro de la Hessiana para determinar si el punto es un extremo local o un punto de silla. Si la Hessiana es **definida positiva** el espectro es positivo y el punto es de m´ınimo. Ana´logamente si la Hessiana es **definida negativa** el espectro es negativo y el punto es de m´aximo.

Si hay al menos un autovalor positivo y uno negativo entonces la Hessiana es **indefinida** y hay un punto de silla. Si un autovalor es nulo entonces el criterior de segundo orden no concluye.

En varias dimensiones hay una derivada direccionales segunda diferente en cada punto y en cada direcci´on. Cuanto pueden variar los valores lo estima el nu´mero de condicionamiento de la Hessiana. Un condicionamiento elevado hace que el m´etodo de descenso se comporte de manera poco eficiente y dificulta estimar la tasa de aprendizaje ´optima.

**Ejercicio 7.2.** *Sea dada la funci´on f* (**x**) = *x*2 *y*2*. Calcular sus puntos cr´ıticos y clasificarlos mediante el an´alisis del espectro de la matriz Hessiana.*

*−*

La matriz Hessiana es

 2 0 

*H*(**x**) =  0 *−*2 

con autovalores *λ*1 = 2 *>* 0 y *λ*2 = *−*2 *<* 0 y autovectores

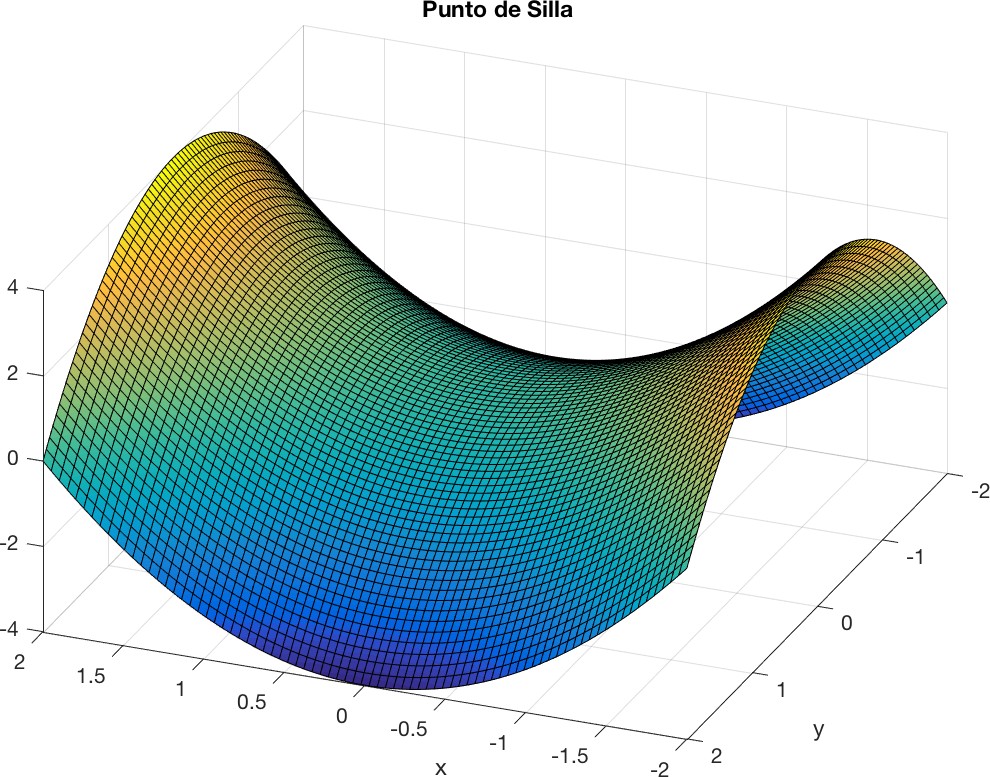


Figura 28: Ana´lisis de la funci´on *f* (**x**) = *x*2 *y*2. Al calcular el espectro de la matriz hessiana se detecta un punto de silla (minmax) con curvatura positiva (convexidad) en la direcci´on del autovector asociado al autovalor positivo y curvatura negativa (concavidad) en la direcci´on del autovector asociado al autovalor positivo.

*−*

 1   0 

   

**u**1 = 0 *,* **u**2 = 1

El primer autovalor avisa de que la curvatura es positiva, la funci´on es con- vexa y hay un m´ınimo a lo largo de la direcci´on dada por el autovector **u**1. El segundo autovalor avisa de que la curvatura es negativa, la funci´on concava y hay un m´aximo a lo largo de la direcci´on dada por el autovector **u**2.

#### C´alculo y Optimizaci´on Multi-variable

##### El M´etodo de Newton para minimizaci´on

Supongamos querer minimizar una energ´ıa (campo escalar)

m´ın *f* (**x**) (28)

**x**

Observa que en este marco muy general se incluyen los problemas de m´ıni- mos cuadrados no lineales a los cuales se puede llegar o porqu´e el sistema no lineal no admite soluci´on o intentando construir un modelo que sea capaz de aproximar y explicar unos datos (Fitting problem). Veremos estos casos en la siguiente clase dedicada a la resoluci´on de sistemas no lineales.

El posible, mal comportamiento del m´etodo de descenso puede ser mejorado utilizando un algoritmo que tenga en cuenta la informaci´on (correcci´on) de la matriz Hessiana. El m´etodo m´as simple para hacerlo es el **m´etodo de Newton** que se basa en la aproximaci´on de Taylor de segundo orden ([26](#_bookmark136)):

*f* (**x**) *≈ f*ˆ(**x***,* **x** ) = *f* (**x** ) + (**x** *−* **x** )*T ∇f*

0

0

0

1

*—* **x** ) *H*(*f* )(**x** )(**x** *−* **x** )

*T*

(**x**0) + 2 (**x**

0

0

0

Calculando el gradiente de la funci´on cuadr´atica aproximante e imponiendo su anulaci´on

se tiene el sistema lineal

*∇f*ˆ(**x***,* **x**0) = **0**

*∇f* (**x**0) + *H*(*f* )(**x**0)(**x** *−* **x**0) = **0** (29)

de donde se deduce la iteraci´on gen´erica del **Algoritmo de Newton**

con paso de correcci´on

**x**(*k*+1) = **x**(*k*) + ∆**x**(*k*) =

= **x**(*k*) *−* [*H*(*f* )(**x***k*)]*−*1*∇f* (**x***k*) (30)

∆**x**(*k*) = *−*[*H*(*f* )(**x***k*)]*−*1*∇f* (**x***k*)

Observamos que en cada iteraci´on hay que resolver el sistema lineal ([29](#_bookmark140)) por lo cual es necesario utilizar m´etodos eficientes en t´erminos de precisi´on y rapid´ez.

**Ejercicio 7.3.** *Se considera el problema de maximizaci´on*

m´ax *f* (**x**) = m´ax *e−*(*x−*1)2*−y*2

*Se pide:*

**x** (*x,y*)

1. *Escribir el problema de optimizaci´on en t´erminos de un problema de minimizaci´on.*
2. *Arrancando con semilla* **x**0 = (0*.*75*,* 0*.*4) *disen˜ar un algoritmo para resolver el problema mediante el algoritmo de Newton definido por la iteraci´on gen´erica (*[*30*](#_bookmark141)*).*

*−*

1. *Arrancar con una semilla aleatoria (random) y analizar la convergen- cia. Repetir el algoritmo varias veces y obtener una conclusi´on sobre el comportamiento del algoritmo para este problema.*
2. *Arrancando con la misma semilla* **x**0 = (0*.*75*,* 0*.*4) *disen˜ar un al- goritmo para resolver el problema mediante el algoritmo de Descenso descrito en la ecuaci´on* ([25](#_bookmark133))*. Utilizar una tasa de aprendizaje ϵ* = 10*−*1*. Comparar la velocidad de convergencia con el algoritmo de Newton.*

*−*

1. *¿Puedes encontrar una tasa de aprendizaje que mejore la velocidad de convergencia del algoritmo de descenso ? Calcular la tasa ´optima.*

Si *f* es una funci´on cuadr´atica asociada a una matriz hessiana definida posi- tiva el problema de m´ınimos cuadrados est´a bien planteado y el algoritmo de Newton converge en un s´olo paso a la soluci´on. Por ejemplo para la funci´on cuadr´atica

*f* (**x**) = (1*/*2)*||A***x** *−* **b***||*2

2

con *A* cuadrada y definida positiva se tiene el gradiente

y la hessiana

*T*

*∇f* (**x**) = *A*

*A***x** *− AT* **b**

*H*(*f*

)(**x**) =

1 (*AT A* + (*AT A*)*T* ) =

2

1 (*AT A* + *AT A*) = *AT A*

2

La soluci´on de la condici´on de optimalidad

*∇f* (**x**) = *AT A***x** *− AT* **b** = **0**

es decir del problema de m´ınimos cuadrados, viene dada por la pseudo-inversa

**x***∗* = *A†***b**

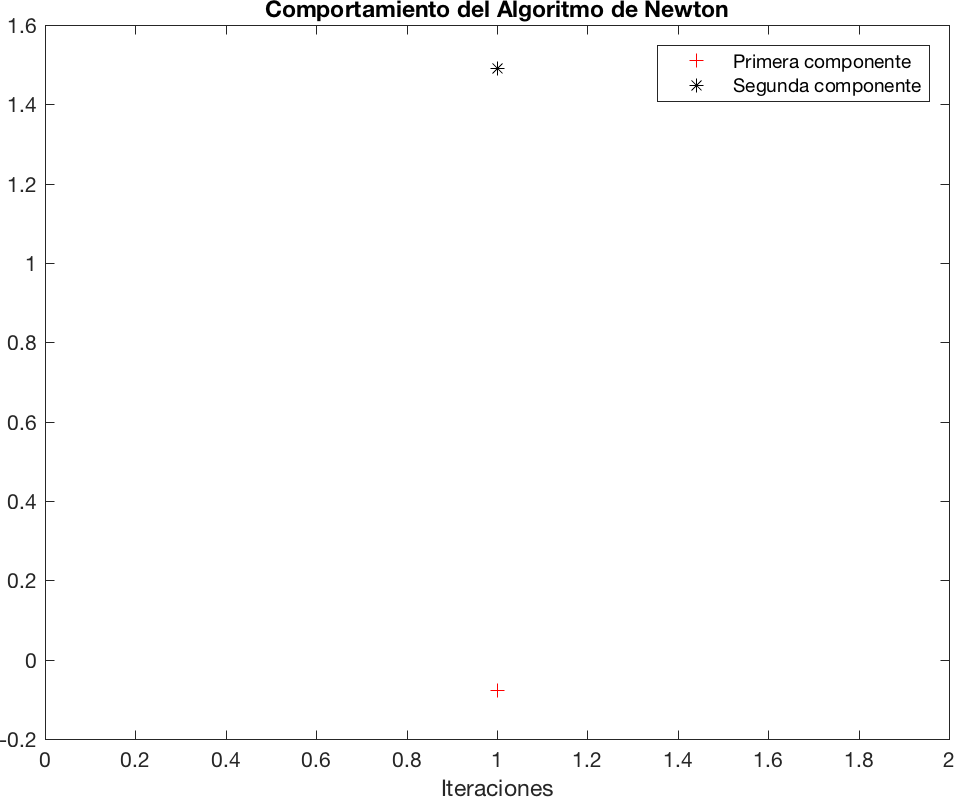


Figura 29: Convergencia en una u´nica iteraci´on del m´etodo de Newton para la funci´on cuadr´atica *f* (**x**) = (1*/*2)*||A***x** *−* **b***||*2.

2

Si *f* no es una funci´on cuadr´atica pero es suficientemente regular para po- derla aproximar (localmente) por una aproximaci´on cuadr´atica el algoritmo de Newton se aplica iterativamente, considerando el gradiente de la forma cuadr´atica aproximante en cada iterada. Si la forma cuadr´atica aproximante es definida positiva el algoritmo de Newton suele converger al m´ınimo mucho m´as r´apidamente que un descenso gradiente.

Los algoritmos que usan s´olo la informaci´on del gradiente (como el **descenso gradiente**) se llaman de **I orden** y los que usan tambi´en la informaci´on de la Hessiana (como el **m´etodo de Newton** y variantes) se llaman de **II orden**. El ´exito en la implementaci´on de uno u otro algoritmo es dependiente del problema de optimizaci´on y de la aplicaci´on.

En el contexto del deep learning se tiene alguna garant´ıa de ´exito (conver- gencia) si consideramos funciones de clase Lipschitz. Definimos para ello una funci´on Lipschitziana como una funci´on cuya tasa de cambio es acotada por una constante *L*:

*|f* (**x**) *− f* (**y**)*| ≤ L||***x** *−* **y***||*2

Esta propiedad es u´til puesto que permite cuantificar la hip´otesis de que pe- quen˜o cambio en el imput dan, en un algoritmo del tipo descenso gradiente, pequen˜o cambio en el output.

El campo de especializaci´on m´as exitoso de la teor´ıa de la optimizaci´on es la **optimizaci´on** convexa en donde la hessiana de la aproximaci´on es semi- definida positiva (funci´on convexa) o definida positiva (funci´on estrictamente convexa). E´stas funciones no tienen punto de silla y sus m´ınimos locales son necesariamente globales. Actualmente la optimizaci´on no convexa (muchos

m´ınimos locales y globales) esta´ avanzando te´oricamente y en las aplicacio-

nes. En deep-learning la optimizaci´on es t´ıpicamente no convexa.

En los ap´endices a estas notas pod´eis encontrar el material necesario para la optimizaci´on convexa y la programaci´on lineal y cuadr´atica.

## X Clase

En la clase anterior, dedicada a la optimizaci´on no lineal multi-variables he- mos utilizado operadores propios del c´alculo diferencial, como el gradiente y la hessiana para deducir dos m´etodos cl´asicos para la minimizaci´on de una energ´ıa dentro del contexto de la optimizaci´on de un campo escalar. En con- creto el m´etodo de Descenso Gradiente y el de Newton.

En esta clase volvemos a considerar el m´etodo de Newton pero para la re- soluci´on de un sistema de ecuaciones no lineales. Nos encontramos en este caso por ejemplo al considerar las condiciones (necesarias) de optimalidad de primer orden dadas por la anulaci´on del gradiente. El m´etodo de Newton sufre tambi´en de serias limitaciones: el sistema tiene que ser cuadrado, se tiene que calcular la inversa de la jacobiana del gradiente (la hessiana) y se necesitan buenas aproximaciones para las semillas iniciales. Sin embargo es un paradigma en resoluci´on num´erica ya que muchos m´etodos que intentan superar estas dificultades se pueden derivar del m´etodo de Newton. En esta clase veremos su aplicaci´on a los problemas de m´ınimos cuadrados no lineales a trav´es de dos variantes muy utilizadas: el m´etodo de Gauss-Newton y el de Levemberg-Marquardt.

#### Sistemas no Lineales: resoluci´on num´erica

Sea dada una funci´on (campo) vectorial **f** : *Df ⊂* R*n →* R*m* siendo

**f** (**x**) = (*f*1(**x**)*, ..., fm*(**x**))*T ∈* R*m*

siendo las componentes *fi*(**x**), *i* = 1*..m* funciones escalares diferenciables que definen la funci´on no lineal. La resoluci´on del sistema de ecuaciones no lineales

**f** (**x**) = **0** *∈* R*m* (31)

es un problema central en c´alculo num´erico y aparece en multitud de apli- caciones y en varias de las asignaturas del master. Como en el caso lineal hablaremos de un problema **sobredeterminado** si *m > n* (m´as ecuaciones que inc´ognitas) e **indeterminado** si *m < n*. Si *m* = *n* el sistema es cuadrado.

##### El M´etodo de Newton para sistemas

Tras deducir el m´etodo de Newton para la minimizaci´on de una funci´on en la ecuaci´on ([30](#_bookmark141)) vamos a deducir su formulaci´on para la resoluci´on de un sistema no lineal de ecuaciones. Para ello se supone la funci´on suficientemente regular, digamos **f** *∈* [*C*(*D***f** )]2, para asegurar la acotaci´on del resto del desarrollo de Taylor. Para cada componente escalar *fj* de **f** podemos considerar su aproximaci´on de Taylor de segundo orden ([26](#_bookmark136)):

*f* (**x**) *≈ f* (**x** )+(**x***−***x** )*T ∇f*

*j*

*j*

0

0

1

*−***x** ) *H*(*f* )(**x** )(**x***−***x** )*, j* = 1*...m*

*T*

*j*(**x**0)+ 2 (**x**

0

*j*

0

0

Sea ∆**x**0 = **x x**0. Suponiendo 0 *<* ∆**x**0 2 1 podemos eliminar los t´erminos cuadr´aticos (pequen˜os) e imponer la anulaci´on de la aproximaci´on lineal obteniendo

*— || || ≪*

*fj*(**x**) *≈ fj*(**x**0) + (**x** *−* **x**0)*T ∇fj*(**x**0) = 0*, j* = 1*...m* es decir, para cada componente

*fj*(**x**0) + ∆**x***T ∇fj*(**x**0) =

0

= *fj*(**x**0) + *∇fj*(**x**0)*T* ∆**x**0 = 0*, j* = 1*...m* luego se verifica, vectorialmente, el sistema lineal

**f** (**x**0) + [*J***f** (**x**0)]∆**x**0 = **0**

del que se obtiene el paso de correcci´on del m´etodo de Newton

∆**x**0 = *−*[*J***f** (**x**0)]*−*1**f** (**x**0)

y la gen´erica iteraci´on es

**x**(*k*+1) = **x**(*k*) + ∆**x**(*k*) = **x**(*k*) *−* [*J***f** (**x**(*k*))]*−*1**f** (**x**(*k*)) (32)

Observa que la matriz jacobiana tiene que ser cuadrada e invertible para que se pueda dar el paso de minimizaci´on de Newton. Es condici´on necesaria para que el algoritmo pueda iterar.

Cuando el problema original es de optimizaci´on (minimizaci´on) de un campo escalar *f* (**x**) de la forma

m´ın *f* (**x**)

**x**

el m´etodo de Newton se utiliza para verificar la condiciones necesarias de optimalidad de primer orden

**f** (**x**) = *∇f* (**x**) = **0**

toma la forma de la iteraci´on expl´ıcita

**x**(*k*+1) = **x**(*k*) *−* [*H*(*f* )(**x**(*k*))]*−*1*∇f* (**x**(*k*)) (33)

Ha sido deducido en la clase anterior, f´ormula ([30](#_bookmark141)).

El m´etodo de newton se puede implementar en matlab usando el algoritmo **newtonsys.m** La dificultad consiste en el c´alculo correcto, anal´ıtico de la matriz Jacobiana. Veamos un ejemplo. Para ello definimos el problema

**Ejemplo 8.1.** *Resolver el sistema no lineal de ecuaciones*

**f** (**x**) = (*f*1(*x, y*)*, f*2(*x, y*))*T* = 



3*x*2 *− y*2 = 0

2 3

3*xy*

*— x −* 1 = 0

Para permitir las evaluaciones de la funci´on vectorial definimos la funci´on

function F=Ffun2(x) F(1,1)=3.\*x(1).^2-x(2).^2;

F(2,1)=3.\*x(1).\*x(2).^2-x(1).^3-1;

return

y calculamos expl´ıcitamente la matriz jacobiana que definimos en la funci´on

function J=Jfun2(x) J(1,1)=6.\*x(1);

J(1,2)=-2.\*x(2); J(2,1)=3.\*x(2).^2-3.\*x(1).^2;

J(2,2)=6.\*x(1).\*x(2);

return

y resolvemos con semilla inicial **x**0 random, tolerancia tol= 1*.e* 5 y un nu´mero m´aximo de 100 iteraciones

*−*

x0=rand(2,1) tol=1e-5 maxiter=100

[x,F,iter]=newtonsys(@Ffun2, @Jfun2, x0, tol, maxiter);

Una alternativa es explotar las posibilidades de c´alculo simb´olico de matlab y calcular la matriz Jacobiana del sistema. Esto es lo que se propone abajo en un c´odigo gen´erico que calcula la matriz jacobiana para sistemas 2 2. La generalizaci´on a dimensiones superiores es directa.

*×*

**Ejemplo 8.2.** *Resolver el sistema no lineal de ecuaciones*

**f** (**x**) = (*f*1(*x, y*)*, f*2(*x, y*))*T* = 



3*x*2 *− y*2 = 0

2 3

3*xy*

*— x −* 1 = 0

La matriz Jacobiana asociada a la funci´on **f** (**x**) es

*J***f** (**x**) = *J***f** (*x, y*) =

(*∇f*1(*x, y*))*T*

(*∇f*2(*x, y*))*T*

= 6*x −*2*y*

3*y*2 *−* 3*x*2 6*xy*

Se puede calcular en simb´olico con Matlab

syms x y f=3.\*x.^2-y.^2;

g=3.\*x.\*y.^2-x.^3-1; F=[f,g];

FN=matlabFunction(F) j=jacobian([f,g],[x,y]);

Observa que durante las iteraciones de Newton es necesario poder invertir la matriz jacobiana que no puede por tanto ser singular. Calculando su deter- minante se tiene

*|J|* = 30*x*2*y* + 6*y*3 = 6*y*(5*x*2 + 1)

e intendemos que no podemos poner semillas en los puntos (*x, y*) = (*x,* 0). Tampoco funcionar´a el algoritmo si alguna de las iteradas es pr´oxima a va- lores del tipo (*x,* 0) ya que en aquella iteraci´on no ser´a invertible la matriz jacobiana. Pasamos a describir el algoritmo de Newton. Empezamos iniciali- zando variables y definiendo semillas:

delt=-j\[f;g]; delta=matlabFunction(delt);

tol=1e-6; nmax=100; niter = 0; err = tol + 1; x0=[1;1]

x0=[1;-1]

x0=rand(2,1) p=x0;

x=p(1)

y=p(2)

El algoritmo es

while err >= tol & niter < nmax p=p+delta(p(1),p(2))

x=p(1); y=p(2);

err = norm(delta(x,y)) res = norm(FN(x,y)) niter = niter + 1

end

Pasamos finalmente a representar las dos ecuaciones del sistema como las curvas de nivel de las funciones componentes

figure axis on

f1=ezplot(f) set(f1,’Color’,’b’) hold on

grid on g1=ezplot(g) set(g1,’Color’,’r’) hold on

plot(p(1),p(2),’k\*’,x0(1),x0(2),’g\*’) legend(’I ecuacion’,’II ecuacion’)

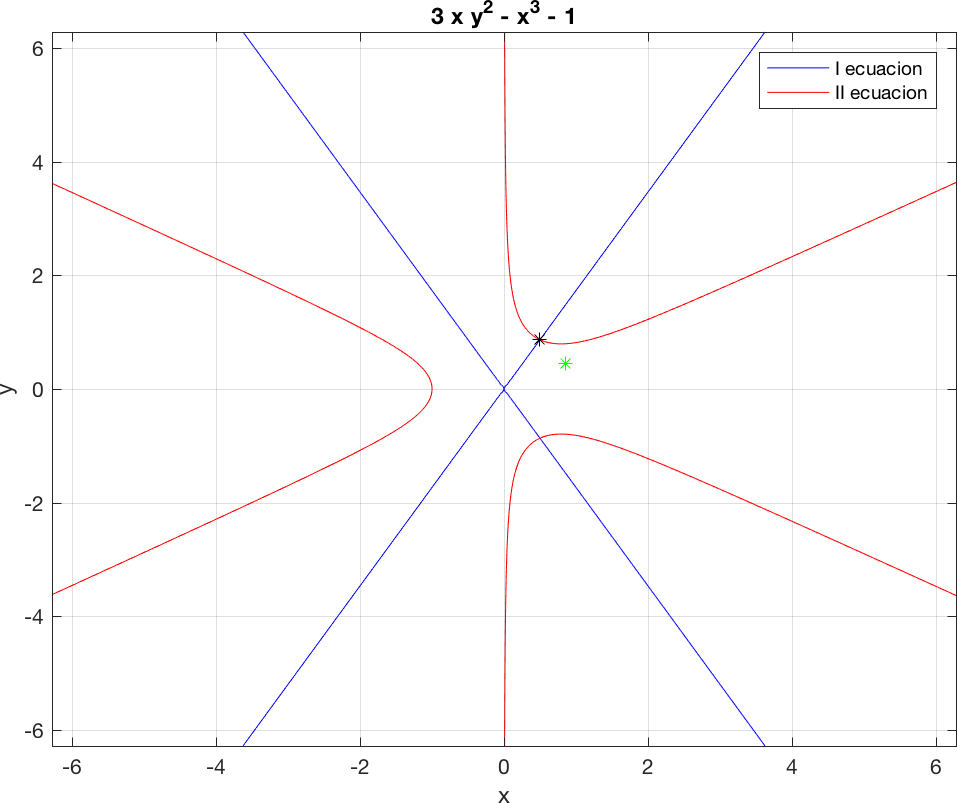


Figura 30: Dos Soluciones del sistema no lineal. Las soluciones vienen dadas por la intersecci´on entre las curvas de nivel *f*1(*x, y*) = 0, *f*2(*x, y*) = 0. En verde la semilla inicial. En negro la soluci´on calculada.

## XI Clase

#### Problemas de M´ınimos Cuadrados No Lineales

Los problemas de m´ınimos cuadrados no lineales consisten en minimizar la diferencia entre un conjunto de datos y una funci´on modelo que los aproxima y explica. El modelo viene dado por una familia param´etrica de funciones y el objetivo es encontrar el par´ametro o´ptimo en el sentido de los m´ınimos cuadrados no lineales.

**Ejemplo 9.1.** *Se quiere calcular un modelo para aproximar (fitting) unos datos. Los datos, ruidosos, se generan a partir de un modelo perturbado de decaimiento exponencial*

*y* = *e−αt* + *n, t ∈* [0*,* 3] (34)

*siendo α* = 1*.*3 *la tasa de decaimiento y n un ruido con distribuci´on normal de media cero y desviaci´on estandar σ* = 0*.*05*. El problema es: a partir de los datos* (*α, y*)*, encontrar la tasa de decaimiento exponencial α∗ que mejor apro- xima los datos en el sentido del problema de m´ınimos cuadrados no lineales. La soluci´on ´optima es:*

*α∗* = arg m´ın *||f* (*α*)*||*2 = 1*.*2645

*α*

2

Mediante el siguiente c´odigo, disponible en la p´agina de informaci´on de matlab para el comando **lsqnonlin.m** se resuelve el problema. Primero ge- neramos los datos utilizando el modelo generativo ([34](#_bookmark147)):

rng default % for reproducibility t = linspace(0,3);

y = exp(-1.3\*t) + 0.05\*randn(size(t));

luego definimos la funci´on input del algoritmo **lsqnonlin** que toma en input un valor de la tasa de decaimiento y da en output un vector de diferencias entre lo que predice el modelo *e−αt* y los datos ruidosos, es decir los residuos:

*f* (*α*) = *e−αt − y*

fun = @(r)exp(-t\*r)-y; alpha\_0 = 4;

alpha\_star = lsqnonlin(fun,alpha\_0)

Finalmente representamos la curva exponencial que aproxima los datos de forma ´optima.

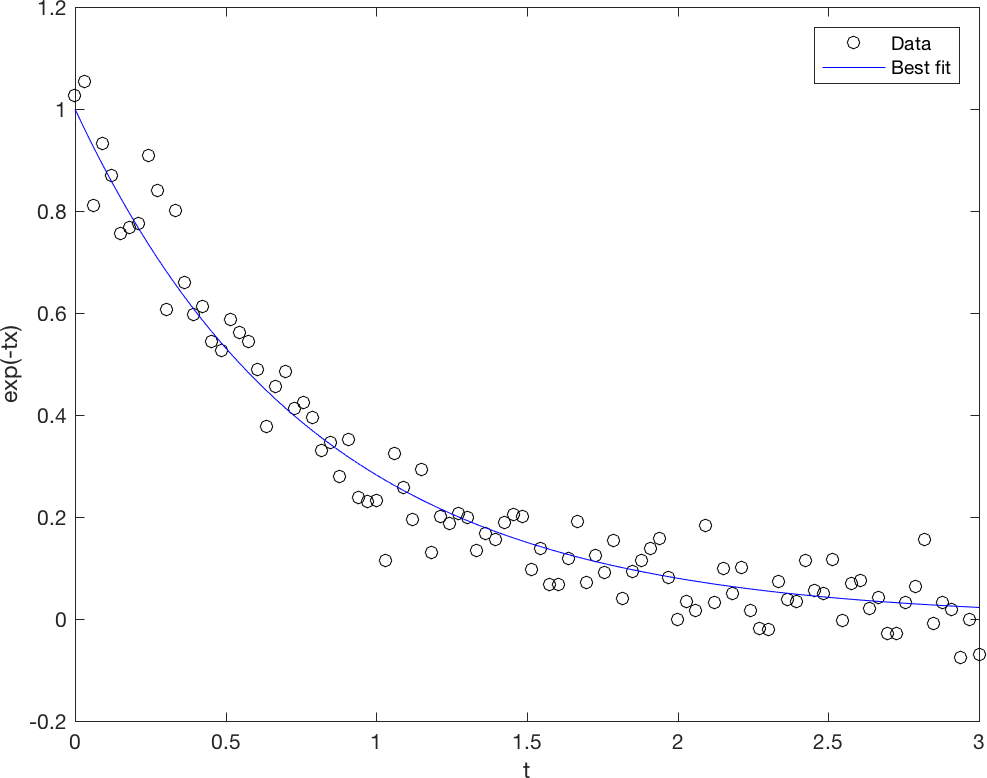


Figura 31: Data fitting (regresi´on).

figure

plot(t,y,’ko’,t,exp(-alpha\_star\*t),’b-’) legend(’Data’,’Best fit’)

xlabel(’t’) ylabel(’exp(-t\*alpha)’)

Tambi´en nacen al intentar dar una soluci´on de m´ınimos cuadrados a siste- mas de ecuaciones que no la tienen. Sea dada una funci´on (campo) vectorial **f** : R*n →* R*m* con **f** = (*f*1(**x**)*, ..., fm*(**x**)), *fi*(**x**), *i* = 1*..m* funciones diferencia- bles que definen un problema no lineal. T´ıpicamente las funciones *fi* son los **residuos** entre los datos y la funci´on modelo.

#### El problema de m´ınimos cuadrados no lineales

El problema de m´ınimos cuadrados no lineales consiste en la minimizaci´on de la energ´ıa cuadr´atica (campo escalar) *g*(**x**)

m´ın *g*(**x**) = m´ın

1

*||***f** (**x**)*||* = m´ın

2

1 *m*

*fi*(**x**)2

(35)

**x x** 2

Si

1. **x** 2

*i*=1

**f** (**x**) = *A***x** *−* **b**

tenemos un problema de m´ınimos cuadrados lineal. Para la resoluci´on del problema, en general no lineal [35](#_bookmark148) consideramos la energ´ıa del problema

*g*(**x**) = 1 *||***f** (**x**)*||*2 = 1 *f* (**x**)2

*m*

2

2

2

*i*

*i*=1

que es un campo escalar *g* : R*n →* R. Calculamos las derivadas parciales

*∂g* (**x**) = *f* (**x**) *∂fi* (**x**)*, j* = 1*...n*

*j*

*i*=1

*j*

*m*

*∂x*

*i*

*∂x*

por lo cual el gradiente en un punto ser´a

*m*

*∇g*(**x**) = *fi*(**x**)*∇fi*(**x**) = *J***f** (**x**)*T* **f** (**x**) (36)

*i*=1

siendo *J***f** (**x**) la matriz jacobiana. La condici´on necesaria (en general no sufi- ciente) de optimalidad es

*∇g*(**x**) = *J***f** (**x**)

*T*

**f** (**x**) = **0** (37)

Los elementos de la matriz hessiana *H*(*g*)(**x**), denotada tambi´en por 2*g*(**x**) vienen dado por todas las derivadas parciales segundas

*∇*

2 *m m* 2

*∂x*

*∂x*

*i*

*∂x*

*∂x*

*H*(*g*)(**x**)

*∂ g*(**x**)

=

*k,l*

*∂x*

*∂x*

*k*

*l*

= *∂fi*(**x**) *∂fi*(**x**) + *f* (**x**) *∂ fi*(**x**)

*i*=1

*k*

*l*

*i*=1

*k*

*l*

y la hessiana se puede escribir como

*m*

*∇*2*g*(**x**) = *J***f** (**x**)*T J***f** (**x**) + *fi*(**x**)*∇*2*fi*(**x**) (38)

*i*=1

El algoritmo de Newton para m´ınimos cuadrados no lineales toma la forma

**x**(*k*+1) = **x**(*k*) *−* [*∇*2*g*(**x**(*k*))]*−*1*∇g*(**x**(*k*)) (39)

La matriz hessiana tiene que ser definida positiva para que los problemas de m´ınimos cuadrados tengan una u´nica soluci´on. La base de los algoritmos de Gauss-Newton y Levemberg-Marquard est´a en la aproximaci´on de la hessiana

*∇ g*(**x**) *≈ J***f** (**x**) *J***f** (**x**) (40)

2 *T*

que es efectiva cuando los residuales son pequen˜os, es decir *fi*(**x**) 0 y estamos cerca de la soluci´on.

*≈*

##### El M´etodo de Descenso Gradiente para m´ınimos cuadrados

Los M´etodos de Descenso son esquemas iterativos del tipo

**x**(*k*+1) = **x**(*k*) + *α*(*k*)∆**x**(*k*)

en donde ∆**x**(*k*) R*n* es la **direcci´on del paso** de correcci´on y *α*(*k*) R es la medida o tamano del paso. Si ∆**x**(*k*) = *f* (**x**(*k*)) tenemos un m´etodo de descenso gradiente:

*−∇*

*∈ ∈*

**x**(*k*+1) = **x**(*k*) *− α*(*k*)*∇f* (**x**(*k*))

Si queremos aplicar un esquema descenso gradiente al problema de minimos cuadrados ([35](#_bookmark148)) definimos

*f* (**x**) = *g*(**x**) = 1 *||***f** (**x**)*||*2

2

2

siendo **f** (**x**) el campo vectorial de residuos y utilizamos la f´ormula del gra- diente ([36](#_bookmark149)) obteniendo la iteraci´on expl´ıcita

**x**(*k*+1) = **x**(*k*) *− α*(*k*)*J***f** (**x**)*T* **f** (**x**) (41)

##### El M´etodo de Gauss-Newton

Sea **x**0 una estimaci´on inicial de la soluci´on buscada, es decir una semilla del m´etodo. El primer paso del **M´etodo de Gauss-Newton** consiste en considerar una linealizaci´on ˆ**f** centrada en el punto **x**0 de la funci´on no lineal **f** que viene dada por una aproximaci´on de Taylor de primer orden:

**f** (**x**) *≈* ˆ**f** (**x***,* **x**0) = **f** (**x**0) + [*J***f** (**x**0)](**x** *−* **x**0)

siendo [*J***f** (**x**0)] la matriz jacobiana de **f** evaluada en **x**0. Se considera entonces el problema de m´ınimos cuadrados lineal

m´ın *||*ˆ**f** (**x***,* **x**0)*||*2 = m´ın *||***f** (**x**0) + [*J***f** (**x**0)](**x** *−* **x**0)*||*2

**x** 2 **x** 2

cuya soluci´on

**x**1 = arg m´ın *||***f** (**x**0) + [*J***f** (**x**0)](**x** *−* **x**0)*||*2

**x**

2

nos da la nueva iterada. Observa que en cada iteraci´on del algoritmo se resuel- ve un problema de m´ınimos cuadrados lineal. Si las columnas de [*J***f** (**x**0)]*m,n* son linealmente independientes (es decir rango *n*) la soluci´on exacta (a menos de errores de redondeo) viene dada por:

**x** = **x** *−* [*J* (**x** )]*T* [*J* (**x** )] *−*1 [*J* (**x** )]*T* **f** (**x** )

1

0

**f**

0

**f**

0

**f**

0

0

es decir la pseudo-inversa de la matriz Jacobiana (v´ease el teorema [3.4](#_bookmark65))

[*J* (**x** )]*†* = [*J* (**x** )]*T* [*J* (**x** )] *−*1 [*J* (**x** )]*T*

**f**

0

**f**

0

**f**

0

**f**

0

El paso de correcci´on de Gauss-Newton ∆**x**0 = **x**1 *−* **x**0 es por tanto:

∆**x**0 = *−*[*J***f** (**x**0)]*†***f** (**x**0) =

= *−* [*J* (**x** )]*T* [*J* (**x** )] *−*1 [*J* (**x** )]*T* **f** (**x** ) =

**f**

0

**f**

0

**f**

0

0

= *−* [*J* (**x** )]*T* [*J* (**x** )] *−*1 *∇g*(**x** ) (42)

**f**

0

**f**

0

0

con iteraci´on

**x**1 = **x**0 + ∆**x**0

El paso de iteraci´on gen´erico del algoritmo de Gauss-Newton es

**x***k*+1

= **x***k*

*—* [*J***f** (**x***k*

)]*T* [*J***f** (**x***k*

)] *−*1 [*J* (**x**

)]*T* **f** (**x***k*

) (43)

y en t´erminos de la pseudo-inversa se puede escribir en la forma

**f**

*k*

**x***k*+1 = **x***k −* [*J***f** (**x***k*)]*†***f** (**x***k*)

Comparando con la f´ormula ([38](#_bookmark150)) observa que no se est´a considerando e in- virtiendo la matriz hessiana sino la aproximaci´on ([40](#_bookmark151)).

**Ejemplo 9.2.** *Sea dado el sistema no lineal de ecuaciones*

**f** (**x**) = (*f*1(*x, y*)*, f*2(*x, y*)*, f*3(*x, y*))*T* = **0**

*definido por las ecuaciones*



*x −* 0*.*4 = 0

*y −* 0*.*8 = 0



*x*2 + *y*2 *−* 1 = 0

*Resolver el sistema en el sentido de los m´ınimos cuadrados.*

El sistema es sobre-determinado (riesgo de no existencia..) y es inmediato ver que el sistema no tiene soluci´on ya que (0*.*4)2 + (0*.*8)2 = 1. Buscamos entonces una soluci´on de m´ınimos cuadrados. La matriz Jacobiana asociada es

*̸*

 (*∇f*1(*x, y*))*T* 



 1 0 

*J***f** (**x**) = *J***f** (*x, y*) = (*∇f*2(*x, y*))*T*

(*∇f*3(*x, y*))*T*

 = 

0 1

2*x* 2*y*

3*,*2

y rg(*J*) = 2 = *n* luego tienen columnas linealmente independientes y es de rango m´aximo. Podemos aplicar la iteraci´on de Gauss-Newton ([43](#_bookmark154))

**Ejercicio 9.1.** *Escribir un c´odigo que resuelva el sistema anterior mediante el M´etodo de Gauss-Newton calculando de manera simb´olica la matriz jaco- biana del sistema.*

##### El M´etodo de Levenberg-Marquard

El M´etodo de Levenberg-Marquard resuelve dos problemas fundamentales del M´etodo de Gauss-Newton:

* + - 1. ¿qu´e hacer cuando las columnas de la jacobiana no son linealmente independientes ?
      2. ¿qu´e hacer cuando el paso de Gauss-Newton no reduce *||***f** (**x**)*||*2 segu´n

2

avanzan las iteraciones ?

El M´etodo de Levenberg-Marquard calcula la nueva aproximaci´on resolviendo un **problema de m´ınimos cuadrados regularizado**:

m´ın *||*ˆ**f** (**x***,* **x**0)*||*2 + *λ*0*||***x** *−* **x**0*||*2 =

**x**

2

2

= m´ın *||***f** (**x**0) + [*J***f** (**x**0)](**x** *−* **x**0)*||*2 + *λ*0*||***x** *−* **x**0*||*2

**x**

2

2

Iterando tenemos una secuencia de par´ametros de regularizaci´on *λk*. Si *λk >* 0 el problema admite una u´nica soluci´on y la secuencia **x***k* est´a bien definida. No hay condici´on sobre la jacobiana. La soluci´on viene dada por

**x** = **x** *−* [*J* (**x** )]*T* [*J* (**x** )] + *λ I −*1 [*J* (**x** )]*T* **f** (**x** ) (44)

*k*+1

*k*

**f**

0

**f**

0

*k*

**f**

0

*k*

El paso de correcci´on de Levenberg-Marquard ∆**x***k* = **x***k*+1 *−* **x***k* es por tanto:

∆**x** = *−* [*J* (**x** )]*T* [*J* (**x** )] + *λ I −*1 [*J* (**x** )]*T* **f** (**x** ) =

*k*

**f**

0

**f**

0

*k*

**f**

0

*k*

= *−* [*J* (**x** )]*T* [*J* (**x** )] + *λ I −*1 *∇g*(**x** )

**f**

0

**f**

0

*k*

*k*

Si *λk* = 0 se tiene el m´etodo de Gauss-Newton. Hay distintas estrategias para estimar el par´ametro. Por ejemplo, fijado *λk >* 0, en cada iteraci´on se calcula la soluci´on

**x**ˆ = arg m´ın *||*ˆ**f** (**x***,* **x***k*)*||*2 + *λk||***x** *−* **x***k||*2 =

**x**

2

2

= arg m´ın *||***f** (**x***k*) + [*J***f** (**x**0)](**x** *−* **x***k*)*||*2 + *λk||***x** *−* **x***k||*2

**x** 2 2

Si

*g*(**x**ˆ) = *||***f** (**x**ˆ)*||*2 *< g*(**x***k*) = *||***f** (**x***k*)*||*2

2

2

entonces **x***k*+1 = **x**ˆ y se disminuye *λk*. En otro caso no se actualiza **x***k* (**x***k*+1 =

**x***k*) y se aumenta *λk*.

En matlab los problemas de m´ınimos cuadrados no lineales se pueden resolver mediante el comando **lsqnonlin.m** que permite implementar el m´etodo de Levemberg-Marquardt entre otros. Vamos a desarrollar algu´n ejemplo que se propone en la documentaci´on online de matlab. La sint´axis general es

x = lsqnonlin(fun,x0,lb,ub,options)

en donde es posible introducir restricciones del tipo lower bound (lb) y upper bound (ub). Para seleccionar el algoritmo de Levemberg-Marquardt se utiliza:

options.Algorithm = ’levenberg-marquardt’; x\_lm= lsqnonlin(fun,x0,[],[],options);

Finalmente, consideramos un ejemplo en donde se analiza la diferencia entre utilizar o no la expresi´on anal´ıtica del jacobiano frente a su aproximaci´on num´erica mediante diferencias finitas.

**Ejemplo 9.3.** *Resolver el problema de m´ınimo cuadrado no lineal (*[*35*](#_bookmark148)*)*

m´ın *g*(**x**) = m´ın *||***f** (**x**)*||*2 = m´ın

*m*

*fi*(**x**)2

**x x** 2

**x**

*i*=1

*definido por (para i* = *k, m* = 10*,* **x** = (*x*1*, x*2)*)*

10

*− −*

m´ın 2 + 2*k ekx*1 *ekx*2 2

**x**

*k*=1

*utilizando una semilla inicial* **x**0 = (0*.*3*,* 0*.*4)*.*

Para ello consideramos las componentes escalares

*fk*(**x**) = 2 + 2*k − ekx*1 *− ekx*2 *, k* = 1*..*10

y escribimos una funci´on, llamada **mls.m** que calcule el valor de las compo- nentes de la funci´on objetivo

function F = mls(x) k = 1:10;

F = 2 + 2\*k-exp(k\*x(1))-exp(k\*x(2));

Resolvemos por defecto mediante

x0 = [0.3,0.4];

[x,resnorm,res,eflag,output1] = lsqnonlin(@mls,x0);

Puesto que la funci´on **mls.m** no calcula el jacobiano de la funci´on vectorial

*k*=1

**f** (**x**) = (*fk*(**x**))10

*k*=1

= (2 + 2*k − ekx*1 *− ekx*2 )10

el algoritmo utilizado para la resoluci´on (Trust-region reflective) utiliza f´ormu- las en diferencias finitas para su aproximaci´on. Este algoritmo no funciona para sistemas indeterminados. Si queremos an˜adir la informaci´on del Jaco- biano calculamos

*Jk,j*

(**x**) = *∂fk* (**x**)

*∂x*

es decir

*j*

*Jk,*1(**x**) = *−kekx*1 *, Jk,*2(**x**) = *−kekx*2

Escribimos entonces la funci´on **lmsjac.m** que incluye el Jacobiano

function [F,J] = mlsjac(x) k = 1:10;

F = 2 + 2\*k-exp(k\*x(1))-exp(k\*x(2)); if nargout > 1

J = zeros(10,2);

J(k,1) = -k.\*exp(k\*x(1));

J(k,2) = -k.\*exp(k\*x(2));

end

y definimos las opciones que permiten utilizar el jacobiano

opts = optimoptions(@lsqnonlin,’SpecifyObjectiveGradient’,true);

resolviendo mediante

x0 = [0.3 0.4];

[x,resnorm,res,eflag,output2] = lsqnonlin(@mlsjac,x0,[],[],opts); Comparamos los resultados mediante [output1.funcCount,output2.funcCount]

lo que nos permite ver que el uso del jacobiano permite alcanzar la soluci´on con muchas menos evaluaciones de la funci´on.

## XII Clase

En esta clase se imparten los conceptos fundamentales del C´alculo Variacio- nal. Utilizaremos este marco te´orico para resolver problemas de procesado de imagen de bajo-nivel, low-level[11](#_bookmark159), en Computer Vision. Trabajaremos en concreto problemas de filtrado, lineal y no lineal en el espacio de la imagen, eliminaci´on de ruido, reconstrucci´on e impainting. A partir de modelos de tipo Bayesiano, se introducen distintos a priori, priors, que regularizan el problema en el sentido de Tikhonov generalizado. Las ecuaciones se resuel- ven mediante m´etodos num´ericos de discretizaci´on del tipo diferencias finitas. Este material sirve de base para la resoluci´on de las pr´acticas dedicadas a este tema que finalizan la primera parte del curso de Fundamentos.

#### C´alculo Variacional

El C´alculo Variacional se basa en el an´alisis de las variaciones frente a per- turbaciones de un funcional de energ´ıa que queremos minimizar. Esto nos lleva al c´alculo del gradiente del funcional considerado lo que se plasma en la derivaci´on de las Ecuaciones de Euler-Lagrange que representan condiciones necesarias de primer orden para la optimalidad. En esta clase nos concentra- remos en la definici´on y c´alculo de los operadores fundamentales para resolver num´ericamente el problema revisando las distintas estrategias que se han ido desarrollando durante los u´ltimos an˜os. La base te´orica del calculo variacio- nal es el an´alisis funcional cuyo conocimiento excede los objetivos del curso por su dificultad. Sin embargo unos conocimientos m´ınimos son necesarios para entender los formalismos y formalizaciones que consideraremos lo que nos permitir´a entender los algoritmos de resoluci´on propuestos.

##### Funcionales de Energ´ıa

Empezamos definiendo los funcionales de energ´ıa y su dominio, es decir donde estan bien definidos. Los funcionales de energ´ıa son funciones de funciones definidos en un espacio infinito dimensional. Estamos pasando del marco discreto al continuo. En el contexto de los espacios infinito dimensionales la operaci´on de producto escalar se generaliza a la forma integral

11La clasificaci´on de los problemas de procesado de imagen de bajo-nivel re refiere a que son pre-procesos que sirven para resolver mejor problemas avanzados t´ıpicos de la Visi´on Artificial como el de la comprensi´on autom´atica de la sem´antica de im´agenes y videos.

*⟨f, g⟩* = *fgdx*

Ω

en donde el s´ımbolo de sumatorio discreto se ha sustituido por el de integral, continuo. Podemos as´ı calcular la norma inducida por el producto escalar en la forma

*⟨f, f⟩* = *||f||*2 = *|f| dx*

Ω

2

2

M´as en general podemos definir una norma *p* en la forma

( *p*

1*/p*

*||f||p* = *|f| dx*

Ω

*≤ ≤ ∞ ∈ || || ∞*

Para 1 *p* diremos que *f Lp*(Ω) si *f p <* . Los espacios *Lp*(Ω) se conocen con el nombre de **espacios de Lebesgue**. Si *p* = hablamos del espacio de funciones esencialmente[12](#_bookmark160) acotadas, es decir, *L∞*(Ω). Se tienen las siguientes inclusiones

*L∞*(Ω) *⊂ .... ⊂ Lp*(Ω) *⊂ ... ⊂ L*1(Ω) *⊂ M*(Ω)

*∞*

siendo (Ω) el espacio de **medidas acotadas de Rado´n**. S´olo observamos que estas medidas no son funciones sino distribuciones. Observa que en la cadena de inclusiones el espacio m´as pequen˜o es *L∞*(Ω) ya que si una funci´on es acotada entonces es integrables a cualquier potencia *p* luego pertenece a todos los *Lp*(Ω), *p ≥* 1.

*M*

Si queremos m´as control sobre el comportamiento de las funciones podemos definir, para valores de *p >* 1 los **espacios de Sobolev**

*W* 1*,p . p p n*

(Ω) = *{v ∈ L* (Ω) *| ∇v ∈* [*L* (Ω)] *}*

Para *p* = 1 se define el espacio de funciones de variaci´on acotada, **Bounded Variation**

12Por esencialmente acotadas se entienden las funciones acotadas en casi todo punto de Ω. Los puntos en donde la funci´on podr´ıa no ser acotadas forman un conjunto de medida cero (del tipo una recta en un plano) en el sentido de la medida de Lebesgue. Se trata de una teor´ıa de la medida que extiende la teor´ıa de la integraci´on de Riemann. Se utiliza en el contexto de las formulaciones d´ebiles de la EDP. Las formulaciones d´ebiles permiten dar sentido a las soluciones que aparecen en procesamiento de im´agenes

*BV .* 1 *n*

(Ω) = *{v ∈ L* (Ω) *| ∇v ∈* [*M*(Ω)] *}*

Se puede considerar que el espacio *BV* (Ω) es el adecuado para describir las im´agenes naturales ya que las funciones de *BV* (Ω) pueden ser discontinuas mientras que las funciones de *W* 1*,p*(Ω) son continuas para todo *p ≥* 1.

##### Modelos lineales: eliminaci´on de ruido y reconstrucci´on

Sea *K* un operador lineal. A partir de un modelo de eliminaci´on de ruido (*K ≡ I*) y reconstrucci´on (*K ̸*= *I*) de una imagen del tipo aditivo y gaussiano

*f* = *Ku* + *n* (45)

el funcional *E*2(*u*) que define la energ´ıa del problema a minimizar

*E*2(*u*) = *G*2(*u*) + *λF* (*u*) = (46)

= *||∇u||*2 + *λ||Ku − f||*2 =

2 2

= *|∇u|*2*dx* + *λ* (*Ku − f* )2*dx*

Ω

Ω

se compone de dos t´erminos que modelan la **fidelidad de datos** *F* (*u*) y la

**regularizaci´on** *G*2(*u*). El operador lineal

*K* : *L*2(Ω) *→ L*2(Ω)

se puede intender como un filtro que modela varios procesos de degradaci´on sufridos por la imagen durante su adquisici´on. Por ejemplo *Ku* puede modelar el efecto de un emborronamiento de la imagen o una p´erdida de resoluci´on (downsampling) debida a una adquisici´on r´apida. Si *K I* tenemos un modelo de eliminaci´on de ruido (denoising).

*≡*

##### El dominio del funcional

Dada *f L*2(Ω) el t´ermino *F* (*u*) est´a bien definido en el espacio de las funciones de cuadrado integrables, es decir su norma al cuadrado es acotada:

*∈*

*F* (*u*) =

Ω

(*Ku − f* )2*dx* = *||Ku − f||*2 *< ∞, ∀ u ∈ L*2(Ω)

Si adem´as *∇u ∈* [*L*2(Ω)]*n* es decir

2

*G*2(*u*) =

Ω

*|∇u| dx* = *||∇u||*2 *< ∞, ∀ u ∈ L* (Ω) *| ∇u ∈* [*L* (Ω)]

diremos que el funcional de energ´ıa

2 2 2 2 *n*

*E*2(*u*) = *G*2(*u*) + *λF* (*u*) =

= *|∇u|*2*dx* + *λ* (*Ku − f* )2*dx* =

Ω Ω

= *||∇u||*2 + *λ||Ku − f||*2 *< ∞*

2 2

est´a bien definido en el espacio de Sobolev

*W* 1*,*2 *.* 2 2 *n*

(Ω) = *{v ∈ L* (Ω) *| ∇v ∈* [*L* (Ω)] *}*

ya que la energ´ıa es acotada, *E*2(*v*) *<* , independientemente de la funci´on *v W* 1*,p*(Ω) que se considere. Este funcional es adem´as continuo en el espacio ya que es una suma de normas y las normas son siempre funcionales continuos. Observa que en muchos textos de an´alisis funcional y art´ıculos se denota *W* 1*,*2(Ω) = *H*1(Ω) para enfatizar que el espacio de Sobolev tiene estructura de espacio de Hilbert y es posible definir un producto escalar.

*∈*

*∞*

##### La Regularizacio´n de Tikhonov. Generalizaci´on

La Regularizaci´on de Tikhonov, en su forma original, consiste en an˜adir un t´ermino de regularizaci´on/penalizacion para las soluciones de un problema de minimizaci´on mal planteado. Por ejemplo, considerando el funcional ([46](#_bookmark162)) vemos que el problema de m´ınimo cuadrados asociado

m´ın *F* (*u*) = m´ın (*Ku − f* )2*dx* = m´ın *||Ku − f||*2

Ω

*≡*

*u*

*u*

*u*

2

podr´ıa estar mal planteado (ill-posed) por ser un problema indeterminado y mal condicionado, dependiendo de las propiedades del operador *K*. Para *K* = *Id* que define el problema de eliminaci´on de ruido se tendr´ıa la solu- ci´on *u f* lo que equivaldr´ıa a quedarnos con los datos (imagen) ruidoso. Matem´aticamente hemos visto en clases anteriores (y en el caso discreto) que podemos regularizar el problema de m´ınimos cuadrados desplazando el espectro del operador *K* mediante consideraci´on (penalizaci´on) de la energ´ıa de la soluci´on. En el caso continuo esto se escribe en la forma

m´ın *Fλ*(*u*) = m´ın

*u*

*u*

(*Ku − f* )2*dx* + *λ*

*u*2*dx* =

Ω

Ω

= m´ın *||Ku − f||*2 + *λ||u||*2

*u*

2

2

siendo *λ >* 0 un par´ametro a determinar. Si bien el problema obtenido est´a ahora bien planteado matem´aticamente desde el punto de vista pr´actico de las aplicaciones vemos que favorecemos soluciones de baja energ´ıa pero sin controlar sus oscilaciones de alta frecuencia t´ıpicamente asociadas a bordes y ruido. Para remediar este problema la t´ecnica de regularizaci´on de Tikhonov consiste en penalizar las oscilaciones excesivas en la imagen mediante penali- zaci´on de la energ´ıa del m´odulo del gradiente de la soluci´on. La introducci´on de este t´ermino hace que las ecuaciones de optimalidad de primer orden de anulaci´on del gradiente del funcional, denotado por *E*2(*u*) sean ecuaciones diferenciales en derivadas parciales de tipo el´ıptico, estacionario, Lineales. La linealidad de estas ecuaciones, conocidas como **Ecuaciones de Euler- Lagrange** del funcional hace que las soluciones sean m´as regulares, digamos una versi´on suavizada (smooth) de los datos. Si bien este proceso lineal se utiliza muy a menudo para la obtenci´on de estad´ısticas m´as robustas sobre los datos, se puede demostrar, utilizando la teor´ıa de la regularidad de las EDP el´ıpticas, que las soluciones del problema se ven ahora afectadas por el problema de la excesiva regularizaci´on lo que se conoce como **image overs- moothing**. Una alternativa consiste en generalizar la t´ecnica de Tikhonov

*∇*

introduciendo **a priori**, es decir regularizantes, que no causen esta excesiva regularizaci´on preservando una propiedad fundamental de las im´agenes na- turales que es su **discontinuidad**. Matem´aticamente esto se traduce en la introducci´on de operadores de regularizaci´on No Lineales. M´as en general y en este marco podemos definir energ´ıas del tipo

*Ep*(*u*) = *Gp*(*u*) + *λF* (*u*) = (47)

= 1 *|∇u|pdx* + *λ*

*p* Ω

Ω

(*Ku − f* )2*dx* =

= 1 *||∇u||p* + *λ||Ku − f||*2 *< ∞*

*p*

*p*

2

que recuperan el caso lineal para *p* = 2 pero permiten considerar procesos no lineales menos agresivos sobre la soluci´on del problema para 1 *p <* 2. El ejemplo estrella en el procesado de im´agenes digitales mediante m´etodos variacionales es el operador de Variaci´on Total cuya definici´on (no rigurosa[13](#_bookmark165)) es

*≤*

*G*1(*u*) = *TV* (*u*) =

Ω

*|∇u|dx* = *||∇u||*1

y se corresponde a la elecci´on de *p* = 1. La energ´ıa correspondiente es

*E*1(*u*) = *G*1(*u*) + *λF* (*u*) = (48)

= Ω *|∇u|dx* + *λ* Ω(*Ku − f* )2*dx* =

= *||∇u||*1 + *λ||Ku − f||*2 *< ∞*

2

El operador de Variaci´on Total fue introducido en el procesado de im´agenes y en Computer Vision por L. Rudin, S. Osher y E. Fatemi[14](#_bookmark166) al comienzo de los an˜os noventa. Desde entonces ha recibido m´as de 12000 citas en art´ıculos con ´ındice de impacto. La caracter´ıstica principal de este operador es que las soluciones de los problemas asociados no son continuas y preservan los bordes de la imagen. El efecto de la regularizaci´on depende del par´ametro *p*:

13La definici´on no es rigurosa ya que no es posible asegurar que *∇u ∈ L*1(Ω) es decir que *u* 1 *<* . Para ello es necesaria una ulterior regularizaci´on.

14L. Rudin, S. Osher y E. Fatemi. Nonlinear total variation based noise removal algo- rithms. Physica D: Nonlinear Phenomena, ISSN: 0167-2789, Vol: 60, Issue: 1, Page: 259-268

*||∇ || ∞*

Publication Year: 1992

* + - 1. Para *p >* 1 el funcional ([47](#_bookmark164)) es Diferenciable y estrictamente convexo.
      2. Para *p* = 1 el funcional ([47](#_bookmark164)) no es Diferenciable (es necesaria regulari- zacio´on) pero es estrictamente convexo. El operador *G*1(*u*) es convexo.
      3. Para 0 *< p <* 1 el funcional ([47](#_bookmark164)) no es Diferenciable (es necesaria regularizacio´on) y no es convexo.

##### Ecuaciones de Euler-Lagrange

Siguiendo el m´etodo variacional el problema de optimizaci´on ([56](#_bookmark178)) (estric- tamente convexo) se resuelve considerando pequen˜as perturbaciones de la energ´ıa en todas las direcciones y pasando al l´ımite. Para ello se define la **Diferencial de Gateaux** del funcional

*⟨∂E*(*u*)*, v⟩*

a partir de la cual obtenemos la expresi´on del gradiente del funcional.

Sea *V* un espacio vectorial infinito dimensional y *E* : *V* R un funcional definido en *V* . Si existe finito el l´ımite

*→*

*⟨∂E*(*u*)*, v⟩* = l´ım (*E*(*u* + *αv*) *− E*(*u*) (49)

*α→*0 *α*

diremos que *E* es **Gateaux Diferenciable** y

*∂vE*(*u*) = *⟨∂E*(*u*)*, v⟩*

es la derivada direccional de *E* en el punto *u* y en la direcci´on de *v* siendo

*∂E*(*u*) denotada (impropiamente) por el gradiente del funcional:

*∂E*(*u*) = *∇E*(*u*)

##### Operador laplaciano

Considerando la energ´ıa cuadr´atica en ([46](#_bookmark162)) se tiene

*∇E*(*u*) = *−*∆*u* + *λK* (*Ku − f* )

*T*

Imponiendo su anulaci´on (condiciones necesarias de I orden) obtenemos la Ecuaci´on de Euler-Lagrange

*∇E*(*u*) = 0

del funcional a minimizar

*−*∆*u* + *λK* (*Ku − f* ) = 0 (50)

*T*

que en matem´aticas se clasifica como una ecuaci´on diferencial lineal de se- gundo orden de tipo el´ıptico para el operador de Laplace ∆*u*. La soluci´on *u*(**x**) depende del punto (pixel o voxel) y no depende del tiempo. Se considera por tanto de tipo estacionario. Describe el estado de equilibrio de un proceso de difusi´on lineal

*∂u* = *−∇E*(*u*) = ∆*u − λKT* (*Ku − f* )

*∂t*

que se estabiliza (para *t →* +*∞*) en la soluci´on *u∗*(**x**) de la ecuaci´on ([50](#_bookmark169)):

l´ım

*t→*+*∞*

##### Operadores *p−*laplacianos

*u*(**x***, t*) = *u∗*(**x**)

Considerando un problema de reconstrucci´on o restauraci´on modelado por un operador *K* = *I* y utilizando un regularizante de la forma ([47](#_bookmark164)), repetimos el procedimiento anterior para calcular la derivada de Gateaux del funcional *Ep*(*u*)

*̸*

*∇Ep*(*u*) = *−*∆*pu* + *λK* (*Ku − f* ) obteniendo la Ecuaci´on de Euler-Lagrange

*T*

*−*∆*pu* + *λK* (*Ku − f* ) = 0*, p ≥* 1 (51)

*T*

que es una ecuaci´on en derivadas parciales el´ıptica no lineal. Describe el estado de equilibrio de un proceso de difusi´on no lineal

*∂u* = ∆ *u − λKT* (*Ku − f* )

*∂t*

*p*

que se estabiliza (para *t →* +*∞*) en la soluci´on *u∗*(**x**) de la ecuaci´on ([51](#_bookmark170)):

l´ım

*t→*+*∞*

*u*(**x***, t*) = *u∗*(**x**)

El operador ∆*pu* se conoce con el nombre de **operador** *p***-laplaciano** y se define por

∆*pu* = div(*|∇u|p−*2*∇u*) (52)

Para *p* = 2 recuperamos el operador de Laplace definido en ([57](#_bookmark181)). Para *p >* 2 la ecuaci´on no lineal asociada a ([52](#_bookmark171)) se define de **degenerada** ya que

*|∇u|p−*2 *→* 0*,* para *|∇u| →* 0

Para 1 *< p <* 2 la ecuaci´on no lineal asociada a ([52](#_bookmark171)) se define de **singular**

ya que

*|∇u|p−*2 *→* +*∞,* para *|∇u| →* 0

Utilizando las ecuaciones ([50](#_bookmark169)) y ([51](#_bookmark170)) definimos formalmente el problema a resolver para la minimizaci´on de funcionales de energ´ıa convexos del tipo ([47](#_bookmark164)):

*−*∆*pu* + *λKT* (*Ku − f* ) = 0*,* **x** *∈* Ω

f

*∇u ·* **n** = 0*,* **x** *∈ ∂*Ω

(53)

en donde hemos impuesto condiciones de contorno de tipo Neumann ho- mog´eneas (de no flujo o flujo nulo en la frontera) definidas por

*∇u*(**x**) *·* **n** = 0*,* **x** *∈ ∂*Ω

Este tipo de condiciones permite conservar la masa de la soluci´on:

*ud***x** = *fd***x**

Ω Ω

y son las que se utilizan t´ıpicamente en el procesado de im´agenes. La soluci´on

*u∗* del problema

*u∗* = arg m´ın *Ep*(*u*)

*u*

es soluci´on del problema ([53](#_bookmark172)).

##### Descenso Gradiente y Filtrado

La resoluci´on num´erica del del problema ([53](#_bookmark172)) (definido a trav´es de la ecuaci´on lineal en ([50](#_bookmark169)) o no lineal ([51](#_bookmark170))) implica, tras discretizaci´on de las derivadas que aparecen en las ecuaciones, la resoluci´on de un sistema de ecuaciones lineales del taman˜o del nu´mero de p´ıxeles de la imagen o de una secuencia

de sistemas lineales en el caso no lineal, *p* = 2. Una alternativa que permite introducir unas ideas y conceptos b´asicos para la optimizaci´on en aprendizaje autom´atico es el m´etodo del descenso gradiente que presentamos a continua- ci´on. Para ello introducimos un tiempo ficticio, artificial *t* y consideramos una relajaci´on de la igualdad en ([51](#_bookmark170)) en la forma

*̸*

*∂u*

*∂t* = *−∇Ep*(*u*)

es decir siguiendo la direcci´on contraria al gradiente del funcional. Por lo anterior se tiene que resolver un problema transitorio (en el tiempo) definido por

****

****

*∂u* = ∆ *u − λKT* (*Ku − f* )*,* (**x***, t*) *∈* Ω *×* (0*,* +*∞*)

(54)

*∂t p*

*∇u ·* **n** = 0*,* (**x***, t*) *∈ ∂*Ω *×* (0*,* +*∞*)

*u*(**x***,* 0) = *f* (**x**)*,* **x** *∈* Ω

lo que se corresponde a realizar un proceso f´ısico de difusi´on sobre el dominio de la imagen Ω complementado con condiciones de contorno de tipo Neumann homog´eneas (de no flujo) en la frontera *∂*Ω definidas por

*∇u*(**x***, t*) *·* **n** = 0*,* **x** *∈ ∂*Ω Dependiendo del valor de *p* se modelan distintos fen´omenos.

* + - 1. Para *p >* 2 la ecuaci´on es degenerada, hablamos de **difusi´on lenta** y hay emborronamiento de la imagen debido a la excesiva regularizaci´on.
      2. Para *p* = 2 la ecuaci´on es lineal. Se sigue utilizando este caso en el pro- cesamiento de im´agenes pero produce emborronamiento de los bordes y p´erdida de informaci´on.
      3. Para 1 *< p <* 2 la difusi´on es r´apida y aparecen regiones homog´eneas que favorecen la clasificaci´on de las im´agenes.
      4. Para *p* = 1 se tiene el operador de variaci´on total, el m´as utilizado en procesamiento de im´agenes por su capacidad de preservar los bordes.
      5. Bajo ciertas hip´otesis y previa aproximaci´on y posible incluso conside- rar fen´omenos difusivos para valores 0 *< p <* 1. Se generan procesos de difusi´on que tienden a gradientes sparse.

El proceso de difusi´on empieza en *t* = 0 con *u*(**x***,* 0) = *f* (**x**), es decir los datos iniciales vienen dados por la imagen ruidosa y termina cuando

*∂u*

*∂t →* 0

lo que ocurre en tiempo (artificial) infinito. Cuando utilizamos el m´etodo del descenso gradiente para determinar un m´ınimo del funcional se trata de un proceso de **estabilizaci´on** en donde el destino final de la soluci´on *u*(**x***, t*) de la ecuaci´on transitoria ([54](#_bookmark174)) es la soluci´on *u*(**x**) de la Ecuaci´on de Euler- Lagrange ([50](#_bookmark169)):

l´ım *u*(**x***, t*) = *u*(**x**)

*t→∞*

En la pr´actica se utiliza el resultado

*∂u*

*∂t ≈* 0

ya que la convergencia en este esquema es monotona. Observamos finalmente que, cuando *λ* = 0, las ecuaciones de tipo ([54](#_bookmark174)) modelan **procesos de filtrado** (paso bajo) de las im´agenes:

****

****



*∂u*

*∂t* = ∆*pu,* (**x***, t*) *∈* Ω *×* (0*,* +*∞*)

(55)

*∇u ·* **n** = 0*,* (**x***, t*) *∈ ∂*Ω *×* (0*,* +*∞*)

*u*(**x***,* 0) = *f* (**x**)*,* **x** *∈* Ω

La diferencia fundamental entre el **descenso gradiente** ([54](#_bookmark174)) y el **proceso de filtrado** ([55](#_bookmark175)) radica en que ([54](#_bookmark174)) se tiene que resolver hasta estabilizaci´on de la soluci´on mientras que ([55](#_bookmark175)) se tiene que resolver por poco tiempo para no perder estructuras (bordes) en la imagen. Pasaremos ahora a resolver la ecuaci´on transitoria ([54](#_bookmark174)) mediante el m´etodo de la diferencias finitas.

##### Formulacio´n Bayesiana

El funcional modelo introducido en ([46](#_bookmark162)) puede ser justificado a partir de la estad´ıstica bayesiana. Esto nos propone un marco muy general que ser´a explotado en la segunda parte del curso al definir las redes bayesianas. Su- pondremos a lo largo de la deducci´on del modelo y por simplicidad, que *p* = 2 y *K* = *Id*. De momento es suficiente considerar la f´ormula de Bayes

p(*u f* ) = p(*f|u*)p(*u*)

*|*

p(*f* )

que define la probabilidad condicionada **a posterior** p(*u f* ) en t´erminos de la probabilidad de la **verosimilitud** con los datos p(*f u*) y las probabilidades **a priori** p(*u*), p(*f* ). La formulaci´on bayesiana, que ser´a presentada en detalles al comienzo de la segunda parte del curso, permite realizar una estimaci´on *u∗* de tipo MAP (**Maximum A Posteriori**) de la imagen ´optima *u* que maximiza esta probabilidad a partir de una imagen *f* (los datos):

*|*

*|*

*u∗* = arg m´ax (log p(*f u*) + log p(*u*))

*|*

*u*

Para ello se razona de la siguiente forma. Los detalles se pueden encontrar en el famoso art´ıculo de Geman y Geman, (1984) [15](#_bookmark177) Tomando logaritmos y cambiando de signo, aplicamos las propiedades de los logaritmos para obtener la energ´ıa a minimizar:

*E*(*u*) = *−* log p(*u|f* ) = *−* log (p(*f|u*)p(*u*) =

p(*f* )

= *−* log p(*f|u*) *−* log p(*u*) + log p(*f* )

Pasando a la formulaci´on de argumento m´ınimo vemos que la estimaci´on MAP *u∗* es

*u∗* = arg m´ın *E*(*u*) = arg m´ın(*−* log p(*f|u*) *−* log p(*u*) + log p(*f* ))

*u u*

= arg m´ın (*−* log p(*f|u*) *−* log p(*u*))

*u*

ya que p(*f* ) no depende de *u* luego no cambia el punto (funci´on) en donde se alcanza el m´ınimo. En un modelo de ruido gaussiano

15S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Machine Intell., vol. 6, pp. 721-741, July 1984.

*f* = *u* + *n*

se tiene que *f u* = *n* es una distribuci´on de tipo Gaussiano que se modela en la forma

*−*

p(*f|u*) = *e−λ||u−f||*2

2

siendo *λ* un par´ametro positivo (hiper-par´ametro en la terminolog´ıa bayesia- na) que depende de la variancia del ruido *σ*2 y que se tiene que estimar y fijar antes de realizar las simulaciones num´ericas. Veremos en las aplicaciones al- gunas variaciones como por ejemplo el uso de un hiperpar´ametro adaptativo *λ*(**x**). Tomando logaritmo y cambiando de signo se tiene

*—* log p(*f|u*) = *−* log *e−λ||u−f||*2 = *λ||u − f||*2

2

2

De manera an´aloga modelamos el t´ermino a priori en la forma

p(*u*) = *e−||∇u||*2

2

luego

*—* log p(*u*) = *−* log *e−||∇u||*2 = *||∇u||*2

2

2

y hemos deducido el problema de minimizaci´o

arg m´ın *E*(*u*) = arg m´ın (*−* log p(*f|u*) *−* log p(*u*)) =

*u u*

= arg m´ın *||∇u||*2 + *λ||u − f||*2 (56)

*u*

2

2

La relaci´on entre la estimaci´on probabilistica y la variacional se puede en- contrar en el art´ıculo de Hamza y Krim (2001)[16](#_bookmark179)

16A. Ben Hamza and H. Krim, A variational approach to maximum a pos- teriori es- timation For Image denoising, Lecture Notes in Comput. Sci., vol. 2134, pp. 19-34, Sept. 2001

##### C´alculo riguroso de las Ecuaciones de Euler-Lagrange

Por simplicidad consideraremos el caso lineal para eliminaci´on de ruido. Cal- culamos el l´ımite ([49](#_bookmark168)). En el caso del funcional ([46](#_bookmark162)), *E*(*u*) = *E*2(*u*) y se tiene

*E*2(*u* + *αv*) *− E*2(*u*) =

= *||∇*(*u* + *αv*)*||*2 + *λ||*(*u* + *αv*) *− f||*2 *− ||∇u||*2 *− λ||u − f||*2 =

2

2

2

2

= *|∇*(*u* + *αv*)*|*2*dx − |∇u|*2*dx* + *λ* (*u* + *αv − f* )2*dx − λ* (*u − f* )2*dx* Considerando cada t´ermino por separado

Ω Ω Ω Ω

*|∇*(*u* + *αv*)*|*2*dx* =

2

2

Ω

Ω

*|∇u* + *α∇v*)*| dx* =

= *|∇u|*2*dx* + 2*α*

Ω

Ω

y

*∇u∇vdx* + *α*2

Ω

*|∇v| dx*

*λ* (*u* + *αv − f* )2*dx* = *λ*

Ω

Ω

(*u − f* + *αv*)2*dx* =

luego

2

= *λ*

Ω

(*u − f* )2*dx* + 2*λα*

Ω

(*u − f* )*vdx* + *λα*2

Ω

*v*2*dx*

*|∇*(*u* + *αv*)*|*2*dx −*

Ω

Ω

y

*|∇u|*2*dx* = 2*α*

Ω

*∇u∇vdx* + *α*2

Ω

*|∇v| dx*

*λ* (*u* + *αv − f* )2*dx − λ*

Ω

Ω

(*u − f* )2*dx* =

= 2*λα*

Ω

(*u − f* )*vdx* + *λα*2

Ω

*v*2*dx*

El cociente en ([49](#_bookmark168)) es por tanto

*E*(*u* + *αv*) *− E*(*u*) =

*α*

= 2

Ω

*∇u∇vdx* + *α*

Ω

*|∇v|*2*dx* + 2*λ*

Ω

(*u − f* )*vdx* + *λα*

Ω

*v*2*dx*

y pasando al l´ımite para *α →* 0 se obtiene

*⟨DE*(*u*)*, v⟩* = 2 *∇u∇vdx* + 2*λ*

Ω

Ω

(*u − f* )*vdx* =

= *−*2*⟨*∆*u, v⟩* + 2*λ⟨u − f, v⟩* = 2*⟨−*∆*u* + *λ*(*u − f* )*, v⟩*

en donde hemos utilizado las f´ormulas de Green y las condiciones de contorno de no flujo a trav´es de la frontera:

*∇u∇vdx* = *−*

Ω

Ω

∆*uvdx* +

*∂*Ω

*∇uv***n***dx* = *−*

Ω

∆*uvdx*

siendo ∆*u* el operador de Laplace definido por

∆*u* = div(*∇u*) (57)

Imponiendo la condici´on necesaria de anulaci´on del gradiente

*⟨∂E*(*u*)*, v⟩* = 2*⟨−*∆*u* + *λ*(*u − f* )*, v⟩* = 0*, ∀ v ∈ V*

se deduce la Ecuaci´on de Euler-Lagrange del funcional a minimizar

*−*∆*u* + *λ*(*u − f* ) = 0

M´as en general se pueden calcular las Ecuaciones de Euler-Lagrange utili- zando las siguientes f´ormulas: Sea dado el funcional a minimizar

*E*(*u*) =

Ω

*F* (**x***, u, ∇u*)*d***x**

siendo Ω *⊂* R*n*. Entonces las Ecuaciones de Euler-Lagrange son

( *∇*

*n*

*— i*=1

*∂*

*∂xi*

*∂F* (**x***, u, u*)

*∂pi*

*∂F*

+ *∂u* (**x***, u, ∇u*) = 0

**Ejemplo 10.1.** *Sea* Ω R2 *y sea dado el funcional de energ´ıa E*(*u*) *definido por*

*⊂*

*F* (**x***, u, ∇u*) = 1 *|∇u|*2 +

2

*λ*

(*u f* )2

*−*

2

*Calcular las Ecuaciones de Euler-Lagrange del problema de minimizaci´on asociado.*

Para ello definimos **x** = (*x, y*) y

**p** = (*p*1*, p*2)*T* = (*∇u*)*T* = (*uxuy*)*T*

siendo

*F* (**x***, u, ∇u*) = *F* (**x***, u,* **p**) = 1 *|***p***|*2

*λ* 2

*u − f*

1 *p*2 + *p*2

*λ*

*u − f* )2

Se tiene

*∂F*

+ ( )

2 2

*∂F*

= 2 ( 1

2) + 2 (

*∂p*1

y

(**x***, u, ∇u*) = *p*1 = *ux,*

*∂F*

*∂p*2

(**x***, u, ∇u*) = *p*2 = *uy*

luego la ecuaci´on

*∂u* (**x***, u, ∇u*) = *λ*(*u − f* )

*n*

*— i*=1

es

*∂*

*∂xi*

*∂F* (**x***, u, u*)

*∂pi*

( *∇*

*∂F*

+ *∂u* (**x***, u, ∇u*) = 0

y se escribe en la forma

*−*(*uxx* + *uyy*) + *λ*(*u − f* ) = 0

*−*∆*u* + *λ*(*u − f* ) = 0

##### Discretizaci´on: F´ormulas en diferencias finitas

Describimos brevemente a las f´ormulas en diferencias finitas que permiten discretizar las derivadas parciales que aparecen en los operadores gradiente, divergencia y p-laplaciano. Definimos una f´ormula en **diferencia finita de primer orden de tipo progresivo** para estimar la derivada parcial de una funci´on en un punto mediante

*∂u* (*x, y*) *≈ u*(*x* + *h, y*) *− u*(*x, y*) *, ∂u* (*x, y*) *≈ u*(*x, y* + *k*) *− u*(*x, y*)

*∂x*

*h*

*∂y*

*k*

Ana´logamente definimos una f´ormula en **diferencia finita de primer or- den de tipo regresivo** como

*∂u* (*x, y*) *≈ u*(*x, y*) *− u*(*x − h, y*) *, ∂u* (*x, y*) *≈ u*(*x, y*) *− u*(*x, y − k*)

*∂x*

*h*

*∂y*

*k*

Otra posibilidad es la f´ormula en **diferencias finitas de primer orden centradas**:

*∂u* (*x, y*) *≈ u*(*x* + *h, y*) *− u*(*x − h, y*) *, ∂u* (*x, y*) *≈ u*(*x, y* + *k*) *− u*(*x, y − k*)

*∂x*

2*h*

*∂y*

2*k*

Para las derivadas segundas utilizaremos f´ormula en **diferencias finitas de segundo orden centradas**:

*∂*2*u*

*∂x*2 (*x, y*) *≈*

*∂*2*u*

*∂y*2 (*x, y*) *≈*

*u*(*x* + *h, y*) *−* 2*u*(*x, y*) + *u*(*x − h, y*) *, h*2

*u*(*x, y* + *k*) *−* 2*u*(*x, y*) + *u*(*x, y − k*) *k*2

a partir de las cuales contruimos una aproximaci´on centrada del operador de Laplace.

∆*u*(*x, y*) *≈*

*≈ u*(*x* + *h, y*) *−* 2*u*(*x, y*) + *u*(*x − h, y*) + *u*(*x, y* + *k*) *−* 2*u*(*x, y*) + *u*(*x, y − k*)

*h*2

*k*2

Suponiendo *h* = *k* (mallado is´otropo) y *h* = *k* = 1 (t´ıpico en procesado de imagen) se tiene

∆*u*(*x, y*) *≈ u*(*x* + *h, y*) + *u*(*x, y* + *k*) *−* 4*u*(*x, y*) + *u*(*x − h, y*) + *u*(*x, y − k*)

conocida como la aproximaci´on de 5 puntos en cruz del operador de Laplace. Se observa que la f´ormula centrada anterior para el c´alculo del operador de Laplace se puede obtener calculando primero el operador gradiente median- te diferencias finitas progresiva y luego su divergencia mediante diferencias finitas regresivas lo que genera una aproximaci´on centrada. Tambi´en discre- tizaremos en el tiempo, en la forma *tk* = *t*0 + *k*∆*t*, *uk* = *u*(**x***, tk*) y usando una f´ormula progresiva de I orden

*∂u*

*∂t ≈*

*uk*+1 *− uk*

∆*t*

## XIII Clase

#### C´alculo Variacional: Planteamiento de la Pr´actica

##### Resoluci´on Num´erica: Minimizaci´on de Funcionales de Energ´ıa

## XIV Clase

#### Recuperaci´on, dudas y ejercicios

## Ap´endice: Aplicaciones del gradiente de una imagen

Las posibles aplicaciones de este marco son much´ısimas y abarcan el filtrado, la eliminaci´on de ruido, la reconstrucci´on mediante deconvoluci´on, la restau- raci´on por p´erdida de informaci´on o inpainting, el flujo o´ptico, el registro, la segmentaci´on, la super-resoluci´on entre otros. Veamos a continuaci´on unos primeros ejemplos b´asicos que ser´an desarrollados durante las pr´acticas de- dicadas a la resoluci´on num´erica mediante matlab.

#### C´alculo del Gradiente de una imagen digital

El c´alculo del Gradiente de una imagen digital *u* se realiza calculando las derivadas parciales de la imagen. Cada una de las derivadas nos informa de la localizaci´on de los bordes en la imagen. Por ejemplo si calculamos la derivada parcial *ux* tenemos informaci´on sobre los bordes verticales que se dectan calculando las variaciones horizontales de la imagen:



Figura 32: Derivada parcial *Ix* de la imagen obtenida con una f´ormula de derivaci´on impl´ıcita de 7 puntos.

Los bordes horizontales se detectar´an calculando la derivada *uy*

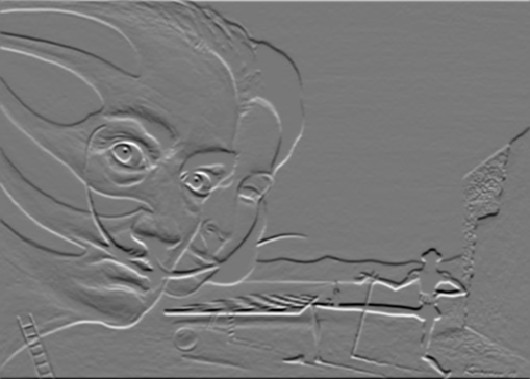


Figura 33: Derivada parcial *Iy* de la imagen obtenida con una f´ormula de derivaci´on impl´ıcita de 7 puntos.

Podemos entonces juntar la informaci´on de las derivadas parciales calculando la magnitud o m´odulo del gradiente

*|∇u|* = *u*2 + *u*2

Una vez calculado el gradiente podemos calcular el operador de laplace, o laplaciano en la forma

*x*

*y*

div(*∇u*) = ∆*u*

Este operador servir´a para modelar procesos de difusi´on (filtrado) y Descen- sos del Gradiente. Al igual que el m´odulo del gradiente tiene informaci´on de alta frecuencia. Al tenerse que calcular dos derivadas de la misma imagen y al introducirse por tanto mayores errores de discretizaci´on, la imagen re- sultante del operador de laplace se considera un detector de bordes menos fiable que el m´odulo del gradiente. Los m´etodos que se basan en el c´alculo



Figura 34: M´odulo del Gradiente de la imagen

del laplaciano se llaman **zero-crossing** ya que al buscar la anulaci´on del la- placiano estamos imponiendo condiciones de optimalidad sobre el gradiente que en aquellos puntos es m´aximo. Un simple c´odigo para el c´alculo de todos estos operadores se presenta a continuaci´on

%% Aplicaciones clear all; close all;

IO=imread(’./imagenes/dali-bw.jpg’); I = im2double(IO);

[m, n]=size(I); hx=1;

hy=1;

%% Calculo derivadas parciales en diferencias finitas disp(’Calculo derivadas parciales en diferencias finitas’)

%primera derivada progresiva (i+1 - i)

I\_x = [I(:,2:end)-I(:,1:end-1), zeros(m,1)]/hx;

I\_y = [I(2:end,:)-I(1:end-1,:); zeros(1,n)]/hy;

%% funcion gradient matlab

disp(’Calculo derivadas parciales mediante gradient’) [fx,fy]=gradient(I);

%% Calculo derivadas parciales implicitas 5 puntos disp(’Calculo derivadas parciales implicitas 5 puntos’)

% derivadas implicitas 5 puntos

[gx5, gy5] = derivative5(I, ’x’, ’y’);

%% Calculo derivadas parciales implicitas 7 puntos disp(’Calculo derivadas parciales implicitas 7 puntos’) [gx7, gy7] = derivative7(I, ’x’, ’y’);

%% magnitud gradiente

disp(’Calculo del M´odulo del Gradiente’) mag\_gradiente = sqrt((gx7).^2 + (gy7).^2);

%% orientacion gradiente

th = atan2(-gy7,gx7); % a´ngulos de -pi a + pi.

th(th<0) = th(th<0)+pi; % Mapea ´angulos a 0-pi. th = th\*180/pi;

%% c´alculo derivadas segundas y operador de Laplace disp(’Calculo derivadas segundas’) I\_xx=[zeros(m,1), -I\_x(:,1:end-1)+I\_x(:,2:end)]/hx;

I\_yy=[zeros(1,n); -I\_y(1:end-1,:)+I\_y(2:end, :)]/hy; disp(’Calculo operador Laplaciano’)

Lap=(I\_xx+I\_yy);

#### Determinacio´n de los bordes de una imagen

Tras haber visto la aplicabilidad del campo escalar del m´odulo del gradiente para la detecci´on de bordes consideraremos varios algoritmos basados en el c´alculo de las derivadas parciales que pueden ser utilizados como detectores de bordes. El comando **edge.m** de matlab permite por ejemplo la detecci´on de bordes mediante la implementaci´on de los algoritmos de Canny, Prewitt, Roberts, Logaritmos de gaussiana, Zero-Crossing. La sintaxis para imple- mentar los distintos algoritmos es

%% algoritmos para detecci´on de bordes BWC = edge(I,’Canny’);

BWA = edge(I,’approxcanny’); BWP = edge(I,’Prewitt’);

BWS = edge(I,’Sobel’); BWR = edge(I,’Roberts’); BWL = edge(I,’log’);

BWZ = edge(I,’zerocross’);



Figura 35: Detector de bordes de Canny

Tambi´en es posible generar una m´ascara binaria de los bordes utilizando el algoritmo de Canny mediante las siguientes instrucciones que se basan en el comando **canny.m**

IO=imread(’./imagenes/dali-bw.jpg’); I = im2double(IO);

sigma=0.5

[gr, or] = canny(I, sigma);

%% mascara canny rad=1.5

nm = nonmaxsup(gr, or, rad);

T1=0.02; T2=0.12

bw = hysthresh(nm, T1, T2);



Figura 36: Detector de bordes de Canny: los bordes se almacenan en una m´ascara binaria

## Ap´endice: Ecuaciones no lineales

#### Motivaci´on y generalidades

**Ejemplo 14.1.** *(Hanna et al.[9]): La ecuaci´on de Peng-Robinson es una ecuaci´on de estado que proporciona la presi´on P de un gas mediante:*

*P* = *R · T*

*V − b*

*a*

*— V ·* (*V* + *b*) + *b ·* (*V − b*)

*donde a y b son constantes, T es la temperatura absoluta a la que se encuentra el gas, V es el volumen espec´ıfico y R es la constante de los gases perfectos (8.31441J/(mol .oK)). Para el CO*2 *las constantes a y b toman los valores a*

*= 364.61 m*6*.kPa / (kg.mol)*2 *y b = 0.02664 m*3*/kg.mol. Supongamos que se desea encontrar la densidad (es decir 1/V) del CO*2 *a una presi´on de 1 10*4 *kPa y a una temperatura de 340oK usando la ecuaci´on de Peng-Robinson. Ello implicar´ıa tener que encontrar el valor de V para el que:*

*·*

1 *·* 104 = 8*.*31441 *·* 340 *−* 364*.*61

*V −* 0*.*02664 *V ·* (*V* + 0*.*02664) + 0*.*02664 *·* (*V −* 0*.*02664)

*lo cual no es en modo alguno evidente. La soluci´on ..... m´as adelante.*

La cuesti´on es ¿c´omo se han calculado estas soluciones?

El ejemplo anterior pretende ilustrar el hecho de que en numerosas aplica- ciones f´ısicas y t´ecnicas, y en concreto en muchos problemas propios de la Ingenier´ıa, aparece la necesidad de tener que resolver ecuaciones o sistemas de ecuaciones no lineales.

Este tipo de sistemas tiene peculiaridades que los diferencian notablemente de los sistemas lineales. As´ı por ejemplo, los sistemas lineales de n ecuaciones con n inc´ognitas en los que la matriz del sistema es regular s´olo admiten una soluci´on. A diferencia de este caso, los sistemas no lineales, aunque tengan el mismo nu´mero de inc´ognitas que de ecuaciones, desde un punto de vista matem´atico, pueden admitir una, ninguna o varias soluciones. El elegir entre ellas las que sirven a la aplicaci´on concreta que motiv´o el sistema de ecua- ciones debe hacerse en funci´on de los criterios f´ısicos, qu´ımicos y t´ecnicos que regulen el problema en cuesti´on (por ejemplo, aunque matem´aticamente puedan tener sentido, qu´ımicamente ser´ıan inadmisibles fracciones molares negativas o superiores a 1 de una especie qu´ımica).

Una segunda diferencia es la debida al hecho de que un sistema lineal que ad- mita soluci´on u´nica puede ser resuelto de forma exacta mediante un nu´mero

finito de operaciones (recu´erdense los m´etodos directos de resoluci´on de sis- temas lineales de ecuaciones: Gauss, LU, Choleski, Crout, QR, etc...). En el caso de los sistemas no lineales, en general, la soluci´on no podr´a ser encontra- da mediante un nu´mero finito de operaciones. En este sentido, los m´etodos de resoluci´on de sistemas de ecuaciones no lineales ser´an m´etodos de tipo iterativo mediante los cuales se construir´a una sucesi´on de vectores que, en los casos en que el m´etodo funcione, se ir´an aproximando hacia uno de los vectores soluci´on del sistema no lineal.

**Observaci´on 14.1.** *Interpr´etese correctamente lo que se acaba de leer. No quiere ello decir que ninguna ecuaci´on no lineal pueda resolverse de forma directa. Ah´ı est´an las ecuaciones de segundo grado que, siendo no lineales, pueden resolverse de forma exacta mediante un nu´mero finito de operaciones. O si se buscan las soluciones de la ecuaci´on:* (*x* 1)10 = 0 *tambi´en es obvio que estas son x* = 1 *(con multiplicidad* 10*). Lo que se est´a diciendo es que no hay, por ejemplo, ningu´n m´etodo directo que nos permita calcular las ra´ıces de cualquier polinomio de grado* 10*.*

*−*

El hecho de que los m´etodos de resoluci´on de ecuaciones y sistemas no lineales sean de tipo iterativo nos plantea muchas cuestiones. Entre ellas cabe citar las siguientes:

1. ¿C´omo se genera la sucesi´on de vectores que puedan aproximarse a la soluci´on?
2. Dado que es imposible evaluar los infinitos vectores de la sucesi´on an- terior, ¿c´omo se sabe que ya se est´a “suficientemente” cerca de una soluci´on?.
3. Si la soluci´on encontrada mediante un m´etodo no es la que pueda in- teresarnos ¿c´omo buscar otras posibles soluciones?.
4. En el caso de tener diferentes m´etodos que nos proporcionen las solu- ciones de un sistema ¿c´omo elegir el mejor entre ellos?.

A estas y otras cuestiones intentaremos dar respuesta en el presente tema. La descripci´on general de los principales m´etodos de resoluci´on puede hacerse de una forma muy intuitiva sin necesidad de recurrir a artificios matem´aticos complicados. No obstante la justificaci´on rigurosa de las t´ecnicas de resoluci´on

y el an´alisis de las condiciones que pueden garantizar su convergencia as´ı como el estudio de la velocidad con que convergen exigir´a acudir a conceptos matem´aticos previos.

Conviene por u´ltimo que el lector tome conciencia desde el primer momento de un hecho relativo a los m´etodos de resoluci´on de sistemas de ecuaciones no lineales: **no existe un m´etodo universal de resoluci´on de sistemas de ecuaciones no lineales**. Algunos de ellos funcionar´an sobre ciertos sistemas y no servir´an para resolver otros. Los m´etodos que presenten un buen com- portamiento sobre algunos sistemas pueden no ser los mejores para resolver otros sistemas diferentes. M´as bien cabr´ıa decir que **cada sistema no lineal requerir´a su m´etodo de resoluci´on id´oneo**.

En los m´etodos iterativos que plantearemos en este tema la sucesi´on de vec- tores que, en su caso, vayan aproxim´andose hacia la soluci´on se generar´a a partir de un vector inicial **x**(0) mediante un esquema de c´alculo que se tradu-

cira´

a:

en determinar una funci´on **g**(**x**) con la que el proceso iterativo se reducir´a

**x**(0) *dado*

**x**(*i*+1) = **g**(**x**(*i*)) (*i* = 0*,* 1*,* 2*,* )

M´as adelante estudiaremos la forma de definir esta funci´on **g**(**x**) (que obvia- mente tendr´a que ver con el sistema que se quiera resolver). No obstante ya puede apreciarse con la consideraci´on que acaba de hacerse que el buen fun- cionamiento de un m´etodo depender´a de c´omo sea la funci´on **g**(**x**) escogida. Por tanto, nos interesar´a trabajar, cuando ello sea posible con aplicaciones **g**(**x**) para las que se pueda asegurar que la sucesi´on que con ellas se genere es convergente hacia alguna soluci´on del sistema que se quiere resolver.

#### M´etodos generales para la resoluci´on de una u´ni- ca ecuacio´n no lineal

En este apartado se expondr´an m´etodos generales que nos permiten encontrar soluciones de una ecuaci´on no lineal. Como se sen˜ala en el u´ltimo subaparta- do, para ecuaciones no lineales concretas (por ejemplo polin´omicas) es posible adaptar algunos de estos m´etodos y, en algunos casos, existen m´etodos de aplicaci´on espec´ıfica a ellas.

Una ecuaci´on no lineal la representaremos gen´ericamente en la forma *f* (*x*) =

1. Obs´ervese que lo anterior no quiere decir que la funci´on *f* (*x*) sea la funci´on id´enticamente nula. Simplemente es una forma de representar la ecuaci´on a

la que nos enfrentemos. M´as concretamente tras la expresi´on *f* (*x*) = 0 se ocultar´a el siguiente problema:

*Dada una funci´on f* (*x*) *determ´ınese, si es posible, algu´n valor x∗ para el que se verifique que f* (*x∗*) = 0.

A los valores *x∗* para los que la funci´on *f* (*x*) se anula habitualmente se les denomina ra´ıces (o ceros) de la funci´on. En general si *f* (*x*) admite el valor *x∗* como ra´ız, se podr´a encontrar un nu´mero positivo *m* y una funci´on *φ*(*x*) tales que *φ*(*x∗*) *̸*= 0 y, en un entorno de *x∗*,

*f* (*x*) = (*x − x∗*)*m · φ*(*x*)*.*

En esa situaci´on se dir´a que *x∗* es una ra´ız de multiplicidad *m.* As´ı por ejemplo, la funci´on *f* (*x*) = (*x −* 1)2(*x* + 2) sin(*x*) admite, entre otras, las ra´ıces *x∗*1 = 1 (de multiplicidad 2) y *x∗*2 = *−*2 (de multiplicidad 1). O la funci´on *f* (*x*) = 3 *x*(*x −* 4)2*ex* admite la ra´ız *x∗*1 = 0 (de multiplicidad 1/3) y la ra´ız *x∗*2 = 4 (de multiplicidad 2).

*√*

No siempre est´a tan claro cu´al es la multiplicidad de una ra´ız. Por ejemplo,

la funci´on *f* (*x*) = sin(*x*) tiene una ra´ız en *x* = 0*.* Para expresarla en la forma *f* (*x*) = *x g*(*x*) la funci´on *g*(*x*) deber´ıa ser *g*(*x*) = sin(*x*)*/x* pero esta funci´on no est´a definida en *x* = 0. Por ello, teniendo en cuenta que l´ım*x→*0[sin(*x*)*/x*] = 1 la funci´on *g*(*x*) se define en este caso como:

*·*

*g*(*x*) = 

sin(*x*)

*x*

si *x ̸*= 0

 1 si *x* = 0

De forma an´aloga podr´ıamos operar con las dem´as ra´ıces de la funci´on (*x* =

*nπ*, siendo *n* un nu´mero entero no negativo.

A las ra´ıces de multiplicidad 1 se las llama ra´ıces simples. A las de multiplici- dad 2 se las designa como ra´ıces dobles, a las de multiplicidad 3 como ra´ıces triples, etc... Pero segu´n como se ha definido anteriormente las multiplicidad de una ra´ız, estas pueden ser en general nu´meros reales no necesariamente enteros.

No obstante, el concepto de multiplicidad de una ra´ız que nosotros conside- raremos se referir´a s´olo a valores enteros positivos. En este sentido siendo *m* un entero positivo y *f* (*x*) una funci´on de clase *Cm*(*I*) (donde *I* es el intervalo

en el que est´a definida), diremos que una ra´ız *x∗* de *f* (*x*) es de multiplicidad

*m* si se verifica que:

*f* (*x∗*) = *f′*(*x∗*) = *f ′′*(*x∗*) = *....* = *f* (*m−*1(*x∗*) = 0*, f* (*m*(*x∗*) *̸*= 0

**Observaci´on 14.2.** *Es importante hacer las siguientes observaciones:*

1. *Los m´etodos num´ericos que se abordan a continuaci´on, en general, per- der´an velocidad de convergencia cuando las ra´ıces a determinar tengan multiplicidad superior a* 1*. De aqu´ı la importancia de poder distinguir entre las ra´ıces simples y las ra´ıces mu´ltiples.*
2. *Si se ha determinado una ra´ız x∗ de multiplicidad m de la funci´on f* (*x*)

*y se desean determinar otras ra´ıces, puede calcularse la funci´on:*

*φ*(*x*) =

****

1

*−*

(*x x∗*)*m*

1

*· f* (*x*) *si x ̸*= *x∗*

**** l´ım *∗ m · f* (*x*) *si x* = *x∗*

*x→x∗* (*x − x* )

*y determinar las ra´ıces de φ*(*x*) *que ya no admitir´a a x∗ como ra´ız y sin embargo s´ı que admitir´a a las dem´as ra´ıces de f* (*x*) *como ra´ıces suyas.*

Como ya se sen˜al´o, los m´etodos iterativos que analizaremos en este apartado consistir´an en, a partir de un valor *x*0 dado, generar una sucesi´on *xi ∞i*=0 que converja hacia alguna soluci´on de la ecuaci´on.

*{ }*

Comencemos presentando un m´etodo muy intuitivo que nos permitir´a ir asen- tando ideas.

#### El m´etodo de biparticio´n

Consid´erese una ecuaci´on no lineal de la forma *f* (*x*) = 0 y supongamos que se conocen dos puntos *a* y *b* del dominio en el que est´a definida *f* (*x*) tales que: *a < b* y que en ellos *f* (*a*) tiene signo contrario a *f* (*b*), es decir que *f* (*a*) *f* (*b*) *<* 0. Obviamente estamos suponiendo que *f* (*a*) y *f* (*b*) son no nulos pues si alguno de ellos fuese nulo ya se tendr´ıa una soluci´on de la ecuaci´on. En estas condiciones si *f* (*x*) es una funci´on continua en [*a, b*]*,* por aplicaci´on del teorema de los valores intermedios, existir´a al menos un punto *x∗* de este intervalo en el que *f* (*x*) se anule. Por ello, junto a la hip´otesis de que *f* (*a*) *· f* (*b*) *<* 0 supondremos tambi´en que *f ∈ C*([*a, b*])*.*

*·*

**Observaci´on 14.3.** *El que exista ”al menos” un punto en el que se anule f* (*x*) *no quiere decir que s´olo haya uno. Contando cada ra´ız de f* (*x*) *tantas veces como sea su multiplicidad, si f* (*a*) *f* (*b*) *<* 0*, habr´a en general un nu´mero impar de ra´ıces de f* (*x*) *en el intervalo* [*a, b*]*. Y si f* (*a*) *f* (*b*) *fuese positivo o no hay ninguna ra´ız o habr´a un nu´mero par de ellas.*

*·*

*·*

Una primera aproximaci´on de este punto *x∗* puede ser el punto medio:

*a* + *b*

*x*1 = 2

Si *f* (*x*1) = 0 ya se tendr´ıa calculada una ra´ız. Pero por lo general, salvo que se tenga mucha suerte, se tendr´a que *f* (*x*1) *̸*= 0*.* Pero, al haber supuesto que la funci´on es continua, si *f* (*a*)*f* (*x*1) *<* 0 se podr´a afirmar que en el intervalo [*a, x*1] habr´a al menos una soluci´on de la ecuaci´on. Y si *f* (*a*)*f* (*x*1) *>* 0 se verificar´a que *f* (*x*1)*.f* (*b*) *<* 0 lo que nos indicar´ıa que en el intervalo [*x*1*, b*] existir´a al menos una ra´ız. Por tanto se habr´a definido as´ı un nuevo intervalo [*a*1*, b*1] en el que existir´a una soluci´on. A ´el puede aplic´arsele nuevamente el proceso anterior.

En general, partiendo de un intervalo [*aj, bj*] en el que *f* (*aj*)*.f* (*bj*) *<* 0 se denotar´a por *xj*+1 al punto medio del intervalo:

*xj*+1

= *aj* + *bj*

2

procediendo a continuaci´on de la forma siguiente:

1. Si *f* (*xj*+1) = 0 se habr´a obtenido una soluci´on de la ecuaci´on: el punto

*xj*+1*.*

1. Si *f* (*aj*) *· f* (*xj*+1) *<* 0 se denotar´a por: *aj*+1 = *aj* y por *bj*+1 = *xj*+1.
2. Si *f* (*aj*) *· f* (*xj*+1) *>* 0 se denotar´a por: *aj*+1 = *xj*+1 y por *bj*+1 = *bj*. Al nuevo intervalo [*aj*+1*, bj*+1] se le vuelve a aplicar el mismo proceso.

El problema que se nos puede plantear es: y si ningu´n valor *f* (*xj*) (j = 1, 2, ....) tiene la gracia de anularse ¿cu´ando se detiene el proceso iterativo?. La respuesta a esta pregunta depender´a de la precisi´on con la que se desee obtener la aproximaci´on de la soluci´on buscada. En efecto, si se parte de un intervalo [*a, b*] la longitud del mismo es *|b − a|.* En la primera iteraci´on se obtendr´a un intervalo [*a*1*, b*1] cuya longitud ser´a la mitad del anterior, es

decir *|b−a|* . A su vez, en la segunda iteraci´on se obtendr´a un intervalo [*a*2*, b*2] de longitud mitad que el anterior, es decir *|b−a| .* Siguiendo este proceso, en la

2

22

1. ´esima iteraci´on se obtendra´

un intervalo [*aj, bj*] cuya longitud ser´a *|b−a| .* Si

se tomara como aproximaci´on de la soluci´on *x∗* existente en dicho intervalo el punto medio *xj*+1 es evidente que

2*j*

*|x − x∗| ≤ |bj − aj|* = *|b − a| .*

*j*+1

2

2(*j*+1)

Por tanto, si se desease estar seguro de que la distancia de la aproximaci´on *xj*+1 a la ra´ız *x∗* fuese inferior a un determinado valor *ε* deber´ıan realizarse un nu´mero *j* de iteraciones tal que:

*|b − a| < ε* (*j* + 1)*.* log(2) *>* log 2(*j*+1)

*⇒*

(

*|b − a| ε*

*⇒ j >*

*|b−a|*

*ε*

log(2) *−* 1

log

**Observaci´on 14.4.** *Obs´ervese que el nu´mero de iteraciones anterior ase- gurar´ıa la precisi´on deseada pero que ello no impide el que algu´n valor xi calculado en alguna iteraci´on anterior pueda estar igual de cerca (o m´as) de la soluci´on buscada.*

Todo el proceso que hasta aqu´ı se acaba de describir podr´ıa sintetizarse en el siguiente algoritmo:

##### Algoritmo del m´etodo de biparticio´n:

Dada la ecuaci´on *f* (*x*) = 0, el indicador de precisi´on *ε* y dos puntos *a* y *b* en los que *f* (*a*) *· f* (*b*) *<* 0,

* 1. Estimar el menor nu´mero natural *N* tal que:

log *|b−a|*

*ε*

*N >* log(2) *−* 1

* 1. **Para** *j* = 1*,* **hasta** *j* = *N,* **con paso** 1, **hacer:**

*a* + *b*

*xj ←* 2

**Si** (*f* (*xj*) = 0) **entonces:**

tomar *xj* como ra´ız *x∗* y finalizar el proceso

##### si no:

**Si** (*f* (*xj*) *· f* (*a*) *<* 0) **entonces**:

*b ← xj*

##### si no:

*a ← xj*

##### fin condici´on. fin condicio´n.

**Fin bucle** en j.

*x∗ a* + *b* 2

*≈*

##### Fin del algoritmo.

El algoritmo anterior nos permitir´a encontrar, en las condiciones que garan- tizan el ´exito del proceso (esto es que *f* (*x*) sea continua), una de las ra´ıces existentes en [*a, b*]. ¿Pero qu´e condiciones nos podr´ıan garantizar que dicha ra´ız es la u´nica existente en [*a, b*]?. Una posibilidad para ello podr´ıa ser el que la funci´on *f* (*x*) fuese estrictamente mon´otona en el intervalo [*a, b*].

**Ejemplo 14.2.** *Estudiar la funci´on*

*f* (*x*) = *x* 1 cos2(*x*)

*−*

2

*en el intervalo I* = [ 2*,* 2] *determinando el nu´mero de soluciones de la ecua- ci´on f* (*x*) = 0 *en el intervalo dado. Calcular (al menos) una soluci´on de la ecuaci´on mediante el m´etodo de bisecci´on.*

*−*

Empezamos definiendo el intervalo *I*, la funci´on de manera simb´olica y calcu- lamos su derivada mediante el comando **diff**. Definimos tambi´en una funci´on que representa el eje de las abscisas.

%% Ejemplo 1 syms x

a=-2

b=2

f(x)=x-1/2\*(cos(x))^2 df=diff(f,x) eje(x)=0.\*x

Los cortes de la gr´afica de la soluci´on con el eje nos dar´an el nu´mero de soluciones en el intervalo. La derivada servir´a para argumentar la unicidad de soluciones. Representamos la funci´on y el eje

I=[a,b];

figure ezplot(f,I) hold on ezplot(eje,I) hold off

Verificamos las hip´otesis del m´etodo de bisecci´on. La funci´on es ciertamente continua (suma de una polin´omica con una trigonom´etrica continua) y hay un cambio de signo

p=f(a)\*f(b) pp=double(p)

Representamos la derivada de la funci´on. La derivada es siempre positiva, luego la funci´on es creciente y la soluci´on u´nica

figure ezplot(df,[a,b])

Finalmente calculamos la soluci´on con una llamad al algoritmo de bisecci´on

tol=1e-6 maxit=100

[zero,res,niter]=bisection(f,a,b,tol,maxit) residuo=abs(double(res))

Evaluamos el residuo para ver la calidad de la soluci´on

residuo=abs(double(res))

#### El m´etodo de aproximaciones sucesivas (punto fijo) .

El m´etodo de aproximaciones sucesivas (o del punto fijo) para determinar una soluci´on de la ecuaci´on no lineal *f* (*x*) = 0 consiste en reescribir la ecuaci´on *f* (*x*) = 0 en la forma *x* = *g*(*x*).

Advi´ertase que existen mu´ltiples posibilidades para transformar la ecuaci´on *f* (*x*) = 0 en otra del tipo *x* = *g*(*x*). Por ejemplo podr´ıa despejarse (de la forma que sea) *x* de la expresi´on de la ecuaci´on *f* (*x*) = 0. O podr´a sumarse la variable *x* en ambos lados de la ecuaci´on y designar por *g*(*x*) a (*f* (*x*) + *x*):

0 = *f* (*x*) *⇔ x* = *f* (*x*) + *x* = *g*(*x*) O, siendo *α ̸*= 0 podr´a realizarse el proceso:

0 = *f* (*x*) *⇔ x* = *αf* (*x*) + *x* = *g*(*x*) O bien, siendo *α ̸*= 0 y *β ̸*= 0 :

O bien:

0 = *f* (*x*) *⇔ αx* = *αx* + *βf* (*x*) *⇔ x* =

*αx* + *βf* (*x*)

= *g*(*x*)

*α*

0 = *f* (*x*) *⇔ xk* = *α · f* (*x*) + *xk ⇔ x* = *k*

✓

O por ejemplo:

*αf* (*x*) + *xk* = *g*(*x*)

0 = *f* (*x*) *⇔* cos(*x*) = *f* (*x*) + cos(*x*) *⇔ x* = arc cos(*f* (*x*) + cos(*x*))

Y muchas otras opciones ser´ıan posibles. No obstante, no debe confundirse el hecho de que sea posible considerar mu´ltiples formas de rescribir la ecuaci´on en la forma *x* = *g*(*x*) con el que sea indiferente la forma de hacerlo. En efecto la elecci´on de la funci´on *g*(*x*) no es independiente de la eficacia del m´etodo pudi´endose formar funciones *g*(*x*) que no est´en definidas en parte del intervalo en el que se vaya a trabajar, o que no sean aplicaciones, o que no sean continuas, o que .....no sean contracciones. Desde luego no tendr´an el mismo comportamiento unas que otras. Pero dejemos para un poco m´as adelante el an´alisis sobre cuales son las “buenas” funciones *g*(*x*) que nos interesan. En todo caso, una vez rescrita la ecuaci´on *f* (*x*) = 0 en la forma *x* = *g*(*x*) el m´etodo de aproximaciones sucesivas busca un punto fijo de la aplicaci´on *g*(*x*) mediante el **esquema de c´alculo** siguiente:

*Dado un valor x*0 *se genera la sucesi´on {xi*+1 = *g*(*xi*)*}∞i*=0.

**Observaci´on 14.5.** *Es necesario hacer las siguientes observaciones*

1. *Puesto que la ecuaci´on f* (*x*) = 0 *es equivalente a x* = *g*(*x*)*, en las con- diciones del teorema anterior, x∗ ser´a soluci´on en* [*a, b*] *de la ecuaci´on equivalente f* (*x*) = 0*, es decir f* (*x∗*) = 0*.*
2. *En otros t´erminos las buenas funciones g*(*x*) *que nos interesan son aquellas que sean contracciones sobre un determinado intervalo* [*a, b*] *en el que se buscar´a la u´nica soluci´on en ´el existente. Adem´as cuanto menor sea la constante de Lipschitz de la contracci´on g*(*x*) *m´as r´api- damente converger´a el m´etodo hacia la soluci´on.*
3. *Interpr´etese bien el teorema anterior. En ´el se asegura que bajo ciertas hip´otesis (el ser g*(*x*) *una contracci´on en el espacio m´etrico* ([*a, b*]*, df* )) *el m´etodo de aproximaciones sucesivas nos conduce a la u´nica soluci´on existente en* [*a, b*] *de la ecuaci´on f* (*x*) = 0*. Pero no se impide que el m´etodo funcione si no se verifican las hip´otesis. Simplemente no se asegura su buen funcionamiento.*

El demostrar que una aplicaci´on *g*(*x*) es una contracci´on mediante la deter- minaci´on de su constante de Lipschitz puede, en ciertas ocasiones, resultar algo laborioso. Por ello pueden contemplarse variantes m´as restrictivas (pero m´as f´acilmente aplicables en la pr´actica) del teorema anterior. Un ejemplo de ello es el siguiente teorema:

**Teorema 14.1.** *Si g*(*x*) *es una aplicaci´on de clase C*1([*a, b*])*, que toma va- lores en* [*a, b*] *y verificando la condici´on:*

*∃k <* 1 */ |g′*(*x*)*| ≤ k, ∀x ∈* [*a, b*]

*entonces la sucesi´on xi ∞i*=0 *generada, a partir de cualquier x*0 [*a, b*]*, con- verge hacia la u´nica soluci´on de la ecuaci´on x* = *g*(*x*) *en* [*a, b*]*.*

*{ } ∈*

**Observaci´on 14.6.** *Cuando en las aplicaciones se utilice este teorema para comprobar que la aplicaci´on considerada es una contracci´on se tomar´a como aproximaci´on de la constante de Lipschitz el valor k* = m´ax *g′*(*x*) *.*

*{| |}*

*x∈*[*a,b*]

**Observaci´on 14.7.** *Es importante notar que:*

1. *Si no se conoce el valor exacto de la constante de Lipschitz de la apli- caci´on puede estimarse de forma aproximada de diferentes formas. Por*

*ejemplo, tras cada iteraci´on del m´etodo podr´ıa obtenerse una aproxi- maci´on de dicha constante mediante:*

*k ≈ |g′*(*x* )*| ≈ |g*(*xi−*1) *− g*(*xi−*2)*|* = *|xi − xi−*1*|*

*i*

*|xi−*1 *− xi−*2*| |xi−*1 *− xi−*2*|*

1. *Una interpretaci´on gr´afica del m´etodo consiste simplemente en buscar la intersecci´on entre la bisectriz del primer cuadrante y la contracci´on g*(*x*) *mediante sucesivos ”escalones” comprendidos entre la gr´afica de g*(*x*) *y la bisectriz del primer cuadrante.*

En la pr´actica, en lugar de calcular a priori el nu´mero de iteraciones a rea- lizar se va estimando en cada iteraci´on la distancia del valor en ella hallado a la soluci´on exacta. Esta estimaci´on se realiza simplemente evaluando la diferencia entre las dos u´ltimas aproximaciones halladas que, cuando *g*(*x*) es una contracci´on, son un indicador de la cercan´ıa a la soluci´on exacta. M´as concretamente un algoritmo del m´etodo de aproximaciones sucesivas, en el que se parte de la ecuaci´on equivalente *x* = *g*(*x*) es el siguiente:

##### Algoritmo del m´etodo de aproximaciones sucesivas:

Dada la ecuaci´on *x* = *g*(*x*), el indicador de precisi´on *ε* , un valor m´aximo del nu´mero de iteraciones que se permiten realizar (*maxiter*) y un punto *x*0 con el que inicializar el proceso,

*tol* 2 *ε*

*← ·*

*iteracio*´*n* 0

*←*

**Mientras (** (*iteracio*´*n < maxiter*) **y** (*tol > ε*) **)**, **hacer:**

*x*1 *← g*(*x*0) *tol ← |x*1 *− x*0*|*

*iteracio*´*n ← iteracio*´*n* + 1

*x*0 *← x*1

**Fin bucle condicional**. **Si** (*tol < ε*) **entonces:**

tomar *x*1 como soluci´on

##### si no:

**Escribir** un mensaje de error en el proceso de c´alculo

**fin condicio´n**.

**Fin del algoritmo.**

#### El m´etodo de Newton-Raphson

Consid´erese la ecuaci´on *f* (*x*) = 0 en la que supongamos que *f* (*x*) es una fun- ci´on de clase *C*2([*a, b*])*.* Supongamos adem´as que la ecuaci´on anterior admite una soluci´on *x∗* en el intervalo [*a, b*]*.* Para cualquier otro valor *x*0 [*a, b*], denotando por *h* al valor tal que *x∗* = *x*0 + *h,* la expresi´on del desarrollo en serie de Taylor nos permitir´ıa escribir que:

*∈*

*\* ′ h*2 *′′*

0 = *f* (*x* ) = *f* (*x*0 + *h*) = *f* (*x*0) + *hf* (*x*0) + 2 *f* (*x*0 + *θh*)*, θ ∈* [0*,* 1]*.*

Si conocido *x*0 se fuese capaz de determinar *h* resolviendo la ecuaci´on:

*f* (*x*0) + *hf′*(*x*0) +

*h*2 *′′*

2

(*x*0 + *θh*) = 0

podr´ıa evaluarse *x∗* como *x∗* = *x*0 + *h.* Pero para resolver esta ecuaci´on primero deber´ıamos conocer el valor de *θ* (lo cual no es obvio) y una vez conocido resolver una ecuaci´on, en general, no lineal pues obs´ervese que *h* interviene en la expresi´on *f ′′* (*x*0 + *θh*)*.* Por tanto, salvo en situaciones muy particulares, no se ganar´ıa gran cosa reemplazando el problema de resolver *f* (*x*) = 0 por el de resolver

*f*

*F* (*h*) = *f* (*x*0) + *hf′*(*x*0) +

*h*2 *′′*

2

(*x*0 + *θh*) = 0*.*

El **m´etodo de Newton-Raphson** (o m´etodo de linealizaci´on de Newton) se sustenta en simplificar la expresi´on anterior linealiz´andola. Para ello con- sidera que si se est´a suficientemente cerca de la soluci´on (es decir, si *h* es

*f*

*f*

suficientemente pequen˜o) el t´ermino

*h*2 *′′*

2

(*x*0 + *θh*) podr´a despreciarse frente

a los otros t´erminos de la ecuaci´on. Por ello resuelve la ecuaci´on lineal:

*f* (*x*0) + *Hf′*(*x*0) = 0

de la que se obtiene que:

*f* (*x*0)

*−*

*H* =

*f′*(*x*0)

Obviamente, al ser diferente la ecuaci´on linealizada que la proporcionada por el desarrollo de Taylor, se tendr´a que *H ̸*= *h* y por tanto

*x∗* = *x*0 + *h ̸*= *x*1 = *x*0 + *H.*

De una forma intuitiva (que m´as adelante deberemos precisar cu´ando es co- rrecta) puede pensarse que aunque *x*1 sea diferente de *x∗* ser´a un valor m´as pr´oximo a *x∗* que *x*0 pues lo hemos obtenido “aproximando” el valor *h* que nos llevaba de *x*0 a *x∗.* Ello, al menos, ser´a as´ı cuando *h* sea suficientemente pequen˜o, es decir cuando *x*0 sea suficientemente pr´oximo a *x∗.* Con ello el m´etodo de Newton-Raphson propone repetir este proceso de forma recursiva hasta estar lo suficientemente cercanos a la soluci´on buscada. M´as concreta- mente el **m´etodo de Newton-Raphson** consiste en:

*−*

*Dado un valor x*0*, generar la sucesi´on*  *x*

*i*+1

= *xi*

*f* (*xi*) *∞*

*.*

*f′*(*x* )

*i i*=0

Sobre este m´etodo, en primer lugar, puede observarse que si denotamos por:

*f* (*x*)

*g*(*x*) = *x − f′*(*x*)

estamos en presencia de un caso particular del m´etodo de aproximaciones sucesivas antes contemplado.

**Observaci´on 14.8.** *Obs´ervese que la aplicaci´on del m´etodo de Newton- Raphson exige que los valores de f′*(*xi*) *no se anulen.*

Al igual que se hizo con el m´etodo de aproximaciones sucesivas, las condicio- nes que garantizan la convergencia global del m´etodo pueden ser sustituidas por otras que garantizan su convergencia local (esto es si el punto *x*0 con el que se inicializa el m´etodo es suficientemente cercano a la soluci´on buscada). Con ello se pueden rebajar las “exigencias” sobre la funci´on que garanticen el correcto funcionamiento del m´etodo.

Ello nos permite escribir un algoritmo recogiendo el m´etodo de Newton- Raphson como el que sigue:

##### Algoritmo del m´etodo de Newton-Raphson:

Dada la ecuaci´on *f* (*x*) = 0, los indicadores de precisi´on *ε* y *δ*, un valor m´aximo del nu´mero de iteraciones que se permiten realizar (*maxiter*) y un punto *x*0 con el que inicializar el proceso,

*tolx* 2 *ε*

*← ·*

*tolf* 2 *δ*

*← ·*

*iteracio*´*n* 0

*←*

**Mientras (** (*iteracio*´*n < maxiter*) **y** ((*tolx > ε*) **o** (*tolf > δ*) **)**, **hacer: Si** (*f′*(*x*0) = 0) **entonces:**

**Escribir** mensaje de error (derivada nula) y

finalizar el proceso

##### si no:

*x*1 *← x*0

*f* (*x*0)

*— f′*(*x*0)

*tolx ← |x*1 *− x*0*| tolf ← |f* (*x*1)*|*

*iteracio*´*n ← iteracio*´*n* + 1

*x*0 *← x*1

##### fin condicio´n. Fin bucle condicional.

**Si** ((*tolx < ε*) **y** (*tolf < δ*) ) **entonces:**

tomar *x*1 como soluci´on

##### si no:

**Escribir** un mensaje de error en el proceso de c´alculo

##### fin condici´on.

**Fin del algoritmo.**

**Observaci´on 14.9.** *En muchas ocasiones la diferencia entre dos valores consecutivos de las aproximaciones obtenidas se relativizan para expresarlas porcentualmente. Para ello en el algoritmo anterior puede sustituirse la l´ınea:*

*tolx ← |x*1 *− x*0*|*

*por otras que sean de la forma:*

**Si** (*x*1 *̸*= 0 ) **entonces:**

*tolx ← |x*1*−x*0*| ·* 100

##### si no:

*|x*1*|*

*tolx ← |x*1 *− x*0*|*

**fin condici´on**.

#### Variantes del m´etodo de Newton-Raphson: m´eto- dos de la secante y de “regula falsi”

El m´etodo de Newton que se acaba de exponer es un m´etodo que, gene- ralmente, tiene un buen comportamiento en la resoluci´on de muy diferentes

ecuaciones no lineales. Su principal inconveniente pr´actico consiste en la ne- cesidad de calcular los valores de *f′*(*xi*) en cada iteraci´on. Por ello existen variantes del m´etodo de Newton que tratan de obviar este c´alculo aproxi- mando el valor de *f′*(*xi*). Entre ellos, los m´as populares son los denominados **m´etodo de la secante** y **m´etodo de regula falsi**.

#### M´etodo de la secante.

Este m´etodo aproxima el valor de *f′*(*xi*) mediante:

*f′*(*x* ) *≈ f* (*xi*) *− f* (*xi−*1)

*i*

*xi − x*

*i−*1

con lo que el esquema iterativo del m´etodo de Newton-Raphson se ve modi- ficado a:

*x* = *x*

*— f* (*xi*) = *xi−*1 *· f* (*xi*) *− xi · f* (*xi−*1)

*i*+1

*i f* (*xi*)*−f* (*xi−*1)

*xi−xi−*1

*f* (*xi*) *− f* (*x*

*i−*1)

Obs´ervese que para aplicar el m´etodo se necesitan dos valores *x*0 y *x*1 con los que inicializar el proceso. Por ello en el m´etodo de la secante la primera iteraci´on se realiza mediante el m´etodo de Newton sigui´endose el siguiente proceso:

*Dado x*0

*x*1 *← x*0

*f* (*x*0)

*— f′*(*x*0)

*x ← xi−*1 *· f* (*xi*) *− xi · f* (*xi−*1)

*i* = 1*,* 2*, .....*

*i*+1

*f* (*xi*) *− f* (*x*

*i−*1)

**Observaci´on 14.10.** *El m´etodo de la secante toma su nombre del hecho de que gr´aficamente se puede interpretar el m´etodo de forma similar al de Newton pero sustituyendo la recta tangente a la curva por el punto* (*xi, f* (*xi*)) *por la recta secante que pasa por los puntos* (*xi−*1*, f* (*xi−*1)) *y* (*xi, f* (*xi*))*.*

#### El m´etodo de “Regula Falsi”.

Este m´etodo es una combinaci´on del m´etodo de bipartici´on y el m´etodo de la secante. En ´el se considera una ecuaci´on *f* (*x*) = 0 y un intervalo [*a, b*] en el que *f* (*x*) sea continua y adem´as se verifique que *f* (*a*) *f* (*b*) *<* 0*.* Con ello, segu´n se indic´o al analizar el m´etodo de bipartici´on se puede estar seguro de que en [*a, b*] existe al menos una ra´ız. Tras ello se denomina *x*1 al punto de corte con el eje de abscisas de la recta secante que pasa por los puntos (*a, f* (*a*))*,* (*b, f* (*b*)), es decir, que ser´a el punto:

*·*

*x* = *a · f* (*b*) *− b · f* (*a*)

1

*f* (*b*) *− f* (*a*)

Si *f* (*x*1)*f* (*a*) *<* 0 se puede asegurar que en el intervalo (*a, x*1) existir´a una soluci´on de la ecuaci´on. En el caso de que *f* (*x*1)*f* (*a*) *>* 0 se puede afirmar lo mismo para el intervalo (*x*1*, b*). Y en el caso de que *f* (*x*1) = 0 se habr´a determinado ya la soluci´on. En todo caso o se tiene la soluci´on de la ecuaci´on o se dispone de un intervalo m´as pequen˜o en el que volver a repetir el proceso. Esta forma de proceder es repetida las veces que sea necesario hasta encontrar un intervalo en el que exista una soluci´on y con una longitud inferior a la precisi´on deseada. M´as concretamente el algoritmo del m´etodo ser´a:

##### Algoritmo del m´etodo de Regula Falsi:

Dada la ecuaci´on *f* (*x*) = 0, el indicador de precisi´on *ε* y dos puntos *a* y *b* en los que *f* (*a*) *· f* (*b*) *<* 0,

**Mientras** *|b − a| > ε,* **hacer:**

*x a · f* (*b*) *− b · f* (*a*)

*←*

*f* (*b*) *− f* (*a*)

**Si** (*f* (*x*) = 0) **entonces:**

tomar *x* como ra´ız *x∗* y finalizar el proceso

##### si no:

**Si** (*f* (*x*) *f* (*a*) *>* 0) **entonces**:

*·*

*b x*

*←*

##### si no:

*a x*

*←*

##### fin condici´on. fin condicio´n.

**Fin bucle condicional**.

*x∗ x*

*←*

**Fin del algoritmo.**

#### Aceleraci´on de la convergencia de los m´etodos iterativos: m´etodo ∆2 de Aitken

Cuando el m´etodo de resoluci´on de ecuaciones no lineales que se est´e em- pleando para resolver una ecuaci´on no lineal no posea convergencia, al menos, cuadr´atica, puede utilizarse la estrategia conocida con el nombre de m´etodo delta-dos (∆2) de A. C. Aitken (matem´atico del siglo XX) para mejorar su velocidad de convergencia.

#### Ejercicios

Durante la sesi´on resolveremos algunos ejercicios propuestos.

**Ejercicio 14.1.** *Sea dada la funci´on*

*f* (*x*) = *−*2*x*3 + 5*x* + 2

1. *Aplicar el algoritmo de bisecci´on para calcular todas las soluciones de la ecuaci´on f* (*x*) = 0 *trabajando con una tolerancia de* 10*−*3 *y un nu´mero m´aximo de* 100 *iteraciones.*
2. *Aplicar el M´etodo de Newton para resolver la misma ecuaci´on con los mismos valores de tolerancia y nu´mero m´aximo de iteraciones.*

**Ejercicio 14.2.** *Sea dada la funci´on*

*f* (*x*) = *x*3 + 4*x*2 *−* 10

1. *Aplicar el algoritmo de la bisecci´on para calcular todas las soluciones de la ecuaci´on f* (*x*) = 0 *trabajando con una tolerancia de* 10*−*3 *y un nu´mero m´aximo de 100 iteraciones.*
2. *Aplicar el m´etodo de Newton para resolver la misma ecuaci´on con los mismos valores de tolerancia y nu´mero m´aximo de iteraciones.*

**Ejercicio 14.3.** *Sea dada la funci´on*

*f* (*x*) = *x*4 *−* 1 *− e−xx*2

1. *Considerando el esquema num´erico asociado al M´etodo de Newton co- mo un esquema de punto fijo, definir la funci´on ϕN* (*x*) *asociada al m´etodo y utilizar el algoritmo de punto fijo para calcular las soluciones de la ecuaci´on f* (*x*) = 0 *trabajando con una tolerancia de* 10*−*6 *y un nu´mero m´aximo de* 1000 *iteraciones. Utilizar el algoritmo Aitken para calcular las mismas soluciones mediante los mismos par´ametros de en- trada (semillas, tolerancia y nu´mero m´aximo de iteraciones). Comparar los resultados obtenidos en t´erminos de velocidad de convergencia.*
2. *Considerar la funci´on*

*ϕ*1(*x*) = (1 + *e−xx*2)1*/*4

*que define el esquema de punto fijo x* = *ϕ*1(*x*)*. Utilizar los algoritmos de puntofijo y de Aitken para resolver la ecuaci´on f* (*x*) = 0*. Utilizar los mismos par´ametros de entrada (semillas, tolerancia y nu´mero m´aximo de iteraciones) del apartado anterior. Comparar los resultados.*

1. *Considerar la funci´on*

*ϕ*2(*x*) = *x* + *f* (*x*)

*que define el esquema de punto fijo x* = *ϕ*2(*x*)*. Utilizar los algoritmos de puntofijo y de Aitken para resolver la ecuaci´on f* (*x*) = 0*. Utilizar los mismos par´ametros de entrada (semillas, tolerancia y nu´mero m´aximo de iteraciones) del apartado anterior. Comparar los resultados.*

**Ejercicio 14.4.** *Sea la funci´on*

*f* (*x*) = *ex −* 5*x*2

1. *Considerando el esquema num´erico asociado al M´etodo de Newton co- mo un esquema de punto fijo, definir la funci´on ϕN* (*x*) *asociada al m´etodo y utilizar el algoritmo de punto fijo para calcular las soluciones de la ecuaci´on f* (*x*) = 0 *trabajando con una tolerancia de* 10*−*6 *y un nu´mero m´aximo de 1000 iteraciones.*
2. *Utilizar el algoritmo de Aitken para calcular las mismas soluciones con los mismos valores de tolerancia y nu´mero m´aximo de iteraciones. Uti- liza las semillas utilizadas en el apartado anterior. Compara en t´ermi- nos de velocidad de convergencia los resultados obtenidos en el apartado anterior.*
3. *Considerar la funci´on*

*ϕ*1(*x*) = ln(3*x*2)

*que define el esquema de punto fijo x* = *ϕ*(*x*)*. Utilizar los algoritmos de punto fijo y de Aitken para resolver la ecuaci´on f* (*x*) = 0 *resuelta en los apartados anteriores. Compara los resultados.*

1. *Considerar la funci´on*

*ϕ*1(*x*) = ✓*ex/*3

*que define el esquema de punto fijo x* = *ϕ*(*x*)*. Utilizar los algoritmos de punto fijo y de Aitken para resolver la ecuaci´on f* (*x*) = 0 *resuelta en los apartados anteriores. Compara los resultados.*

## Ap´endice: Optimizaci´on Convexa

En esta clase consideraremos problemas de optimizaci´on convexa con res- tricciones. Empezaremos con las definiciones y conceptos b´asicos para llegar al criterio de optimalidad (condiciones necesarias y suficientes) dado por las Condiciones de Karush-Kuhn-Tucker (KKT) para resolver problemas de mi- nimizaci´on con restricciones para funciones convexas y diferenciables.

#### Optimizaci´on Convexa

Empezamos definiendo la forma general de un problema de optimizaci´on. Luego pasaremos a considerar los problemas de optimizaci´on convexa.

Un problema de Optimizaci´on se expresa en la *Forma Estandar*

donde

m´ın *f*0(**x**) (58)

**x**

s.a *fi*(**x**) *≤* 0*, i* = 1*...m, hi*(**x**) = 0*, i* = 1*...p*

**x** *∈* R*n*

es la **variable a optimizar**, un vector en general y la funci´on

*f*0 : R*n →* R

es la **funci´on objetivo** o funci´on de coste. Las desigualdades

*fi*(**x**) *≤* 0*, i* = 1*...m*

son las *m* **restricciones de desigualdad** y las igualdades

*hi*(**x**) = 0*, i* = 1*...p*

son las *p* **restricciones de igualdad**. Si no existen restricciones (*m* = *p* = 0) definiremos el problema de **minimizaci´on sin restricciones**.

**Definici´on 15.1.** *Definimos el* **dominio** *del problema de* **minimizaci´on con restricciones** ([58](#_bookmark204)) *como el conjunto de punto en donde la funci´on ob- jetivo y todas las funciones restricci´on est´an definidas*

*D*

*m*

*.*

*D* =

*i*=0

*dom fi*

*p*

*∩*

*i*=1

*dom hi*

No todos los puntos del dominio *D* son admisibles.

**Definici´on 15.2.** *Un punto del dominio,* **x** *es* **admisible** *si satisface todas las restricciones. El problema* ([58](#_bookmark204)) *es admisible si existe al menos un punto admisible, en caso contrario se define de no admisible. El conjunto de todos los puntos admisibles se llama* **conjunto admisible** *o conjunto de restricciones y se define por*

*∈ D*

*A* = *{***x** *∈ D /* **x** *es admisible} ⊂ D*

El ´optimo del problema con restricciones se tendr´a que buscar en el conjunto admisible.

**Definici´on 15.3.** *El* **valor o´ptimo** *p∗ del problema* ([58](#_bookmark204)) *se define por*

= ´ınf*{f*0(**x**)*, | fi*(**x**) *≤* 0*, i* = 1*...m, hi*(**x**) = 0*, i* = 1*...p} ∈* R

*p∗ .*

*siendo el* **punto o´ptimo**

**x***∗* = arg m´ın *f*0(**x**) R*n*

*∈*

**x***∈*R*n*

*el punto (vector) en donde se alcanza. Diremos por tanto que* **x***∗ es un* **punto**

**´optimo** *si* **x***∗ es admisible y f*0(**x***∗*) = *p∗.*

Los puntos o´ptimos pueden ser Globales o Locales. En optimizaci´on convexa todos los o´ptimos locales son globales. Se permite que *p∗* tome los valores

. Si el problema no es admisible tendremos *p∗* = (siguiendo la con- venci´on estandar que el ´ınfimo de un conjunto vac´ıo es ). Si existen puntos admisibles **x***k* tales que *f*0(**x***k*) cuando *k* entonces *p∗* = y diremos que el problema ([58](#_bookmark204)) **no es inferiormente acotado**.

*→ −∞ → ∞ −∞*

*∞*

*±∞ ∞*

**Definici´on 15.4.** *Si* **x** *es admisible y fi*(**x**) = 0 *diremos que la i-´esima restricci´on de desigualdad es* **activa***. Si fi*(**x**) *<* 0 *diremos que la restricci´on es* **inactiva***.*

Se define finalmente el **Problema de Admisibilidad** como el problema de determinar si todas las restricciones son consistentes:

Determinar **x** subjeto a *fi*(**x**) *≤* 0, *i* = 1*...m* y *hi*(**x**) = 0 , *i* = 1*...p*.

**Definici´on 15.5.** *El problema* ([58](#_bookmark204)) *se dice en* **standard form** *si adoptamos la convenci´on de que los t´erminos derechos de las restricciones de desigualdad e igualdad son cero.*

Observa que esto siempre se puede hacer. Si tenemos restricciones de igualdad del tipo *gi*(**x**) = *g*˜*i*(**x**) definimos *hi*(**x**) = *gi*(**x**) *− g*˜*i*(**x**) y de manera similar si tenemos desigualdades de la forma *f*˜*i*(**x**) *≥* 0 las escribimos en la forma *fi*(**x**) = *−f*˜*i*(**x**) *≤* 0.

**Ejercicio 15.1.** *Se considera el problema de minimizaci´on con restricciones*

m´ın *f*0(**x**)*, s.a li ≤ xi ≤ ui, i* = 1*...n,*

*en donde este tipo de restricciones se conoce como* **box constrained***. Expre- sar el problema en forma estandar. Comp´arese la respuesta con el ejemplo*

*4.2 de Boyd.*

Dos problemas del tipo ([58](#_bookmark204)) se dicen **equivalentes** si tienen el mismo conjun- to de soluciones. Los problema de Optimizaci´on pueden tener una, ninguna, varias o infinitas soluciones, locales y/o globales, siendo ´estas m´ınimos o m´aximos de la funci´on objetivo. Un caso particularmente importante es el de la **Optimizaci´on Convexa** que pasamos a definir.

##### Optimizaci´on Convexa

**Definici´on 15.6.** *Un problema de Optimizaci´on Convexa es un problema de Minimizaci´on con Restricciones en la forma Estandar*

m´ın *f*0(**x**) (59)

*s.a fi*(**x**) *≤* 0*, i* = 1*...m, aT* **x** = *bi, i* = 1*...p*

*i*

*en donde las funciones (objetivo y restricciones de desigualdad)*

*f*0(**x**)*,* *, fm*(**x**)

*son funciones convexas* [17](#_bookmark207) *y las restricciones de igualdad vienen dadas por funciones afines*

*hi*(**x**) = *aT* **x** *− bi*

*i*

*es decir, rectas, planos o hyperplanos que no pasan por el origen si bi ̸*= 0*.*

Esta formulaci´on Estandar de un problema de Optimizaci´on Convexa se en- cuentra en la secci´on 4.2.1 del libro de S. Boyd. Es importante observar que el conjunto de puntos admisibles de un problema de optimizaci´on convexa es convexo. Esto tiene la importante implicaci´on de que

**Teorema 15.1.** *En un problema de optimizaci´on convexa todos los ´optimos son globales.*

##### Criterio de Optimalidad

El Criterio de Optimalidad (de primer orden) para funciones (objetivos) *f*0 convexas y diferenciables es como sigue. Sabemos[18](#_bookmark208) que dada una funci´on diferenciable *f* existe su gradiente en todo punto del dominio, *dom*(*f* ) y una funci´on *f* diferenciable es convexa si y s´olo si *dom*(*f* ) es convexo y

*f* (**y**) *≥ f* (**x**) + *∇f* (**x**)*T* (**y** *−* **x**)*,* **x***,* **y** *∈ dom*(*f* )

Gr´aficamente quiere decir que la funci´on est´a por encima del plano tangen- te[19](#_bookmark209). La funci´on afin

*f* (**x**) + *∇f* (**x**)*T* (**y** *−* **x**)

17v´ease la definici´on de convexidad en las notas del curso de c´alculo multi-variable o en la secci´on 1.3 del libro de S. Boyd

18ver secci´on 1.3.1 del Boyd

19V´ease por ejemplo la figura 8 de la II clase para un ejemplo gr´afico de funci´on convexa y comp´arese con la figura 9 en donde la funci´on no es convexa.

representa la aproximaci´on de Taylor de primer orden de la *f* y por la des- igualdad ([60](#_bookmark210)) representa un sub-estimador global de la funci´on ya que se encuentra por debajo de su gr´afica. Si **x** (es decir si estamos resolviendo un **problema con restricciones**) diremos que **x** es **´optimo** si y s´olo si:

*∈ A*

*∇f* (**x**) (**y** *−* **x**) *≥* 0*, ∀* **y** *∈ A* (60) ya que la condici´on ([60](#_bookmark210)) asegura, al ser la funci´on convexa

*T*

*f* (**y**) *≥ f* (**x**) + *∇f* (**x**)*T* (**y** *−* **x**) *≥ f* (**x**)*, ∀* **y** *∈ A*

Como consecuencia del criterio ([60](#_bookmark210)) si

*∇f* (**x**) = 0

entonces

*f* (**y**) *≥ f* (**x**)*, ∀* **y** *∈ A*

luego **x** es un m´ınimo global. Si el problema de minimizaci´on no tiene res- tricciones (*m* + *p* = 0) entonces la condici´on ([60](#_bookmark210)) se reduce a la conocida **condici´on necesaria y suficiente** para funciones convexas (la demostra- ci´on se encuentra en Boyd pg 140).

*∇f* (**x**) = 0 (61)

**Ejemplo 15.1.** *Aplicar el criterio de optimalidad para el* **problema de minimizaci´on sin restricciones** *de la forma cuadr´atica*

m´ın *f* (**x**) = m´ın (1 **x***T P* **x** + **q***T* **x** + *r*

0

**x x** 2

*siendo P ∈ Sn*

+

*(cuadrada, sim´etrica, semidefinida positiva).*

Respuesta La funci´on *f*0(**x**) es convexa. La condici´on necesaria y suficiente para funciones convexas ([61](#_bookmark211)) es

*∇f*0(**x**) = *P* **x** + **q** = 0

Se trata de un sistema lineal que puede tener una, ninguna o infinitas solu- ciones.

1. Si **q** *∈/ Rg*(*P* ) el sistema no tiene soluciones y la funci´on *f*0(*x*) no es

inferiormente acotada.

1. Si *P >* 0 (definida positiva) entonces la funci´on *f*0(**x**) es estrictamente convexa y existe una u´nica soluci´on:

**x***∗* = *−P−*1**q***.*

1. Si *P* es singular pero **q** *Rg*(*P* ) entonces el conjunto de puntos o´ptimos

*∈*

*X∗* es

*X∗* = *−P†***q** + *N* (*P* )

siendo *P†* la pseudo-Inversa de Moore-Penrose y (*P* ) es el nu´cleo de la aplicaci´on lineal asociada a *P* .

*N*

Consideremos ahora el caso de un problema de minimizaci´on con s´olo res- tricciones de igualdad.

**Ejemplo 15.2.** *Aplicar el criterio de optimalidad para el problema de mini- mizaci´on con restricciones de igualdad*

m´ın *f*0(**x**)*, s.a A***x** = **b**

**x**

Observamos que el conjunto admisible es af´ın (al ser definido por las solu- ciones de un sistema no homog´eneo) y, supondremos, no vac´ıo ya que de lo contrario el problema no ser´ıa admisible. La condici´on de optimalidad para **x** *∈ A* viene dada por el criterio ([60](#_bookmark210)):

*∇f*0(**x**)*T* (**y** *−* **x**) *≥* 0*, ∀* **y** */ A***y** = **b**

Puesto que **x** es admisible se tiene que **y** admisible tiene que ser de la forma

**y** = **x** + **v***, v ∈ N* (*A*)

Luego **y** *−* **x** = **v** y la condici´on ([60](#_bookmark210)) se puede escribir en forma

*∇f*0(**x**)*T* **v** *≥* 0*, ∀* **v** *∈ N* (*A*)

Si una funci´on lineal es no negativa en un subespacio vectorial entonces es necesariamente nula en el subespacio. La funci´on

*∇f*0(**x**) **v** = *⟨∇f*0(**x**)*,* **v***⟩*

*T*

es lineal y es no negativa en el nu´cleo, que es un sub-espacio luego

*∇f*0(**x**) **v** = 0*, ∀* **v** *∈ N* (*A*) lo que implica la ortogonalidad:

*T*

*∇f*0(**x**) *⊥ N* (*A*)*.*

Utilizando la propiedad algebraica

*N* (*A*)*⊥* = *R*(*AT* )

la condici´on anterior se puede escribir en la forma

*∇f*0(**x**) *∈ R*(*A* ) es decir: existe *ν*¯ *∈* R*p* tal que

*T*

*∇f*0(**x**) + *AT ν*¯ = **0** (62)

Junto con el requerimiento que *A***x** = **b** (**x** admisible) obtenemos la cl´asica

##### condici´on de optimalidad para multiplicadores de Lagrange.

Finalmente vamos a describir el uso de un algoritmo general (caja negra) para la resoluci´on de un problema de minimizaci´on con restricciones. Esta incluido en Matlab y se llama **fmincon.m**

**Ejemplo 15.3.** *Sea* **x***T* = (*x, y, z*)*T y dada la funci´on*

*f* (**x**) = *f* (*x, y, z*) = *−xyz*

*Se trata de calcular la soluci´on del problema de minimizaci´on*

m´ın *f* (**x**) *s.a* 0 *x* + 2*y* + 2*z* 72

*≤ ≤*

**x***∈*R3

*arrancando con semilla* **x**0 = (10*,* 10*,* 10)*.* Empezamos escribiendo las restricciones en la forma

*−x −* 2*y −* 2*z ≤* 0*, x* + 2*y* + 2*z ≤* 72

Al ser lineales se trata de un sistema de desigualdades del tipo

*A*2*,*3**x**3*,*1 *≤* **b**2*,*1

siendo

*A* = *−*1 *−*2 *−*2

1 2 2

2*,*3

*,* **b** = 0 72

2*,*1

function f = myfun(x)

f = -x(1) \* x(2) \* x(3);

Tras definir el sistema se utiliza

x0 = [10; 10; 10]; % semilla inicial [x,fval] = fmincon(@myfun,x0,A,b)

La soluci´on proporcionada es **x***∗* = (24*,* 12*,* 12)*T* y el valor (m´ınimo) de la energ´ıa es *p∗* = *−*3*.*4560*e* + 03.

**Ejercicio 15.2.** *Resolver el problema anterior de manera anal´ıtica utilizando variables slacks e imponiendo la condici´on de los multiplicadores de Lagrange,* ([62](#_bookmark212))*.*

##### Slack Variables

Introducimos el concepto de **slack variables**, es decir variables de holgura en castellano. La idea es que siempre podemos sustituir las restricciones de desigualdad por unas igualdades y unas restricciones de no negatividad. En concreto se tiene que

*fi*(**x**) *≤* 0 si y s´olo si existen *si*, *i* = 1*..m* tal que

*fi*(**x**) + *si* = 0*, i* = 1*..m.*

Escribimos entonces el problema (en forma estandar) ([58](#_bookmark204))

m´ın *f*0(**x**)*,*

s.a *fi*(**x**) *≤* 0*, i* = 1*...m, hi*(**x**) = 0*, i* = 1*...p*

en la forma

m´ın *f*0(**x**) (63)

s.a *si ≥* 0*, fi*(**x**) + *si* = 0*, i* = 1*...m, hi*(**x**) = 0*, i* = 1*...p*

donde **x** R*n*, **s** R*m* que tiene ahora *n* + *m* variables, *m* restricciones de desigualdad de tipo no negatividad (lower bound) para las slacks variables y *m* + *p* restricciones de igualdad. Las dos formulaciones son equivalentes: Si (**x***,* **s**) es admisible para el problema ([63](#_bookmark214)) entonces **x** es admisible para el problema ([58](#_bookmark204)) (y al rev´es). Lo mismo se verifica que (**x***,* **s**) es o´ptimo para el problema ([63](#_bookmark214)) si y s´olo si **x** es ´optimo para el problema ([58](#_bookmark204)) (y al rev´es). Observa que la introducci´on de variables de holgura preserva la convexidad s´olo si las restricciones son afines:

*∈ ∈*

*hi*(**x**) = *aT* **x** = *bi, i* = 1*...p.*

*i*

##### Epigraph Formulation

La formulaci´on del Ep´ıgrafe del problema de optimizaci´on convexa ([59](#_bookmark205)) es

m´ın *t* (64)

s.a *f*0(**x**) *− t ≤* 0*, fi*(**x**) *≤* 0*, i* = 1*...m, aT* **x** = *bi, i* = 1*...p*

*i*

El objetivo es ahora lineal (luego convexo) y la nueva restricci´on *f*0(**x**) *t* es tambi´en convexa en las variables (**x***, t*) luego la formulaci´on del Ep´ıgrafe del problema es convexa. Gracias a esta formulaci´on se puede afirmar que un objetivo lineal es **universal** en optimizaci´on convexa ya que cualquier problema se puede reducir en uno con objetivo lineal.

*−*

##### Dualidad

En esta secci´on se introducen los conceptos fundamentales de la teor´ıa de la Dualidad lo que permitir´a desarrollar algoritmos eficientes para la resoluci´on de problemas de minimizaci´on convexa. Como casos particulares se describe la aplicaci´on de la teor´ıa a la Programaci´on Lineal y Cuadr´atica. Empezamos definiendo la **Funci´on Dual de Lagrange**. Se considera un problema de optimizaci´on en la Forma Estandar ([58](#_bookmark204)):

m´ın *f*0(**x**)

**x**

s.a *fi*(**x**) *≤* 0*, i* = 1*...m, hi*(**x**) = 0*, i* = 1*...p*

Se pasa a continuaci´on a escribir el problema con restricciones en un problema sin restricciones (exceptuando unascondiciones de no negatividad) mediante la funci´on lagrangiana

**Definici´on 15.7.** *La* **Lagrangiana** *asociada al problema es una funci´on (campo escalar)*

*definida por*

*L* : R*n ×* R*m ×* R*p →* R

*m p*

*L*(**x***, λ, ν*¯) = *f*0(**x**) + *λifi*(**x**) + *νihi*(**x**) (65)

*i*=1

*i*=1

*donde los λi ≥* 0 *son los* **multiplicadores de Lagrange** *asociados a las restricciones de desigualdad y los νi son los* **multiplicadores de Lagrange** *asociados a las restricciones de igualdad y los vectores λ* R*m, ν*¯ R*p son las* **variables duales del problema***.*

En otros textos se suele llamar multiplicadores de Lagrange a los *νi* que aparecen en las restricciones de igualdad y variables duales a los *λi* que aparecen en las desigualdades.

*∈ ∈*

**Definici´on 15.8.** *Definimos la* **funci´on Dual de Lagrange**

*g* : R*m ×* R*p →* R

*como el ´ınfimo en* **x** *∈* R*n de la funci´on lagrangiana*

*g*(*λ, ν*¯) = ´ınf

**x***∈*R**n**

*L*(**x***, λ, ν*¯) = ´ınf

**x***∈*R**n**

*f*0(**x**) +

*m*

*i*=1

*λifi*(**x**) +

*p*

*i*=1

*νihi*(**x**)

(66)

Observa que si la lagrangiana no est´a acotada inferiormente la funci´on dual toma el valor *−∞*.

**Definici´on 15.9.** *Dado el par de variables duales*

(*λ, ν*¯) *∈* R*m ×* R*p*

*diremos que* (*λ, ν*¯) *es admisible para la funci´on dual de Lagrange si*

*g*(*λ, ν*¯) *> −∞*

**Teorema 15.2.** *La funci´on dual g*(*λ, ν*¯) *es concava aunque el problema ini- cial no sea convexo.*

Por el teorema tiene sentido maximizarla. Sea *p∗* el valor o´ptimo del problema primal. Tenemos entonces la acotaci´on inferior del valor ´optimo:

*g*(*λ, ν*¯) *≤ p∗, ∀λ ≥* 0*, ∀ν*¯

La funci´on dual de Lagrange est´a relacionada con la **funci´on conjugada**

que se introduce a continuaci´on:

**Definici´on 15.10.** *Dada una funci´on*

*f* : R*n →* R

*definimos la funci´on conjugada f∗,*

*f* : R*n →* R

*como*

*T*

*f∗*(**y**) = sup

**x***∈dom*(*f* )

*{***y x** *− f* (**x**)*}* (67)

Veamos como aplicar esta definici´on. Consideremos un problema de minimi- zaci´on con restricciones de desigualdad e igualdad en la forma

m´ın *f*0(**x**)*,* s.a. *A***x b***, C***x** = **d**

*≤*

**x**

Usando la conjugada de la funci´on objetivo podemos calcular la funci´on dual asociada al problema

*g*(*λ, ν*¯) = ´ınf *L*(**x***, λ, ν*¯) = ´ınf *f*0(**x**) +

*m*

*λifi*(**x**) +

*νihi*(**x**) =

**x x**

*i*=1

*p*

*i*=1

= ´ınf *f*0(**x**) + *λT* (*A***x** *−* **b**) + *ν*¯*T* (*C***x** *−* **d**) =

**x**

= *−***b***T λ −* **d***T ν*¯ + ´ınf *f*0(**x**) + (*AT λ* + *CT ν*¯)*T* **x** =

**x**

= *−***b***T λ −* **d***T ν*¯ *− f*0*∗*(*−AT λ − CT ν*¯) en donde hemos utilizado la propiedad

´ınf(*f* ) = *−* sup(*−f* ) y la definici´on de la funci´on conjugada

**x x**

´ınf *f*0(**x**) + (*AT λ* + *CT ν*¯)*T* **x** = *−* sup *−f*0(**x**) *−* (*AT λ* + *CT ν*¯)*T* **x** =

= *−* sup *−*(*AT λ* + *CT ν*¯)*T* **x** *− f*0(**x**) = *−f*0*∗*(*−AT λ − CT ν*¯)

**x**

El dominio de la funci´on dual se define a partir del dominio de la conjugada del objetivo:

*dom*(*g*) = *{*(*λ, ν*¯) */ − AT λ − CT ν*¯ *∈ dom*(*f*0*∗*)*}*

Una vez calculada *g*(*λ, ν*¯) definimos el **Problema Dual de Lagrange**.

**Definici´on 15.11.** *Sea g*(*λ, ν*¯) *la funci´on dual de Lagrange. Se define enton- ces el* **Problema Dual** *como el problema de optimizaci´on (maximizaci´on)*

m´ax *g*(*λ, ν*¯) *s.a. λ* **0** (68)

*≥*

(*λ,ν*¯)

Las soluciones del problema dual se definen de ´optimos duales

**Definici´on 15.12.** *Diremos que* (*λ∗, ν*¯*∗*) *son* **´optimos duales** *o multiplica- dores de Lagrange O´ptimos si resuelven el problema* ([68](#_bookmark218))*.*

El problema ([68](#_bookmark218)) es convexo porqu´e maximizamos una funci´on concava con restricciones convexas. Esto es cierto incluso si el problema primal no es convexo. Podemos pasar entonces a las condiciones cl´asicas de optimalidad de Karush-Kuhn-Tucker.

##### Las Condiciones de Optimalidad KKT

Supondremos la funci´on *f*0 objetivo y las restricciones de desigualdad *fi* e igualdad *hi* diferenciables. No es necesario suponer convexidad del proble- ma primal pero nos pondremos en este caso por ser el m´as cl´asico, comu´n y simple. Empezamos definiendo unas condiciones que permiten resolver pro- blemas de minimizaci´on con restricciones.

**Definici´on 15.13.** *Las condiciones de Karush, Kuhn, Tucker (KKT) se de- finen por el sistema de ecuaciones e inecuaciones*

*fi*(**x**) *≤* 0*, i* = 1*..m*

*hi*(**x**) = 0*, i* = 1*..p*

*λi ≥* 0*, i* = 1*..m λifi*(**x**) = 0*, i* = 1*..m*

*m p*

*∇f*0(**x**) + *λi∇fi*(**x**) + *νi∇hi*(**x**) = 0

*i*=1

*i*=1

(69)

Vamos a definir el concepto de dualidad fuerte lo que garantizar´a que en un punto de ´optimo se verifican las condiciones KKT.

##### Dualidad fuerte

**Teorema 15.3.** *Sea p∗ el valor ´optimo de un problema y denotamos por d∗*

*el valor ´optimo del problema dual de Lagrange. Entonces siempre se tiene:*

*d∗ ≤ p∗*

*propiedad la que se conoce con el nombre de dualidad d´ebil (***weak duality***). A la cantidad no negativa*

*p∗ − d∗*

*se le conoce con el nombre de* **duality gap***. Si*

*d∗* = *p∗*

*hablaremos de dualidad fuerte (***strong duality***).*

**Teorema 15.4.** *Sean* **x***∗ y* (*λ∗, ν*¯*∗*) *puntos ´optimos (primal y duales) para los cuales el* **gap de dualidad** *(la diferencia entre los valores ´optimos) es cero:*

*d∗* = *p∗*

*Entonces se verifican las Condiciones KKT de* **Karush-Kuhn-Tucker***.*

**Teorema 15.5.** *Si el problema primal es convexo las condiciones (KKT) son necesarias y suficientes para los ´optimos del problema. Si el problema primal no es convexo las KKT son s´olo necesarias y no suficientes.*

Las condiciones KKT juegan un papel fundamental en optimizaci´on. En al- gunos casos es posible resolverlas an´aliticamente y resolver por tanto el pro- blema de optimizaci´on. Por ejemplo en el caso de minimizaci´on cuadr´atica convexa con restricciones de igualdad que desarrollaremos m´as adelante. En general los algoritmos de resoluci´on est´an formulados para resolver las ecua- ciones del sistema KKT.

Empezaremos el estudio de la Optimizaci´on Convexa a trav´es del estudio de unos casos fundamentales conocidos como Programaci´on Lineal y Programa- ci´on Cuadr´atica. Nos basaremos en la teor´ıa y ejemplos del libro de Boyd, Optimizaci´on Convexa, secciones 4.3 para el caso lineal y 4.4 para el caso cuadr´atico. Finalizamos la clase con un breve resumen de optimizaci´on con restricciones.

##### Resumen Optimizaci´on con Restricciones

A veces no queremos optimizar una funci´on *f* en todo el dominio *Df* R*n* sino en un conjunto R*n*. Hablaremos de un problema de **optimizaci´on con restricciones**. Los puntos **x** se dir´an **admisibles** ya que cumplen con las restricciones. Por ejemplo queremos soluciones que est´en en la bo- la unidad del espacio y pediremos una restricci´on sobre la norma del tipo **x** 2 1. Para resolver este problema vamos a definir otro problema, sin restricciones, cuya soluci´on es tambi´en soluci´on del problema de optimiza- ci´on con restricciones. Unas condiciones que permiten resolver un problema de optimizaci´on con restricciones se deben a **Karush, Kuhn y Tucker**, m´as brevemente **condiciones KKT** que permiten manejar restricciones de igualdad y desigualdad generalizando as´ı la teor´ıa de los **multiplicadores de Lagrange** que s´olo pueden manejar restricciones de igualdad. Supon- gamos entonces que se puede describir mediante *m* funciones *g*(*i*)(**x**) y *n* funciones *h*(*j*)(**x**) tales que

*S*

*∈ S*

*S ⊂*

*⊂*

*|| || ≤*

*S* = *{***x** *∈* R*n | g*(*i*)(**x**) = 0*, ∀ i, h*(*j*)(**x**) *≤* 0*, ∀ j}*

Introducimos constantes *λi* y *αj* para cada restricci´on y las almacenamos en los vectores *λ*¯ = (*λ*1*, .., λm*), *α*¯ = (*α*1*, .., αn*) . Son los multiplicadores de KKT. Construimos la funci´on **Lagrangiana generalizada**:

*L*(**x***, λ*¯*, α*¯) = *f* (**x**) + *λig*(*i*)(**x**) + *αjh*(*j*)(**x**)

*i j*

El problema con restricciones

m´ın *f* (**x**)

**x***∈S*

Es equivalente al problema sin restricciones

m´ın (m´ax m´ax *L*(**x***, λ*¯*, α*¯)

**x***∈S*

*λ*¯

*α*¯*≥***0**

porqu´e para los puntos admisibles **x** *∈ S* se tiene

m´ax m´ax *L*(**x***, λ*¯*, α*¯) = *f* (**x**)

*λ*¯ *α*¯*≥***0**

mientras que para los puntos **x** *∈/ S* se tiene

m´ax m´ax *L*(**x***, λ*¯*, α*¯) = *∞*

*λ*¯ *α*¯*≥***0**

Diremos que una restricci´on de desigualdad es **activa** si *h*(*j*)(**x**) = 0. Se tiene:

*α*¯ *⊙* **h**(**x**) = **0**

es decir al menos una de las restricciones *αj* 0 o *h*(*j*)(**x**) 0 es activa (se verifica la igualdad). Esta condici´on se llama **complementary slackness**.

*≥ ≤*

Como ejemplo volvemos a la minimizaci´on de la funci´on

*f* (**x**) = (1*/*2)*||A***x** *−* **b***||*2

2

complementada con la restricci´on **x***T* **x** 1. Sin restricci´on la soluci´on de norma m´ınima viene dada por la pseudo-inversa de Moore-Penrose: **x** = *A†***b**. Si **x** verifica la restricci´on tenemos un o´ptimo del problema con restricci´on y *α* = 0. En caso contrario, *α >* 0 tenemos que encontrar una soluci´on en donde la restricci´on sea activa (por la complementary slackness): *h*(**x**) = 0. Construimos la lagrangiana generalizada con multiplicador *α ≥* 0.

*≤*

*L*(**x***, α*) = *f* (**x**) + *αh* (**x**) = 1 *||A***x** *−* **b***||*2 + *α* **x***T* **x** *−* 1

2

2

Consideramos entonces el problema

m´ın m´ax *L*(**x***, α*)

**x***∈S α≥***0**

Imponiendo las condiciones de primer orden

*AT A***x** *AT* **b** + 2*α***x** = **0***, ∂L*(**x***, α*¯) = *h* (**x**) = **x***T* **x** 1

*— −*

*∂α*

es decir

cuya soluci´on es

(*AT A* + 2*αIn*)**x** = *AT* **b**

**x** = (*AT A* + 2*αIn*)*−*1*AT* **b** (70)

y tiene que verificar la restricci´on. La derivada

*∂L*(**x***, α*) = **x***T* **x** 1

*−*

*∂α*

nos dice que si no se verifica la restricci´on **x***T* **x** *>* 1 entonces *∂L*(**x***, α*)*/∂α >* 0. Luego para maximizar *L*(**x***, α*) hay que subir *α*. La soluci´on definida en ([70](#_bookmark221)) tendr´a norma menor (porqu´e la penalizaci´on es mayor). Este proceso se aplica iterativamente hasta cuando **x** tiene norma correcta y

*∂L*(**x***, α*)*/∂α* = *h*(**x**) = 0

**Ejercicio 15.3.** *Escribir un algoritmo que resuelva el problema de optimi- zaci´on con restricci´on*

*x∗* = arg m´ın *f* (**x**) = arg m´ın 1 *||A***x** *−* **b***||*2

*siendo*

**x***∈S*

**x***∈S* 2 2

*S* = *{***x** *∈* R*n |* **x***T* **x** = *||***x***||*2 *≤* 1*}*

#### Programaci´on Convexa

En el marco de la Programaci´on Convexa se incluyen dos casos particulares fundamentales: la Programaci´on Lineal y la Programaci´on Cuadr´atica.

##### Programaci´on Lineal

Vamos a resolver algunos **problemas de Programaci´on Lineal**. Se trata de encontrar un vector **x** que minimiza una forma lineal *cT* **x** y que cumpla con restricciones lineales del tipo desigualdades e igualdades en la forma

m´ın **c***T* **x***,* s.a *G***x** *≤* **h***, A***x** = **b***, lb ≤* **x** *≤ ub*

**x**

Esta formulaci´on es la que se considera en matlab y que se puede resolver mediante el comando

x = linprog(f,A,b,Aeq,beq,lb,ub,options)

Detalles y ejemplos sobre la resoluci´on pr´actica de este tipo de problemas se encuentran en la direcci´on

https://es.mathworks.com/help/optim/ug/linprog.html

Profundizamos en la teor´ıa tal y como se encuentra en el libro de Optimiza- ci´on Convexa de S. Boyd. Dado un problema de Programaci´on Lineal (LP) en la **forma General**

m´ın **c***T* **x***,* s.a *G***x h***, A***x** = **b** (71)

*≤*

**x**

donde *G* R*m,n* y *A* R*p,n* la interpretaci´on geom´etrica del Conjunto Ad- misible *K* del LP es la de un poliedro (cerrado y acotado)[20](#_bookmark225) ya que viene definido por las soluciones de un conjunto finito de desigualdades (d´ebiles, no estrictas)

*P*

*∈ ∈*

e igualdades lineales

*G***x** *≤* **h**

*A***x** = **b***.*

20Se define poliedro a un cuerpo geom´etrico cuyas caras son planas y encierran un volumen finito.

Siempre podemos pasar un LP en la Forma General ([71](#_bookmark224)) a la **Forma Estandar**

m´ın **c***T* **x***,* s.a *A***x** = **b***,* **x 0** (72)

*≥*

**x**

en donde las u´nicas desigualdades son las de no negatividad componente a componente de las variables. El primer paso consiste en introducir variables de holgura no negativas (slacks) *si* para las desigualdades en ([71](#_bookmark224)) obteniendo

m´ın **c***T* **x***,* s.a *G***x** + **s** = **h***, A***x** = **b***,* **s 0**

*≥*

**x**

El segundo paso consiste en expresar la variables **x** como la diferencia de dos variables no negativas llamadas parte positiva **x**+ y parte negativa **x***−* en la forma **x** = **x**+ **x***−*, **x**+, **x***−* **0**. Observa que la parte negativa **x***−* es positiva. Se tiene

*— ≥*

m´ın **c***T* **x**+ **c***T* **x***−,*

*−*

**x**

s.a *G***x**+ *− G***x***−* + **s** = **h***, A***x**+ *− A***x***−* = **b***,* **x**+ *≥* **0***,* **x**+ *≥* **0***,* **x** *≥* **0**

que es de la forma ([72](#_bookmark226)).

Si el LP no tiene restricciones de igualdad entonces se define un LP en

**Forma de Desigualdad** y se escribe

m´ın **c***T* **x***,* s.a *A***x b***,* (73)

*≤*

**x**

Veremos que las dos formas, estandar ([72](#_bookmark226)) y desigualdad ([73](#_bookmark227)) est´an en dua- lidad. Vamos a calcular la funci´on lagrangiana, la funci´on dual y una cota inferior del ´optimo de un problema en la forma estandar ([72](#_bookmark226)). Las restriccio- nes de desigualdad vienen dadas por las funciones

*fi*(**x**) = *−***x***i, i* = 1*..n* es decir *n* = *m*. Introducimos multiplicadores

*λi, i* = 1*..m, νi, i* = 1*..p* = *n.*

##### Funci´on Lagrangiana

La funci´on lagrangiana es

*m p*

*L*(**x***, λ, ν*¯) = *f*0(**x**) + *λifi*(**x**) + *νihi*(**x**) =

*i*=1

*i*=1

*m*

= *f*0(**x**) *− λixi* + *ν*¯*T* (*A***x** *−* **b**) = *−***b***T ν*¯ + (**c** + *AT ν*¯ *− λ*)*T* **x**

*i*=1

en donde hemos utilizado las definiciones previas y las propiedades del ope- rador de trasposici´on:

*m*

*f*0(**x**) = **c***T* **x***, − λixi* = *−λT* **x***,*

*i*=1

*ν*¯*T A***x** = (*AT ν*¯)*T* **x***, −ν*¯*T* **b** = *−***b***T ν*¯

##### Funci´on Dual

La funci´on dual es

*g*(*λ, ν*¯) = ´ınf *L*(**x***, λ, ν*¯) = *−***b***T ν*¯ + ´ınf(**c** + *AT ν*¯ *− λ*)*T* **x**

**x**

**x**

Se trata de una funci´on lineal en **x** luego es acotada s´olo si es identicamente nula:

es decir

en cuyo caso

(**c** + *AT ν*¯ *− λ*)*T* **x** = **0***, ∀* **x**

**c** + *AT ν*¯ *− λ* = **0**

´ınf *L*(**x***, λ, ν*¯) = m´ın *L*(**x***, λ, ν*¯) = *−***b***T ν*¯

**x**

**x**

es decir el ´ınfimo de la Lagrangiana es un m´ıniimo ya que se alcanza. Se tiene entonces la funci´on dual

*g*(*λ, ν*¯) =

 *−bT ν*¯ si **c** + *AT ν*¯ *− λ* = **0**

 *−∞* otherwise

Cuando (*λ, ν*¯) satisface las condiciones

*λ ≥* **0***,* **c** + *AT ν*¯ *− λ* = **0**

tenemos una **cota inferior no trivial del o´ptimo** del LP es forma estandar ([72](#_bookmark226)):

*g*(*λ, ν*¯) = *−***b***T ν*¯ *≤ p∗*

Para encontrar la mejor cota inferior pasamos a definir el problema Dual de Lagrange.

##### El problema Dual

Usando ([68](#_bookmark218)) se tiene que el problema Dual se define por

m´ax *g*(*λ, ν*¯) s.a. *λ* **0**

*≥*

(*λ,ν*¯)

Puesto que *g* es finita s´olo si

**c** + *AT ν*¯ *− λ* = **0**

definimos un problema equivalente expl´ıcitando esta restricci´on

*− − ≥*

m´ax **b***T ν*¯ *,* s.a. **c** + *AT ν*¯ *λ* = **0***, λ* **0**

(*λ,ν*¯)

lo que se puede expresar en la forma

*− ≥*

m´ax **b***T ν*¯ *,* s.a. **c** + *AT ν*¯ **0**

(*λ,ν*¯)

que es un LP en la forma de desigualdad ([73](#_bookmark227)). Resumiendo, el dual de un LP estandar es un LP en forma de desigualdad. Veamos que el dual de un LP en forma de desigualdad ([73](#_bookmark227)) es un LP en forma estandar ([72](#_bookmark226)).

##### Funci´on Lagrangiana

La lagrangiana de un LP en la forma de desigualdad ([73](#_bookmark227)) es:

*L*(**x***, λ*) = **c***T* **x** + *λT* (*A***x** *−* **b**) = *−***b***T λ* + (*AT λ* + **c**)*T* **x Funci´on Dual**

La funci´on dual es

*g*(*λ*) = ´ınf *L*(**x***, λ*) = *−***b***T λ* + ´ınf(*AT λ* + **c**)*T* **x**

**x**

**x**

de donde (recordar que una funcion linel es acotada s´olo si es identicamente nula)

*g*(*λ*) =

 *−***b***T λ* si *AT λ* + **c** = **0**

 *−∞* otherwise

la variable dual es dual admisible si *λ* **0** y *AT λ* + **c** = **0**. Incluyendo las condiciones de admisibilidad como restricciones se tiene

*≥*

##### El problema Dual

m´ax( **b***T λ*)*,* s.a. *AT λ* + **c** = **0***, λ* **0**

*— ≥*

(*λ,ν*¯)

que es un LP en la forma estandar ([72](#_bookmark226)).

##### Dualidad fuerte y d´ebil

En general no hay dualidad fuerte pero es posible encontrar condiciones su- ficientes. Por ejemplo, si el problema primal es convexo (de la forma ([59](#_bookmark205))) una condici´on suficiente se conoce como **Slater’s condition**: existe **x** en el interior del *dom*(*fi*) tal que *fi*(**x**) *<* 0 y *A***x** = **b**. Un punto que satisface la condici´on de Slater se llama **estrictamente admisible** ya que satisface las restricciones con desigualdad estricta. Si algunas de las restricciones de desiguladad son afines la condici´on de Slater s´olo se tiene que verificar para las restricciones no afines (condici´on de Slater d´ebil). La **condicio´n de Sla- ter** implica por tanto la dualidad fuerte para problemas convexos. Adema´s implica que *d∗ > −∞* y que existen (*λ∗, ν*¯*∗*) duales admisibles tales que

*g*(*λ∗, ν*¯*∗*) = *d∗* = *p∗*

Utilizando la condici´on de Slater d´ebil se tiene que la dualidad fuerte se tiene para todo problema de programaci´on Lineal (en la forma estandar ([72](#_bookmark226)) o de desigualdad ([73](#_bookmark227))) para el cual el problema primal es admisible. Aplicando este resultados a los problemas duales asociados se tiene que la dualidad fuerte se tiene para todo problema de programaci´on Lineal que tenga dual admisible. El u´nico caso en Programaci´on Lineal en donde no se tiene dualidad fuerte es cuando ambos problemas (primal y dual) no son admisibles.

##### Problemas Lineales

**Ejercicio 15.4.** *Resolver con el algoritmo de programaci´on lineal* **linprog.m**

*el problema de minimizaci´on con restricciones*

m´ın **c***T* **x** = m´ın (*−*150*x*1 *−* 175*x*2)

*K*

*K*

*siendo K el conjunto convexo definido por las desigualdades*

 7*x*1 + 11*x*2 *≤* 77

****

**** 10*x*1 + 8*x*2 *≤* 80

*K* = *x*1 *≤* 9

*x*2 *≤* 6

**** *x*1 *≥* 0*, x*2 *≥* 0

Hay *n* = 2 variables inc´ognitas y *m* = 2 restricciones de desigualdad junto a acotaciones inferiores y superiores (box contraints). El vector de coeficientes de la funci´on objetivo es **c***T* = (*−*150*, −*175). La matriz de coeficientes para las restricciones de desigualdad *A***x** *≤* **b** es

 7 11 

*A* =  10 8 

El vector t´ermino derecho de las desigualdades es **b***T* = (77*,* 80). No hay restricciones de igualdad por lo cual

*Aeq*

0 0 = 0

0 0

= *,* **b**

0

*eq*

Los vectores que contienen las acotaciones inferiores y superiores para las variables, box constrained, son

La resoluci´on es

6

c=[-150;-175]; A=[7 11;10 8];

0

**l***b b*

= *,* **u**

0

= 9

b=[77;80];

Aeq=[0 0;0 0];

beq=[0;0];

LB=[0;0];

UB=[9;6];

x=linprog(c,A,b,Aeq,beq,LB,UB)

Observa que si quieres tener m´as informaci´on sobre la calidad de la soluci´on primal y tambi´en los valores de las variables duales puedes usar en output el comando

[x,fval,exitflag,output,lambda] =linprog(f,A,b,Aeq,beq,lb); x,lambda.ineqlin,lambda.lower

Los *λi* = 0, es decir no nulos estar´an asociados a restricciones de desigualdad o acotaciones que se verifican en la frontera, es decir son igualdades.

*̸*

**Ejercicio 15.5.** *Resolver con el algoritmo de programaci´on lineal* **linprog.m**

*el problema de maximizaci´on con restricciones*

m´ax **c***T* **x** = m´ax (*x*1 + 2*x*2)

*K K*

*siendo K el conjunto convexo definido por las desigualdades*

 2*x*1 + *x*2 *≤* 10

****

**** *x*1 + *x*2 *≤* 6

*K* = *−x*1 + *x*2 *≤* 2

*−*2*x*1 + *x*2 *≤* 1

**** *x*1 *≥* 0*, x*2 *≥* 0

**Ejercicio 15.6.** *Escribir en forma Estandar el problema de Maximizaci´on Lineal con Restricciones dadas por Desigualdades Lineales y desigualdades de no negatividad.*

m´ax *f* (*x*1*, x*2) = m´ax (*x*1 + *x*2)

*K K*

*siendo K el conjunto convexo definido por las desigualdades*

*K* = *x*1 + 2*x*2 *≤* 3

**** 2*x*1 + *x*2 *≤* 4

**** *x*1 *≥* 0*, x*2 *≥* 0

*Respuesta* Para ello introducimos *m* = 2 variables de holgura (slack variables)

**** 2*x*1 + *x*2 + *s*1 = 4

*s*1 *≥* 0, *s*2 *≥* 0 en la forma

*K* = *x*1 + 2*x*2 + *s*2 = 3

**** *x*1 *≥* 0*, x*2 *≥* 0*, s*1 *≥* 0*, s*2 *≥* 0

Para su resoluci´on, mediante el m´etodo del simplex, se considera la funci´on objetivo

a trav´es de la ecuaci´on

*z* = *f* (*x*1*, x*2) = *x*1 + *x*2

*z − x*1 *− x*2 = 0

y se considera la matriz de coeficientes de las variables *z*, *x*1, *x*2, *s*1, *s*2

ampliada con los respectivos t´erminos derechos

 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 *−*1 *−*1 | | | 0 | 0 | 0 |
| 0 2 1 | | | 1 | 0 | 4 |
| 0 | 1 | 2 | 0 | 1 | 3 |

 





Realizando un proceso de eliminaci´on de tipo Gauss-Jordan obtenemos

 1 0 0 1*/*3 1*/*3 7*/*3 

 0 1 0 2*/*3 *−*1*/*3 5*/*3 

0 0 1 *−*1*/*3 2*/*3 2*/*3

de donde deducimos que *x*1 = 5*/*3, *x*2 = 2*/*3 son las variables b´asicas, *s*1 = 0, *s*2 = 0 son las variables no b´asicas, la soluci´on es admisible, o´ptima y con valor m´aximo del objetivo *z* = 7*/*3.

**Ejercicio 15.7.** *Sea dado el problema de Minimizaci´on Lineal con Restric- ciones dadas por Desigualdades Lineales*

m´ın *f* (*x, y*) = m´ın *−x − y*

*K*

*K*

3

*siendo K el conjunto convexo definido por las desigualdades*

*x* + *y ≤* 2

****

*y*

*x* + 4 *≤* 1

****

**** *x − y ≤* 2

*K* = *x*

*—* 4 *− y ≤* 1

**** *−x − y ≤ −*1

****

**** *−x* + *y ≤* 2

*Clasificar el LP en una de las tres formas anteriores y escibirlo en la Forma Estandard* ([72](#_bookmark226))*. Resolver el problema utilizando matlab.*

*Respuesta* Se trata de un LP en forma de desigualdad

m´ın *cT* **x***, A***x b**

*≤*

*K*

A partir de los datos del problema definimos, para *n* = 2, *m* = 6, *p* = 0

 





|  |  |
| --- | --- |
| 1 | 1 |
| 1 | 1*/*4 |
| 1 | *−*1 |
| *−*1*/*4 | *−*1 |
| *−*1 | *−*1 |
| *−*1 | 1 |

 

2 

1

 2 

 *−*1 

*A* = *, b* =



1

   

*, c* =  *−*1*/*3 

   *−*1 







6*,*1

2

6*,*2

2*,*1

Para obtener la formulaci´on Estandar necesitamos introducir variables de holgura, slack variables. Sin embargo hay un coeficiente del vector **b** que es negativo. En este caso se opera de la siguiente forma. La ecuaci´on asociada al coeficiente negativo es

de donde

*−x − y ≤ −*1

*x* + *y ≥* 1

y se introduce una slack con signo negativo para obtener una igualdad,

*x* + *y − s*5 = 1

Las ecuaciones del sistema son

*y*

**** *x* + *y* + *s*1 = 2

*x* + + *s* = 1

****

2

4

****



*x − y* + *s*3 = 2

*x*

*—* 4 *− y* + *s*4 = 1

**** *x* + *y − s*5 = 1

**** *−x* + *y* + *s* = 26

****

para *si* 0, *i* = 1*..m* = 6. Las variables *x, y* son **libres**, no tienen res- tricciones de igualdad luego se introducen sus respectivas partes positivas y negativas en la forma **x** = **x**+ **x***−*, siendo **x**+*,* **x***−* **0**. Por componentes se tiene **x** = (*x, y*)*T* , **x**+ = (*x∗, y∗*), **x***−* = (*x−, y−*). La funci´on objetivo a minimizar es

*≥*

*— ≥*

*z* = **c***T*

*y* +

**x** = *−x −* 3 = *−x −*

*y*+ *y−*

+ *x* +

*−*

3 3

Las ecuaciones del sistema son ahora

*y*+

**** *x*+ *− x−* + *y*+ *− y−* + *s*1 = 2

+

*x − x−* + 4 *−*

****

 *x*+ *− x− − y*+ + *y−* + *s*3 = 2

*y−*

4 + *s*2 = 1

*x*+ *x−*

*−* +

****

*— y*+ + *y−* + *s*4 = 1

4

**** *x*+

*−*

4

*x− y*+ *− y− − s* = 1

+ 5

****

****

*−x*+ + *x−* + *y*+ *− y−* + *s*6 = 2

siendo la variable a optimizar

**x**˜ = (*x*+*, x−, y*+*, y−, s*1*, s*2*, s*3*, s*4*, s*5*, s*6) *∈* R10*,*

La matriz del sistema para la forma estandar es

*x*˜ *≥* 0

siendo



*A*˜ = 





*−*1 



 1 



 *,*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | *−*1 | 1 | *−*1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | *−*1 | 1*/*4 | *−*1*/*4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | *−*1 | *−*1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| *−*1*/*4 | 1*/*4 | *−*1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | *−*1 | 1 | *−*1 | 0 | 0 | 0 | 0 | *−*1 | 0 |
| *−*1 | 1 | 1 | *−*1 | 0 | 0 | 0 | 0 | 0 | 1 |

6*,*10



 2 

 *−*1*/*3 

1

˜  2 

1*/*3

 0 

0

**b** = 1

 

1

 2 

*,*

6*,*1

**c**˜ =

 0



0

 0

0







10*,*1

luego el problema en forma estandar (compara con pg 219 del libro de Boyd) es

*T* ( + *y*+

m´ın **c**˜

**x**

**x**˜ = m´ın

**x**

*−x*

*−*

+ *x*

3

*— y−* ˜ ˜

3

Calculamos ahora el problema dual. La lagrangiana de un LP en la forma de desigualdad ([73](#_bookmark227)) es:

+

*,* s.a.

*A***x**˜ = **b***,*

**x**˜ *≥* **0***.*

*L*(**x***, λ*) = **c***T* **x** + *λT* (*A***x** *−* **b**) = *−***b***T λ* + (*AT λ* + **c**)*T* **x** =

 2 *T*  *λ* 

1

  



1

*λ*2

1 1

1 1*/*4





*T*  *λ*  *T*

   







1







*λ*2

 2 

 *λ*3 

 1 *−*1 

 *λ*3  *−*1  *x*

= *−*  1 

*λ*

 *λ*4 

 *−*1*/*4 *−*1 

*λ*4 +  *−*1*/*3  

 *y* 

 *−*1 

+



1*,*6

 *λ*6



2

5  

6*,*1

*−*1 *−*1  5 

*−*1 1

6*,*1

2*,*1







2*,*6

 *λ*6





*λ*

2*,*1

La funci´on dual es



1*,*2

*g*(*λ*) = ´ınf *L*(**x***, λ*) = *−***b***T λ* + ´ınf(*AT λ* + **c**)*T* **x**

**x**

**x**

de donde (recordar que una funci´on linel es acotada s´olo si es identicamente nula)

*g*(*λ*) =

 *−***b***T λ AT λ* + **c** = 0

 *−∞* otherwise

la variable dual es dual admisible si *λ* **0** y *AT λ* + **c** = 0. Incluyendo las condiciones de admisibilidad como restricciones se tiene

*≥*

m´ax(*−bT λ*)*,* s.a. *AT λ* + *c* = 0*, λ ≥* 0

que es un LP en la forma estandar ([72](#_bookmark226)). La funci´on objetivo a maximizar (siendo un problema dual) es

 2 *T*  *λ* 

1







*z* = *f*0(*λ*) = *−***b***T*

1

2





*λ* = *−* 1





*λ*2 *λ*3

*λ*4 = *−*2*λ*1*−λ*2*−*2*λ*3*−λ*4+*λ*5*−*2*λ*6







 *−*1  5 







 *λ*6



*λ*

2

1*,*6

6*,*1

Vamos a estudiar el sistema con restricciones

*AT λ* + **c** = **0***, λ ≥* **0**

escrito en la forma



1 1 *T*

 *λ*1 

1 1*/*4 *λ*2







  

*−*

 1 1   *λ*3  1

=  





 *−*1*/*4 *−*1  *λ*4 1*/*3

es decir

*−*1 *−*1

*−*1 1

2*,*6

*λ*5

 *λ*6





6*,*1

2*,*1

 1 1 1 *−*1 *−*1 *−*1 



 *λ*1 

*λ*2

 *λ*3 

 1 

 1 1*/*4 *−*1 *−*1 *−*1 1

*λ*4

2*,*6 *λ*5





 *λ*6



6*,*1

= 1*/*3

2*,*1

Se tiene *Rg*(*AT* ) = 2 luego por la f´ormula de la dimensi´on

 

*dim N* (*A*) = 6 *−* 2 = 4

y existen infinitas soluciones *λ* R6 dependientes de 4 par´ametros. Escri- biendo la ecuaci´on del objetivo en la forma

*∈*

*z* + 2*λ*1 + *λ*2 + 2*λ*3 + *λ*4 *− λ*5 + 2*λ*6 = 0

podemos aplicar el proceso de eliminaci´on de Gauss-Jordan a la matriz

 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 1 | 2 | 1 *−*1 2 | 0 |
| 0 | 1 | 1 | 1 | *−*1*/*4 *−*1 *−*1 | 1 |
| 0 | 1 | 1*/*4 | *−*1 | *−*1 *−*1 1 | 1*/*3 |

 





obtenemos

 1 0 0 8*/*3 5*/*2 1 4*/*3 *−*10*/*9 





0 1 0 *−*5*/*3 *−*5*/*4 *−*1 5*/*3 1*/*9

0 0 1 8*/*3 1 0 *−*5*/*3 8*/*9

de donde deducimos la soluci´on o´ptima dual *λ∗* = (1*/*9*,* 8*/*9*,* 0*,* 0*,* 0*,* 0) con

*d∗* = *p∗* = *−*10*/*9.

**Ejercicio 15.8.** *Resolver el problema de Programaci´on Lineal anterior su- poniendo adem´as la restricci´on de igualdad*

*y* 1

*x* + =

4 2

*Clasificar el LP en una de las tres formas anteriores. Resolver el problema utilizando matlab.*

**Ejercicio 15.9.** *Resolver el problema de Programaci´on Lineal anterior su- poniendo adem´as la restricci´on de tipo acotaci´on*

1 3 5

*−*1 *≤ x ≤ −* 2 *, −* 2 *≤ y ≤* 4

*Clasificar el LP en una de las tres formas anteriores. Resolver el problema utilizando matlab.*

**Ejercicio 15.10.** *Sea dado el problema de Optimizaci´on Lineal con Restric- ciones dadas por Desigualdades Lineales y condiciones de no negatividad*

m´ın *f* (*x, y*) = m´ın (*−*4*x −* 6*y*)

*K*

*K*

*siendo K el conjunto convexo definido por las desigualdades*

**** *−x* + *y ≤* 11

 *x* + *y ≤* 27

*K* = **** 2*x* + 5*y ≤* 90

**** *x, y ≥* 0

*Clasificar el LP en una de las tres formas anteriores. Resolver el problema utilizando matlab. Verificar la soluci´on* (*x∗, y∗*) = (15*,* 12) *con valor ´optimo p∗* = *−*132*.*

**Ejercicio 15.11.** *Sea dado el problema de Optimizaci´on Lineal con Restric- ciones dadas por Desigualdades Lineales y condiciones de no negatividad*

m´ın *f* (*x, y, z*) = m´ın (*−*2*x* + *y −* 2*z*)

*K*

*K*

*siendo K el conjunto convexo definido por las desigualdades*

 2*x* + *y ≤* 10

****

 *x* + 2*y −* 2*z ≤* 20

*K* = **** *y* + 2*z ≤*5

*x, y, z ≥* 0

****

*Clasificar el LP en una de las tres formas anteriores . Resolver el problema utilizando matlab. Verificar la soluci´on* (*x∗, y∗, z∗*) = (5*,* 0*,* 5*/*2) *con valor*

*´optimo p∗* = *−*15*.*

**Ejercicio 15.12.** *Sea dado el problema de Optimizaci´on Lineal con Restric- ciones dadas por Desigualdades Lineales, igualdades lineales y condiciones de no negatividad*

m´ın *f* (*x, y, z*) = m´ın (*−*3*x −* 2*y − z*)

*K*

*K*

*siendo K el conjunto convexo definido por*

**** 4*x* + *y* + *z* = 30

 2*x* + 3*y* + *z ≤* 60

*K* = **** *x* + 2*y* + 3*z ≤* 40

**** *x, y, z ≥* 0

*Clasificar el LP en una de las tres formas anteriores. Resolver el proble- ma utilizando matlab. Verificar la soluci´on* (*x∗, y∗, z∗*) = (3*,* 18*,* 0) *con valor*

*´optimo p∗* = *−*45*.*

##### Programaci´on Cuadr´atica

Pasamos ahora al caso fundamental de la programaci´on cuadr´atica. Detalles y ejemplos sobre la resoluci´on pr´actica de este tipo de problemas se encuentran en la direcci´on

https://es.mathworks.com/help/optim/ug/quadprog.html

El problema de optimizacion convexa ([59](#_bookmark205)) tiene el nombre de Programa- ci´on Cuadr´atica si la funci´on objetivo es convexa y cuadr´atica y todas las restricciones son afines. Se puede expresar en la forma

m´ın (1 **x***T P* **x** + **q***T* **x** + *r* (74)

**x**

2

s.a *G***x** *≤* **h***, A***x** = **b***, lb ≤* **x** *≤ ub*

siendo *P Sn* (matriz sim´etrica semi-definida positiva), *G* R*m,n*, *A* R*p,n*

+

*∈ ∈ ∈*

y *r* una constante.

Esta formulaci´on es la que se considera en matlab (pero con *r* = 0) y que se puede resolver mediante el comando

x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)

en donde utilizamos las variables de matlab siendo clara la identificaci´on:

*H* = *P, f* = **q***, A* = *G, b* = **h***, Aeq* = *A, beq* = **b**

La interpretaci´on geom´etrica del Conjunto Admisible *K* del QP es la de un polieedro . Cuando *P* = 0 el QP es un LP. Uno de los ejemplos m´as im- portantes de Programaci´on Cuadr´atica es el **Problema de M´ınimos Cua- drados**: Consiste en la minimizaci´on de la funci´on cuadr´atica Convexa

*P*

m´ın *||A***x** *−* **b***||*2 = m´ın **x***T AT A***x** *−* 2**b***T A***x** + **b***T* **b** (75)

**x***∈*R*n*

2

**x**

en todo el espacio de soluciones **x** *∈* R*n*. Observa que

arg m´ın *||A***x** *−* **b***||*2 = arg m´ın **x***T AT A***x** *−* 2**b***T A***x**

**x***∈*R*n*

2

**x***∈*R*n*

ya que el t´ermino **b***T* **b** R es una constante que modifica el valor m´ınimo de la energ´ıa pero no el punto en donde se alcanza. Esto es lo que hace matlab: elimina este t´ermino al calcular el valor de la energ´ıa. El problema ([75](#_bookmark231)) es

*∈*

un QP sin restricciones y existe una soluci´on anal´ıtica en t´erminos de la pseudo-inversa de Moore-Penrose

**x***∗* = *A†***b**

Esta formulaci´on es adecuada cuando el sistema *A***x** = **b** es incompatible. Todos los residuos no nulos y se busca el de m´ınima energ´ıa. Si adem´as hay unas restricciones de desigualdad lineales diremos que tenemos un **problema de M´ınimos cuadrados con restricciones**. Por ejemplo podemos pedir una acotaci´on de las variables en forma

m´ın *||A***x** *−* **b***||*2*,* s.a *li ≤ xi ≤ ui, i* = 1*..., n* (76)

**x**

2

Otra formulaci´on de M´ınimos cuadrados con restricciones de igualdad es

m´ın **x***T* **x***,* s.a *A***x** = **b** (77)

**x**

donde *A ∈* R*p×n* y la forma cuadr´atica est´a asociada a la matriz

*P* = 2*Id*

si usamos la formulaci´on general ([74](#_bookmark230)). No hay restricciones de desigualdad y hay *p* restricciones de igualdad lineal. Esta formulaci´on es adecuada cuando el sistema *A***x** = **b** es compatible indeterminado ya que en este caso el conjunto Admisible es no vac´ıo y es un sub-espacio vectorial de dimensi´on 1. El caso compatible determinado es trivial.

*≥*

Introduciendo *p* multiplicadores de Lagrange la funci´on Lagrangiana

*L* : R*n ×* R*p →* R

es

con funci´on dual

*L*(**x***, ν*¯) = **x***T* **x** + *ν*¯*T* (*A***x** *−* **b**)

*g*(*ν*¯) = ´ınf *L*(**x***, ν*¯)*.*

**x**

La lagrangiana es cuadr´atica y convexa luego podemos minimizarla usando las condiciones de optimalidad de tipo KKT

*∇***x***L*(**x***, ν*¯) = 2**x** + *AT ν*¯ = **0***, ∇ν*¯*L*(**x***, ν*¯) = *A***x** *−* **b** = **0**

de donde el **´optimo primal**

**x***∗* = *−*(1*/*2)*AT ν*¯*∗*

Puesto que **x***∗* verifica la restricci´on, *A***x***∗* = **b**, se tiene

*−*(1*/*2)*AAT ν*¯*∗* = **b**

y el **o´ptimo dual** es

*ν*¯*∗* = *−*2(*AAT* )*−*1**b**

Sustituyendo obtenemos la funci´on dual

*g*(*ν*¯) = ´ınf *L*(*−*(1*/*2)*AT ν*¯*, ν*¯) = *−* 1 *ν*¯*T AAT ν*¯ *−* **b***T ν*¯

**x**

4

que es una funci´on cuadr´atica concava definida en R*p*. Si definimos

*p∗* = ´ınf*{***x***T* **x** */ A***x** = **b***}*

se tiene la acotaci´on inferior

*g*(*ν*¯) = *−* 1 *ν*¯*T AAT ν*¯ *−* **b***T ν*¯ *≤ p∗*

4

El problema dual asociado es

m´ax *g*(*ν*¯) = m´ax (*−* 1 *ν*¯*T AAT ν*¯ *−* **b***T ν*¯

*ν*¯

*ν*¯

4

que es un problema de optimizaci´on (maximizaci´on) sin restricciones de una forma cuadr´atica concava. La condici´on de Slater es simplemente que el pro- blema primal sea admisible: Si *b ∈ rg*(*A*) (es decir *p∗ < ∞*) entonces hay dualidad fuerte *d∗* = *p∗*. Incluso hay dualidad fuerte si *p∗* = *∞*. En este caso *b ∈/ rg*(*A*) luego existe *z* tal que *AT z* = 0 y *bT z ̸*= 0. Entonces la funci´on dual *g*(*ν*¯) no es acotada a lo largo de la recta *{tz, t ∈* R*}* luego *d∗* = *∞* y se tiene la dualidad fuerte ya que *p∗* = *d∗* = *∞*.

Consideremos ahora el problema de Programaci´on Cuadr´atica sin restriccio- nes de Desigualdad m´as general, es decir *G* = 0, *P* = *Id*, **q** = 0, *r* = 0 y s´olo restricciones de igualdad

*̸ ̸ ̸*

m´ın (1 **x***T P* **x** + **q***T* **x** + *r ,* s.a *A***x** = **b**

**x**

2

siendo *P ∈ Sn* (cuadrada, sim´etrica, semi-definida positiva) y *A ∈* R*p×n*,

+

**x** R*n*, **b** R*p*, **q** R*n* y *r* R. Las **condiciones KKT** para este problema son

*∈ ∈ ∈ ∈*

*A***x***∗* = *b, P* **x***∗* + **q** + *AT ν*¯*∗* = **0**

que se pueden escribir en la forma

 *P AT*

  **x***∗*   *−***q** 

 *A* **0**

  *ν*¯*∗*  = 

**b**  (78)

Resolver este sistema de *n* + *p* ecuaciones y *n* + *p* inc´ognitas **x***∗* y *ν*¯‘ nos da los o´ptimos, primal y dual, del problema. Sea dada la matriz en bloques anterior, conocida como matriz KKT,

*∗*

 *P AT* 



*MKKT* =

Se pueden dar los siguientes casos

*A* **0** 

*n*+*p,n*+*p*

1. Si la matriz *MKKT* es no singular existe un u´nico par primal-dual o´pti- mo (**x***∗, ν*¯*∗*).
2. Si la matriz *MKKT* es singular pero el sistema KKT admite soluciones entonces cada soluci´on nos da un par primal-dual o´ptimo.
3. Si el sistema KKT no tiene soluci´on entonces el problema de minimiza- ci´on cuadr´atica no tiene soluci´on, la energ´ıa no es acotada y el problema no admisible.

Existen varias condiciones que aseguran que la matriz KKT es no singular. Por ejemplo si (*P* ) (*A*) = 0 que se traduce diciendo que las matrices *P* y *A* no tienen un nu´cleo comu´n no trivial. Otro caso especial es *P >* 0 (definida positiva) que implica que la matriz KKT es no singular.

*N ∩ N { }*

Apliquemos estas ideas a un problema modelo,

**Ejemplo 15.4.** *Sea dado el problema de Minimizaci´on Cuadr´atica con Res- tricciones de igualdad Lineales*

m´ın *f*0(*x*1*, x*2*, x*3) = m´ın *x*2 + *x*2 + *x*2 = m´ın **x***T* **x** = m´ın *||***x***||*2

*K*

*K*

1

2

3

*K*

*K*

2

*siendo* **x** = (*x*1*, x*2*, x*3)*T* R3 *el vector de inc´ognitas y K el conjunto convexo definido por las igualdades (ecuaciones) A***x** = **b** *siendo*

*∈*

1 1 1

 

*A* = 1 1 0

2*,*3

*b*1

*,* **b** = *b*2





2*,*1



*Resolver el problema describiendo expl´ıcitamente el Conjunto Admisible.*

Se trata de un problema en la formulaci´on ([77](#_bookmark232)). Empezamos estudiando el Conjunto Admisible.

##### Determinaci´on del conjunto admisible

La matriz *A* R*p×n* = R2*×*3 tiene rango 2 luego por la f´ormula de la dimen- si´on el nu´cleo de *A* tiene dimensi´on 1 y el sistema es compatible indetermi- nado para todo **b** R*p* = R2 siendo las soluciones **x** R*n* = R3. El conjunto admisible se puede por tanto describir a partir del conjunto de soluciones del sistema de restricciones. En concreto el sistema es suficientemente simple para obtener una parametrizaci´on mediante

*∈*

*∈ ∈*

*n − rg*(*A*) = 3 *−* 2 = 1

par´ametros. Se tiene

*x*1 + *x*2 + *x*3 = *b*1*, x*1 + *x*2 = *b*2

de donde

**x** = (*x*1*, x*2*, x*3)*T* = (*b*2 *− x*2*, x*2*, b*1 *− b*2)*T* = (*b*2*,* 0*, b*1 *− b*2)*T − α*(1*, −*1*,* 0)*T*

donde hemos elegido el par´ametro *α* = *x*2. Resolvemos ahora el problema utilizando las condiciones de KKT.

##### Resoluci´on via matriz de KKT

Para ello definimos la matriz *P* y los vectores **q** y **b**

 2 0 0 

*P* =  0 2 0 

0 0 2

3*,*3

 0 

0

*,* **q** =  0 

3*,*1

*b*1

*,* **b** = *b*2





2*,*1



Escribimos las condiciones KKT ([78](#_bookmark233))

 *P*3*,*3 *AT*

3*,*2

  **x***∗*3*,*1 

 *−***q**3*,*1 

 *A*2*,*3 **0**2*,*2   *ν*¯2*∗,*1   **b**2*,*1 

=

La matriz KKT es por tanto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 0 | 0 | 1 | 1 |
| 0 | 2 | 0 | 1 | 1 |
| 0 | 0 | 2 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

La matriz es no singular (tiene rango 5) y hay una u´nica soluci´on para todo **b**. Se puede calcular con cualquier m´etodo directo. Observa que la soluci´on del sistema pertenece a R5 siendo las primeras *n* = 3 componentes las variables primales de la soluci´on y las u´ltimas *p* = 2 coordenadas las variables duales. Veamos ahora la resoluci´on del problema mediante la consideraci´on de la lagrangiana.

##### Resoluci´on via Lagrangiana

Introduciendo *p* = 2 multiplicadores de Lagrange *ν*¯ Lagrangiana *L* : R3 *×* R2 *→* R es

*L*(**x***, ν*¯) = **x***T* **x** + *ν*¯*T* (*A***x** *−* **b**)

= (*ν*1*, ν*2)*T* la funci´on

con funci´on dual

*g*(*ν*¯) = ´ınf *L*(**x***, ν*¯)*.*

**x**

Imponiendo las condiciones de optimalidad de tipo KKT

que son

(*∇L*(**x***, ν*¯))*T* = (*∇***x***L*(**x***, ν*¯)*, ∇ν*¯*L*(**x***, ν*¯))*T* = **0** *∈* R*n*+*p*

*∇***x***L*(**x***, ν*¯) = 2**x** + *A ν*¯ = **0***, ∇ν*¯*L*(**x***, ν*¯) = *A***x** *−* **b** = **0**

*T*

se tienen los puntos ´optimos primal y dual

**x***∗* = 1 *AT ν*¯*∗,* 2

*−*

*ν*¯*∗* = *−*2(*AAT* )*−*1**b**

de donde la f´ormula m´as compacta dada por la pseudo-inversa de Moore-

Penrose (v´ease pg 649 del Boyd) v´alida cuando *rg*(*A*) = *p*

**x***∗* = *AT* (*AAT* )*−*1**b** = *A†***b**

Observa que si *rg*(*A*) = *n* entonces

*A†* = (*AAT* )*−*1*AT .*

En general Tenemos que empezar por el dual. Se tiene

 1 1 1   1 1 

 3 2 

*AAT* = 

1 1 0

  1 1  =  2 2 

1 0

La matriz cuadrada sim´etrica *AAT* es no singular ya que

*det*(*AAT* ) = 2 *̸*= 0 luego invertimos para obtener

(*AAT*

)*−*1 =

1  2 *−*2 

luego el o´ptimo dual es

2 *−*2 3

*\* T −*1

 *−*2 2

  *b*1

 *−*2*b*1 + 2*b*2 

 *ν*1 

*ν*¯ = *−*2(*AA* ) **b** =

2 *−*3   *b*2

 = 

2*b*1 *−* 3*b*2

 =  *ν*2 

y se calcula el ´optimo primal

 1 1   *ν* 

1

1

1

 *ν*1 + *ν*2   *b*2 

*\** 1 *T ∗*  

**x**

= *−* 2 *A*

*ν*¯

= *−* 2  1 1   *ν*

2

1 0

 *ν* + *ν*   

*ν*1

##### C´alculo de la energ´ıa primal en el o´ptimo

1

=

 = *−* 2 

1

2

 2 

*b*2

*−*2*b*1 + 2*b*2



Empezamos por el c´alculo de la energ´ıa primal en el ´optimo. Se tiene:

*p∗* = *||***x***∗||*2 = (**x***∗*)*T* **x***∗*=

2

4

1 (*b , b , −*2*b* + 2*b* ) 

*b*2 *b*2

*−*2*b*1 + 2*b*2

 =

= 1 *b*2 + (*b*

2

2

2 3 2

*— b* ) = *b −* 2*b b*

2

1

2

2

1

+ *b*2

2 2 1 2



Por ejemplo, si **b** = (2*,* 2)*T* se tiene

2

1

*ν*¯*∗* = (0*, −*2)*T ,* **x***∗* = (1*,* 1*,* 0)*T*

con valor o´ptimo primal *p∗* = 2. Vemos que esta soluci´on est´a en la familia uniparam´etrica tomando *α* = 1. En efecto la familia de soluciones del sistema (conjunto admisible) es

**x** = (*b*2 *− α, α, b*1 *− b*2)*T , α ∈* R

y su energ´ıa es

2

*||***x***||*2 = (*b*2 *− α*) es decir el polinomio cuadr´atico

2

2

+ *α*2

+ (*b*1 *− b*2)

*p*(*α*) = (*b*2 *− α*)2 + *α*2 + (*b*1 *− b*2)2

Derivando e imponiendo la condici´on de primer orden se tiene

*p′*(*α*) = *−*2(*b*2 *− α*) + 2*α* = 0

de donde el valor del par´ametro en donde se alcanza la energ´ıa m´ınima:

*α* = *b*2*/*2*.*

Si *b*2 = 2 se tiene *α* = 1 y reencontramos el resultado anterior. Adema´s

recordamos que **p***∗* = *||***x***∗||*2 = 2. Veamos si hay dualidad fuerte entre el

2

problema primal y el dual.

##### C´alculo de la energ´ıa dual en el o´ptimo

Utilizando la teor´ıa sabemos que la dualidad fuerte entre los problemas primal y dual se tiene si los valores en el o´ptimo son iguales: *p∗* = *d∗*. Para calcular *d∗*, el valor m´aximo de la energ´ıa dual (concava) razonamos como sigue: la funci´on dual es, por definici´on

*g*(*ν*¯) = ´ınf *L*(*−*(1*/*2)*AT ν*¯*, ν*¯) = *−* 1 *ν*¯*T AAT ν*¯ *−* **b***T ν*¯ =

**x**

= *−* 3 *ν*2 *− ν ν*

4

1

1

4

1 2

* *ν − b ν*

2

2

2

1

* *b ν*

y es una funci´on concava (estudiar el espectro de la matriz que la define).

1

2

2

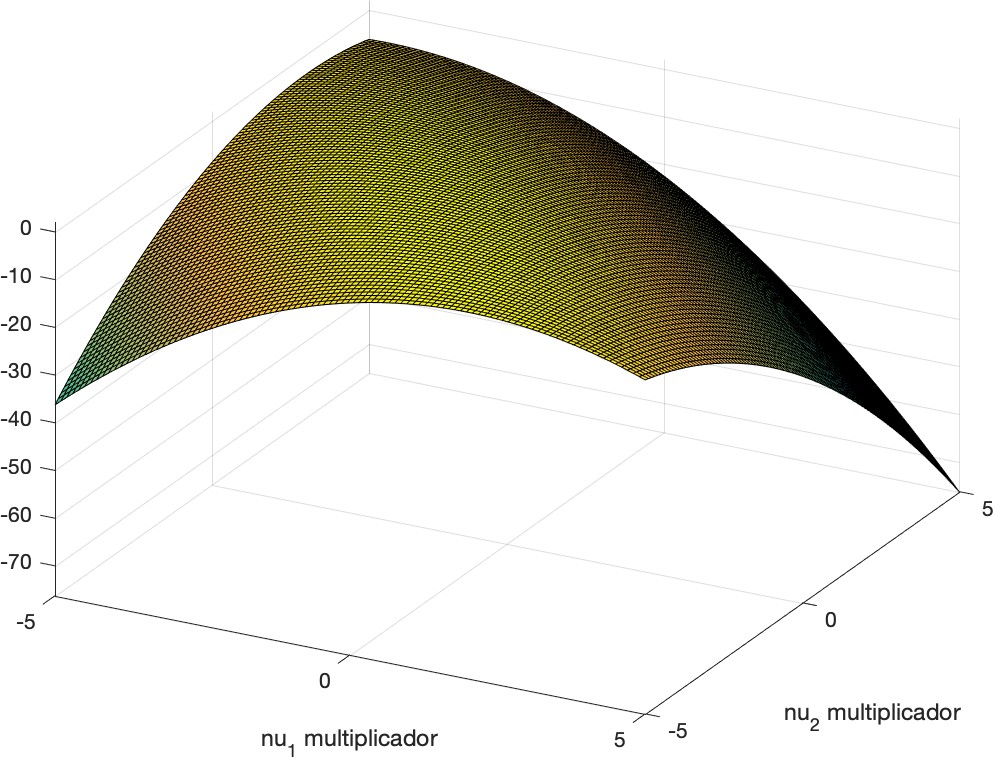


Figura 37: Gr´afica de la Funci´on dual de Lagrange. Se observa que es concava.

Calculamos su gradiente e imponemos las condiciones de primer orden:

*∇g*(*ν*¯) = (*−* 3 *ν*

2

1

* *ν*2

*— b*1*, −ν*1

* *ν*2

*— b*2 = (0*,* 0)

de donde el sistema homog´eneo

3

2 *ν*1 *− ν*2 *− b*1 = 0*, −ν*1 *− ν*2 *− b*2 = 0

que tiene la u´nica soluci´on

*\**  *ν*1 

1  2*b*1 *−* 2*b*2 

*ν*¯ =  *ν*2

 = 5  *−*2*b*1 *−* 3*b*2 

Para *b*1 = *b*2 = 2 se tiene *ν*1 = 0, *ν*2 = *−*2, es decir

*ν*¯*∗* = (*ν*1*, ν*2)*T* = (0*, −*2)

y la funci´on dual toma su valor m´aximo

*d∗* = *g*(*ν*¯*∗*) = 2 = m´ax *g*(*ν*¯)

*ν*¯

luego *d∗* = *p∗* = 2*∗* y hay dualidad fuerte ya que la condici´on de Slater que el problema primal sea admisible se verifica.

##### Problemas Cuadr´aticos

Vamos a resolver unos problemas de Programaci´on Cuadr´atica. Empezamos resolviendo con matlab el ejemplo anterior

**Ejercicio 15.13.** *Sea dado el problema de Minimizaci´on Cuadr´atica con Restricciones de igualdad Lineales*

m´ın *f*0(*x*1*, x*2*, x*3) = m´ın *x*2 + *x*2 + *x*2 = m´ın **x***T* **x** = m´ın *||***x***||*2

*K*

*K*

1

2

3

*K*

*K*

2

*siendo* **x** = (*x*1*, x*2*, x*3)*T* R3 *el vector de inc´ognitas y K el conjunto convexo definido por las igualdades (ecuaciones) A***x** = **b***, es decir*

*∈*

*K* = *{***x** *∈* R3 *| A***x** = **b***}*

*siendo*

 1 1 1 

2

 

*,* **b** =  2 

*Resolver el problema utilizando matlab. Puedes hacerlo resolviendo expl´ıcita- mente las condiciones KKT y/o utilizar el comando* **quadprog.m***. Comparar la soluci´on con la obtenida an´aliticamente en el ejemplo anterior.*

1 1 0

*A* =

A=[ 1 1 1;

1 1 0]

P=2\*eye(n);

Z=zeros(p);

M=[P A’; A Z]

bb=[0 0 0 2 2]’;

sol=M\bb xp=sol(1:n) nud=sol(n+1:end)

A=[];B=[];

H=P;

f=[]; lb=[]; ub=[];

Aeq=[ 1 1 1;

1 1 0];

Beq=[2 2]’;

X0=[0 0 0]’;

options = optimoptions(’quadprog’,... ’Algorithm’,’interior-point-convex’,’Display’,’off’);

[X,fval,exitflag,output,lambda] = quadprog(H,f,A,B,Aeq,Beq,lb,ub,X0,options) lambda.eqlin

**Ejercicio 15.14.** *Sea dado el problema de Minimizaci´on Cuadr´atica con Restricciones Lineales*

m´ın *f* (*x, y*) = m´ın *x*2 + *y*2

*K*

*K*

*siendo K el conjunto convexo definido por las igualdades A***x** = **b** *siendo*

 1 2   1 

   

*A* = 0 0 *,* **b** = 0

*Resolver el problema utilizando matlab. Puedes hacerlo resolviendo expl´ıcita- mente las condiciones KKT y/o utilizar el comando* **quadprog.m***. Comparar la soluci´on con la obtenida an´aliticamente.*

##### Soluci´on anal´ıtica

Empezamos estudiando el conjunto de admisibilidad. Se reduce a los puntos

**x** = (*x, y*)*T* tales que

*x* + 2*y* = 1

de donde se deduce que las infinitas soluciones del sistema que constituyen el conjunto de admisibilidad son:

(*x, y*)*T* = (1 *−* 2*y, y*)*T* = (1*,* 0)*T* + *y*(*−*2*,* 1)*T*

siendo (1*,* 0)*T* + una soluci´on particular y ( 2*,* 1)*T* una base del nu´cleo de la aplicaci´on asociada a la matriz *A*:

*−*

 1 2   *x* 

  

*fA*(**x**) = *A***x** = 0 0 *y*

Las condiciones de optimalidad de KKT son

 = *x* + 2*y*

 2*x* 

 1 0   *ν*1 

 0 

 2*y*

 +  2 0   *ν*2

 =  0 

de donde el sistema de ecuaciones

 2*x* + *ν*1 = 0

2*y* + 2*ν*1 = 0



Resolviendo se tiene *y* = 2*x* luego el sistema de KKT tiene infinitas soluciones del tipo (*x,* 2*x*). Puesto que se tiene que verificar la restricci´on imponemos

 1 2   *x*   1 

     

0 0 2*x* = 0

de donde *x* = 1*/*5, *y* = 2*x* = 2*/*5. Se tiene adem´as *p∗* = 1*/*5 y el multiplicador de Lagrange *ν*1 = *−*2*/*5. La energ´ıa dual (funci´on dual de Lagrange) es

5 2

*g*(*ν , ν* ) = *−ν − ν*

1

2

1

4

1

que tiene m´aximo en *ν*1 = 2*/*5 en donde *d∗* = *g*( 2*/*5) = 1*/*5 luego *p∗* =

*— −*

*d∗* = 1*/*5 y hay dualidad fuerte.

**Ejercicio 15.15.** *Sea dado el problema de Minimizaci´on Cuadr´atica con Restricciones Lineales*

m´ın *f* (*x, y*) = m´ın (1 *x*2 + *y*2 *− xy −* 2*x −* 6*y*

*K*

*K*

2

*siendo K el conjunto convexo definido por las desigualdades*

**** *x* + *y ≤* 2

 *−x* + 2*y ≤* 2

*K* = **** 2*x* + *y ≤* 3

**** *x, y ≥* 0

*Resolver el problema utilizando matlab.*

**Ejercicio 15.16.** *Sea dado el problema de Optimizaci´on Cuadr´atica con Res- tricciones Lineales*

m´ın *f* (*x, y*) = m´ın (1 *x*2 + 3*x* + 4*y*

*K*

*K*

2

*siendo K el conjunto convexo definido por las desigualdades*

*x, y ≥* 0

****

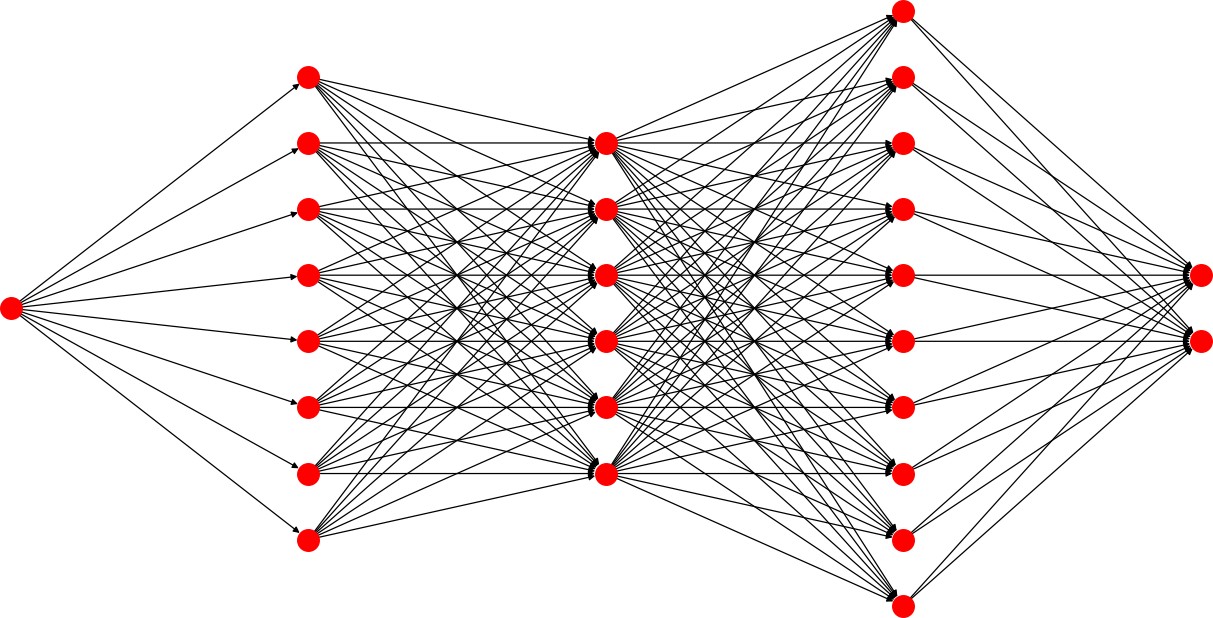
*K* = *x* + 3*y ≥* 15

**** 3*x* + 4*y ≤* 80

*Escribir el problema en forma estandard. Resolver el problema utilizando matlab. Verificar la soluci´on* (*x∗, y∗*) = (0*,* 5) *con valor ´optimo p∗* = 20*.*

# Ma´ster en Visi´on Artificial

### Curso 2023-2024



**APUNTES DE FUNDAMENTOS MATEMA´TICOS: III PARTE - APRENDIZAJE PROFUNDO**

### Autores:

Iv´an Ram´ırez, Victoria Ruiz, Emanuele Schiavi

©2023 Iv´an Ram´ırez D´ıaz, Victoria Ruiz Parrado, Emanuele Schiavi. Al- gunos derechos reservados. Este documento se distribuye bajo la licencia “Atribuci´on-CompartirIgual 4.0 Internacional” de Creative Commons,

disponible en <https://creativecommons.org/licenses/by-sa/4.0/deed.es>.

## ´Indice

1. [**XV Clase**](#_bookmark235) **4**
   1. [Introducci´on](#_bookmark236) 4
      1. [Evidencia](#_bookmark237) 4
      2. [Modelo](#_bookmark239) 5
      3. [M´etodo](#_bookmark240) 7
   2. [Deep Learning](#_bookmark242) 8
   3. [Maximun a Posteriori - Planteamiento del problema](#_bookmark244) 9
      1. [Regresi´on](#_bookmark246) 10
      2. [Clasificaci´on](#_bookmark248) 12
      3. [Funci´on de coste *log´ıstica* - Caso particular](#_bookmark249) 13
      4. [Regularizaci´on](#_bookmark250) 14
2. [**XVI Clase**](#_bookmark251) **16**
   1. [Redes Neuronales (NN)](#_bookmark252) 16
3. [**XVII Clase**](#_bookmark257) **20**
   1. [Introducci´on a Pytorch](#_bookmark258) 20
4. [**XVIII Clase**](#_bookmark259) **20**
   1. [Redes Convolucionales(CNN)](#_bookmark260) 20
   2. [Convoluci´on como Operaci´on Matricial](#_bookmark263) 22
5. [**XIX Clase**](#_bookmark265) **25**
   1. [Introducci´on a Pytorch II](#_bookmark266) 25
6. [**XX Clase**](#_bookmark267) **25**
   1. [Descenso de Gradiente - Backpropagation](#_bookmark268) 25
   2. [Forma Matricial](#_bookmark269) 32
7. [**XXI Clase**](#_bookmark270) **34**
   1. [Optimizadores](#_bookmark271) 34
   2. [Inferencia Bayesiana](#_bookmark272) 37
   3. [Par´ametros Deterministas vs Estoc´asticos](#_bookmark279) 40
   4. [Inferencia Variacional](#_bookmark280) 41
   5. [Dropout](#_bookmark283) 43

|  |  |  |
| --- | --- | --- |
| [**8. XXII Clase**](#_bookmark284) |  | **45** |
| [8.1. II Pr´actica en Pytorch.](#_bookmark285) | . . . . . . . . . . . . . . . . . . . . . | 45 |
| [8.1.1. Denoising con TV](#_bookmark286) 45 | | |
| [**9. XXIII Clase**](#_bookmark287) |  | **45** |
| [9.1. III Pr´actica en Pytorch.](#_bookmark288) | . . . . . . . . . . . . . . . . . . . . . | 45 |
| [**10.XXIV Clase**](#_bookmark289) |  | **45** |
| [10.1. IV Pr´actica en Pytorch.](#_bookmark290) | . . . . . . . . . . . . . . . . . . . . . | 45 |

## XV Clase

#### Introduccio´n

En esta introducci´on daremos una visi´on simple y global del problema que se plantea en Aprendizaje Autom´atico o Machine Learning (ML). M´as adelante, daremos una interpretaci´on Bayesiana del mismo problema, que nos permi- tir´a dar justificaciones y fundamentos a decisiones que en la formulaci´on del propio problema pueden parecer arbitrarias. Para entender correctamente los entresijos del Aprendizaje Autom´atico es necesario primero definir de manera concreta el problema al que nos enfrentamos. Aprender de forma autom´ati- ca es, por tanto, dotar a una m´aquina de las herramientas necesarias para realizar una tarea. En general, distinguiremos 2 categor´ıas de tares:

Clasificaci´on: *estimar la clase a la que pertenece un objeto*

Regresi´on: *estimar el valor que le corresponde a un objeto*

Estos objetos ser´an, en general, vectores o puntos en un espacio *n*-dimensional (R*n*) dependiendo de la estructura de los mismos. Cada componente de dicho vector puede ser interpretado como una caracter´ıstica. Tendremos, por tanto, vectores de caracter´ısticas. Al conjunto de objetos a procesar los llamaremos datos o evidencia.

##### Elementos en ML

En general, se pueden distinguir 3 elementos que van a aparecer siempre en un problema de ML:

1. Evidencia: son los datos que tenemos fruto de la experiencia
2. Modelo: es la **hip´otesis** de la ley o modelo que rige y explica los datos (Evidencia).
3. M´etodo: es la manera de encontrar la hip´otesis que mejor explica los datos.

##### Evidencia

{ }

Sea **X** = **x**(1)*,* **x**(2)*, ...***x**(*N*) un conjunto de elementos (datos u objetos) que corresponden, cada uno de ellos, a una etiqueta del conjunto **Y** = **y**(1)*,* **y**(2)*, ...***y**(*N*) , donde *i* = 1*, .., N* . Cada elemento de **X** e **Y** es, en general, un vector: **x**(*i*) *∈* R*m*, **y**(*i*) *∈* R*n*.

{ }

##### Clasificaci´on

En el caso de clasificaci´on, los posibles valores de cada uno de los **y**(*n*)

*∈*

*a, b, c, ...* ser´an discretos. Es habitual que estas etiquetas tomen un valor entero por cada clase, en cuyo caso tendremos un escalar por cada etiqueta. Tambi´en es comu´n usar diferentes codificaciones como, por ejemplo, la *One- hot*. En la tabla [1](#_bookmark238) vemos un ejemplo de codificaci´on *One-hot* para el caso binario.

*{ }*

Clasificaci´on

**x**(*i*) **y**(*i*)

i=1 (*−*1*,* 3*.*2)*T* 0

Clasificaci´on One-hot

**x**(*i*) **y**(*i*)

(*−*1*,* 3*.*2)*T* (1*,* 0*,* 0)*T*

Regresi´on

**x**(*i*) **y**(*i*)

(0*.*1*,* 0*.*2)*T* 0.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i=2  ... | (7*,* 0*.*2)*T*  ... | 1  ... | (7*,* 0*.*2)*T*  ... | (0*,* 0*,* 1)*T*  ... | (0*.*4*,* 0*.*3)*T*  ... | 0.2  ... |
| i=N | (0*.*6*,* 4*.*9)*T* | 1 | (0*.*6*,* 4*.*9)*T* | (0*,* 0*,* 1)*T* | (0*.*6*,* 2*.*6)*T* | 0.8 |

Cuadro 1: Ejemplos de datos para clasificaci´on y regresi´on.

##### Regresi´on

Por el contrario, si a cada elemento **x**(*i*) le corresponde un valor o valores continuos, los valores de **Y** ser´an valores continuos definidos en un rango **y**(*i*) [*a, b*]. En la tabla [1](#_bookmark238) el valor regresado es un escalar.

*∈*

Tenemos, pues, un conjunto de pares de elementos que constituyen nuestros datos o evidencia:

Evidencia: *D* = *{***X***,* **Y***}*

##### Modelo

Ahora definimos el problema de ML como:

***Encontrar una funcio´n f*** : R*m* R*n tal que* ***Y*** = ***f***(***X***)*. Es decir, para todo elemento* ***x***(*i*) *se cumple que:*

*→*

***f***(***x***(*i*)) = ***y***(*i*)*.*

El problema de Machine Learning es encontrar, o en su defecto como veremos a continuaci´on, aproximar una funci´on.

Somos conscientes de que existen infinitas funciones y que probar una a una no es posible. Por ello, definimos un conjunto de funciones que llamaremos

*tratables*, es decir, consideraremos un subconjunto de funciones que podamos manejar. Este subconjunto podr´ıa ser una lista de las funciones m´as comu- nes y que todos conocemos: sin( )*,* cos( )*,* log( )*, etc*. No obstante, seguimos teniendo que probar una a una si realizan correctamente la transformaci´on que buscamos, lo que es poco pr´actico, o sencillamente inabordable. Por suer- te, tenemos herramientas matem´aticas que nos solventan este problema sin m´as que proponer una *funci´on param´etrica* gen´erica que pueda aproximar ’casi cualquier’ funci´on tan solo cambiando sus par´ametros. Un ejemplo de aproximaci´on polin´omica muy conocido son las series de Taylor.

*· · ·*

Dentro del ML, existen unas t´ecnicas conocidas como Aprendizaje Profundo o Deep Learning (DL). Estas t´ecnicas consisten en hacer uso de unos mo- delos llamados Redes Neuronales (Artificiales). Existe una amplia variedad de redes neuronales: convolucionales, de c´apsulas, recurrentes... pero todas ellas son *funciones universales de aproximaci´on*. El problema de aproximar una funci´on no es para nada nuevo y es de esperar bastantes herramientas y fundamentos matem´aticos para abordarlo. En los u´ltimos an˜os, ha habido un incremento en el inter´es de este tipo de problemas debido a las mejoras en la capacidad de c´omputo de los ordenadores junto con el desarrollo de algorit- mos m´as eficientes. Muchas de las t´ecnicas que se aplican hoy en d´ıa dando resultados muy competitivos se conocen desde hace siglos, sin embargo, eran computacionalmente inaccesibles hasta el desarrollo de nuevos algoritmos y de computadores m´as potentes.

No obstante, incluso a nivel de fundamentos, no todo est´a resuelto. Las fun-

ciones que queremos estimar se obtienen con muchos ejemplos que esperamos sean lo suficientemente buenos para generalizar. Si bien, desde un punto de vista matem´atico, sabemos qu´e hacer para obtener una funci´on que aproxima unos datos, la novedad en estos tiempos, es que los datos est´an en muy alta dimensi´on, ralentizando dr´asticamente su obtenci´on.

La clave de las redes neuronales y su ´exito es triple:

* + - 1. Obtener muchos ejemplos de la tarea a realizar (datos)
      2. Definir una funci´on (estructura)
      3. Encontrar sus par´ametros o´ptimos (entrenamiento)

Debemos entonces pensar con base en estos tres ejes a la hora de proponer una soluci´on a un problema de ML: qu´e datos obtener, qu´e arquitectura de red (funci´on) usar y c´omo entrenarla.

Veamos un ejemplo.

Supongamos que queremos predecir la contaminaci´on en Madrid dados dos valores: la temperatura y la hora. Es decir:

(**x**(*i*) = (*x*1*, x*2)*,* **y**(*i*) = *y*1)

donde *x*1 es el valor correspondiente a la temperatura, *x*2 es la hora e *y*1 el nivel de contaminaci´on. Buscamos una funci´on param´etrica que combine las entradas para obtener la salida:

**f**(***ω****,* **x**(*i*))

donde ***ω*** = *{w*1*, w*2*, ..., wk}* es un vector de *k* par´ametros, por ejemplo:

**f***model*(***ω****,* **x**(*i*)) = *w*5*x*2 + *w*4*x*2 + *w*3*x*1 + *w*2*x*2 + *w*1*x*1*x*2 + *w*0

1 2

entonces, *f* es un modelo param´etrico de la funci´on ideal que buscamos. En este caso concreto, la funci´on es un polinomio de orden 2. Tenemos:

Evidencia: *D* = *{***X***,* **Y***}*

Modelo: **f***model*(***ω****,* **x**(*i*)) de 6 par´ametros.

Este modelo, que asumimos capaz de aproximar correctamente **f** con los par´ametros bien ajustados, constituye nuestra hip´otesis.

##### M´etodo

Nos queda un u´ltimo elemento: el m´etodo para encontrar los par´ametros ade- cuados y aproximar *f* , es decir, los par´ametros o´ptimos para que se cumpla:

**Y** = **f**(***ω****,* **X**)

que equivale a:

**Y** *−* **f**(***ω****,* **X**) = 0*.*

En general, podemos elegir arbitrariamente una funci´on de *coste* ( ) que cumpla la condici´on anterior o, que minimice la diferencia entre ambos t´ermi- nos, lo que se modela definiendo un funcional de coste (**Y***,* **f**(***ω****,* **X**)).

*C*

*C ·*

Habitualmente esta funci´on de coste es referida a todo el dataset o conjunto de datos de entrenamiento, llamando funci´on de *p´erdida* (loss function) al error que se pretende minimizar para cada ejemplo.

Funci´on d e P´erdida

1

*C* (**Y***,* **f**(***ω****,* **X**)) =

Funci´on de Coste

*N*

*L*(**y**(*i*)*,* **f**(***ω****,* **x**(*i*))) *.*

*i*

Si elegimos como funci´on de coste el comu´nmente usado *Error de M´ınimos Cuadrados* (MSE), el problema de optimizaci´on entonces ser´a:

***ω****∗* = arg m´ın *C*(**Y***,* **f**(***ω****,* **X**)) = arg m´ın 1

***ω***

***ω***

*N*

*||***y**(*i*) *−* **f**(***ω****,* **x**(*i*))*||*2*.* (1)

*i*

Recordemos que el objetivo en ML es aprender a clasificar o regresar con base en ejemplos. Para ello, necesitamos encontrar una funci´on que aproxime esta transformaci´on de datos de entrada en datos de salida, es decir, resolver un problema de optimizaci´on. Una vez m´as acudimos a las matem´aticas para obtener herramientas que nos permitan resolver los problemas de optimiza- ci´on. Con todo ello, tendr´ıamos los elementos y herramientas para resolver un problema de ML.

2

Evidencia: *D* = *{***X***,* **Y***}*

Modelo: **f***model*(***ω****,* **x**(*i*)) de 6 par´ametros.

M´etodo: optimizaci´on de un funcional de energ´ıa o funci´on de *coste*, Eq. [1](#_bookmark241).

#### Deep Learning

Como coment´abamos, el Deep Learning o Aprendizaje Profundo es una t´ecni- ca particular dentro del campo del Aprendizaje Autom´atico que emplea al- gunos elementos concretos que han permitido obtener resultados que superan por mucho el estado del arte de las t´ecnicas precedentes. No obstante, el ob- jetivo sigue siendo el mismo que se define en el paradigma del Aprendizaje Automa´tico: encontrar una funci´on **f** que transforme los datos de entrada en los datos de salida de forma correcta. Lo que hace al Deep Learning intere- sante es la caracterizaci´on de la funci´on **f**, es decir, la familia de funciones que se usan en Deep Learning y que han dado buenos resultados, que se conocen como **Feedforward Networks**, as´ı como el uso de un algoritmo llamado Back-Progation que veremos m´as adelante. Dichas funciones tienen una estructura de red neuronal (en realidad **inspiradas** en las redes neuro- nales del cerebro humano), de tal modo que la informaci´on fluye a trav´es de ellas siempre hacia adelante.

La manera en que se estructuran estas redes da nombre al tipo de aprendizaje,

es este caso, **profundo**, pues entre la entrada a la red (funci´on) y la salida, existen distintos niveles o capas. De forma irrelevante se suele considerar una

red como profunda cuando tiene 3 o m´as capas. Esto es sin duda arbitrario, pero nos da pistas del porqu´e de su ´exito. En realidad, el Deep Learning ha tomado protagonismo cuando se ha tenido la t´ecnica y capacidad para ajustar los par´ametros de redes neuronales profundas.

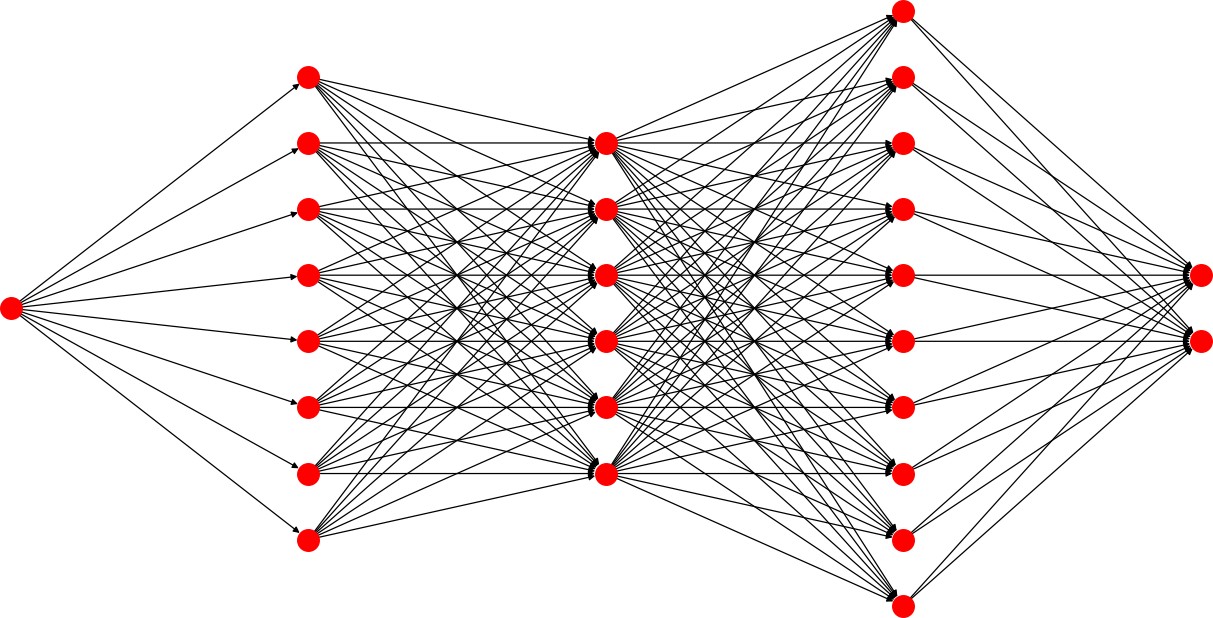


Figura 1: Red Neuronal

La estructura neuronal en la figura ([1](#_bookmark243)) representa una funci´on que se define a su vez como composici´on de otras funciones: **f** = **f**3(**f**2(**f**1(**x**))). Cuantas m´as funciones compongan la funci´on global, m´as profunda ser´a la red que modelemos y m´as capas tendr´a. Sin entrar todav´ıa en los detalles de c´omo se definen estas funciones, consideraremos una familia de funciones **f**(***ω****,* **x**) que dependen de unos par´ametros ***ω*** para definir el problema de ML desde un punto de vista Bayesiano.

#### Maximun a Posteriori - Planteamiento del proble- ma

Para encontrar la funci´on **f** que transforma los datos **X** en **Y**, debemos prime- ro definir una funci´on de coste que evalu´a c´omo de correctas son las predic- ciones realizadas por **f**. Es decir, nos da una medida del error que se comete o coste, y, por tanto, el objetivo ser´a minimizar dicho coste. Podemos in- tuitivamente decidir minimizar una distancia *d*(**y***,* **y**ˆ) entre las predicciones

**y**ˆ = **f**(***ω****,* **x**) y las etiquetas reales **y** o, por el contrario, deducir de forma

fundamentada, a partir de Bayes, cu´ales son las distancias a minimizar. Es lo que haremos a continuaci´on.

Sea **X** = {**x**(1)*,* **x**(2)*, ...***x**(*N*)} un conjunto de elementos (dat{os) que correspon}-

den cada uno de ellos a una etiqueta del conjunto **Y** =

**y**(1)*,* **y**(2)*, ...***y**(*N* )

,

donde *i* = 1*, .., N* . Cada elemento de **X** e **Y** es, en general, un vector: **x**(*i*) *∈* R*m*, **y**(*i*) *∈* R*n*. Usando un esquema MAP, definimos primero la **pro- babilidad condicionada *a posteriori*** *p*(***ω*** *|* **X***,* **Y**) como:

*p*(***ω* X***,* **Y**) = *p*(**Y** *|* ***ω****,* **X**)*p*(***ω***)

*|*

*p*(**Y** *|* **X**)

(2)

en donde *p*(***ω* X***,* **Y**) es la **verosimilitud** o likelihood y *p*(***ω***) el **a priori** o prior. Dependiento de c´omo definamos cada t´ermino de la equaci´on ante- rior, estaremos considerando un problema concreto. Empezaremos por el de regresi´on.

*|*

##### Regresi´on

Partimos de la equaci´on ([2](#_bookmark245)) donde consideraremos la **verosimilitud** y el **a priori** como gaussianos:

*p*(**Y** *|* ***ω****,* **X**) =

1 (2*πσ*2) *N*

exp (*−*

2

*||***Y** *−* **f**(***ω****,* **X**)*||*

2

2*σ*2

1 2 1

*||****ω****||*

*p*(***ω***) =

1

(2*πσ*2) *K*

exp (*−*

2

2

2*σ*2

2 2 2

y ***ω*** = ***ω***1*,* ***ω***2*, ...,* ***ω****k* son los *K* par´ametros de los que se compone **f**. No´tese que en estas definiciones se asumen que cada uno de los ejemplos en **X***,* **Y** son i.i.d. al igual que cada uno de los par´ametros en ***ω***. M´as en detalle:

*{ }*

*p*(**Y** *|* ***ω****,* **X**) =

*N*

*i*

1

*√*2*πσ*

1

exp

(*i*) (*i*) 2

2

*||***y** *−* **f**(***ω****,* **x** )*||*

(

*−*

2*σ*2

1

1

=

(2*πσ*2) *N*

exp

1

*−* 2*σ*2

*N*

*||***y** *−* **f**(***ω****,* **x** )*||*2

1 2 1 *i*

(*i*) (*i*) 2

1 (*||****ω****||*2 1 1

*K*

*K*

2

2

*p*(***ω***) =

*√*

exp

=

2

*K*

exp

*−* 2*σ*2

*||****ω****i||*2

2

*i*

2*πσ*2

2*σ*2

(2*πσ*1 ) 2

1

*i*

Segu´n la regla MAP, el problema de maximizaci´on consiste en encontrar los par´ametros o´ptimos ***ω****∗* que maximizan dicha probabilidad

***ω****∗* = arg m´ax *p*(***ω* X***,* **Y**)*.* (3)

*|*

***ω***

Tomando log( ) a la probabilidad *a posteriori*, re-escribimos el problema como:

*— ·*

siendo *J*(***ω***):

***ω***

***ω****∗* = arg m´ın *{J*(***ω***)*}* (4)

*J*(***ω***) = 1 *||***Y** *−* **f**(***ω****,* **X**)*||*2 + log((2*πσ*2 *N*

2*πσ*2

1

2 1 ) 2 )

+ 1 2 2 *K*

2*πσ*2 *||****ω****||*2 + log((2*πσ*2 ) 2 )

2

Simplificando,

*—* log(*p*(**Y** *|* **X**))*.*

*J*(***ω***) = 1 *||***Y** *−* **f**(***ω****,* **X**)*||*2 + *N* log(*σ* ) + 1 *||****ω****||*2 + *K* log(*σ* )

2*πσ*2

1

2

2 1 2*πσ*2 2 2

*—* log(*p*(**Y** *|* **X**)) (5)

y prescindiendo de los t´erminos que no depende de ***ω***, tenemos un problema de minimizaci´on variacional como sigue:

***ω****∗* = arg m´ın 1 *||***Y** *−* **f**(***ω****,* **X**)*||*2 + 1 *||****ω****||*2 (6)

1

2

***ω***

2*πσ*2

2

2*πσ*2

2

Si multiplicamos todo el funcional por 2*πσ*2, y re-nombramos *λ*1 = 2*πσ*2*/*2*πσ*2

1 1 2

***ω****∗* = arg m´ın {*||***Y** *−* **f**(***ω****,* **X**)*||*2 + *λ*1*||****ω****||*2} (7)

***ω***

2

2

es f´acil identificar estos dos t´erminos como el de fidelidad de datos y regu- larizaci´on (*L*2 en ese caso) sobre los par´ametros. Por completitud, volvemos a re-escribir esta expresi´on para mostrar que es ex´actamente la utilizada en habitualmente en Deep Learning con *λ* = *λ*1*K* :

*N*

***ω****∗* = arg m´ın

***ω***

1 1

*N ||***y** *−* **f**(***ω****,* **x** *||*2 + *λK*

*N*

(*i*) (*i*) 2

*K*

*i i*

*||****ω****i||*2

*.* (8)

2

El primer t´ermino de la minimizaci´on es el conocido *Error Cuadr´atico Medio* MSE y el t´ermino de regularizaci´on *L*2 o de Tikhonov, tambi´en referido en la literatura como *weight decay* por su formulaci´on en la Euler-Lagrange como veremos m´as adelante.

Hasta ahora hemos derivado la formulaci´on del problema de ML definiendo un error gausiano entre las salidas **y**(*i*) y la predicciones **f**(***ω****,* **x**(*i*)). El mismo desarrollo se puede llevar a cabo para un problema de clasificaci´on sin m´as que definir la **verosimilitud** con una funci´on de distribuci´on tipo *softmax*.

##### Clasificaci´on

En clasificaci´on, la salida de una red, es decir, **f**, se asume vectorial en la que cada componente, que corresponde a una clase, representa la *activaci´on* de dicha clase. La funci´on de distribuci´on que se suele usar en clasificaci´on es la *softmax* :

*e***f**

Softmax(**f**) =

¿

*j*

*efj ,*

una funci´on que tiene como entrada variables no acotadas, es decir, *fj* R. A las entradas de la funci´on *Softmax* se las suele llamar *logits*, y es por ello que en clasificaci´on, a las salidas de una red no se le aplican ninguna funci´on de activaci´on previa. Dichas componentes *fj* se transforman de tal forma que la suma de todas ellas es igual a 1, pudiendo interpretar cada una de ellas como una probabilidad. Por ejemplo:

*∈*

**f** = 

2

1

0*.*1

 *→*

 0*.*7 

0*.*1

*Softmax*(*·*)

 ˆ 0*.*2*→* **y** = *.*

Manteniendo el **a priori** gausiano, la probabilidad *a posteriori* sobre los par´ametros ***ω*** es

*p*(***ω*** *|* **X***,* **Y**) *∝* Softmax(**f**(***ω****,* **X**))*p*(***ω***)*.*

Ahora, usando el esquema MAP, queremos maximizar la probabilidad de

aquellas salidas que coinciden con la etiqueta correcta, es decir, que **y**ˆ =

*Softmax*(**f**(***ω****,* **x**) coindica lo m´as posible con la codificaci´on *One-hot* de las etiquetas **y**.

 0*.*7   1 

**y**ˆ =  0*.*2  *≈* **y** =  0  *.*

0*.*1

0

No´tese que, maximizar la salida correcta implica minimizar las probabilidades de las dem´as. Dicho de otra forma, este tipo de enfoque considera clases mutuamente excluyentes. Por lo tanto, el objetivo a maximizar en el t´ermino de **verosimilitud** ser´a

*p*(*yc |* ***ω****,* **x**) = **y**ˆ*c* = *Softmax*(**f**(***ω****,* **x**)*c*

donde *yc* es la componente *correcta* asociada a la entrada **x**. Tomando log( ) tenemos:

*— ·*

*n*

*—* log(*p*(*yc |* ***ω****,* **x**)) = *−* **1** *{y* = *c}* log(*y*ˆ*j*)

*j*=1

( *efj* (***ω****,***x**)

*n*

¿

= *−*

**1** *{y* = *c}* log

*j*=1

*n*

*k*=1

*efk*(***ω****,***x**)

donde **1** *{·}* es la funci´on caracter´ıstica:

*{ }*

**1** *x* = 1*,* si *x* es cierto

0*,* si *x* es falso

Finalmente, el funcional de energ´ıa o funci´on de coste a minimizar es:

(9)

1 *n* ( *efj* (***ω****,***x**)

*N*

¿

*J*(***ω***) = *−N*

*i*

*j*=1

**1** *{y* = *c}* log

1

+

*n k*=1

2

*efk*(***ω****,***x**)

2 *K*

2*σ*2 *||****ω****||*2 + log((2*πσ*2 ) 2 )*.* (10)

2

* + 1. **Funci´on de coste *log´ıstica* - Caso particular**

Un caso particular que se puede derivar de la funci´on de distribuci´on *Softmax* es la funci´on de coste *log´ıstica*, muy usada en el caso de clasificaci´on binaria, es decir, cuando usamos *Softmax* para vectores **f** *∈* R2. Es decir,

( *ef*1(***ω****,***x**) *ef*2(***ω****,***x**)

**y**ˆ = *Softmax*(**f**) =

*ef*1(***ω****,***x**) + *ef*2(***ω****,***x**) *, ef*1(***ω****,***x**) + *ef*2(***ω****,***x**)

*.*

En efecto, tenemos una salida para cada clase **y**ˆ = (*y*ˆ1*, y*ˆ2). Sin embargo, por construcci´on sabemos que *y*ˆ2 = 1 *y*ˆ1, por lo que podemos prescindir de la segunda salida y de los par´ametros necesarios en **f** para generarla (se dice que **f** est´a sobre-parametrizada). El vector de probabilidades entonces es:

*−*

**y**ˆ = (*y*ˆ1*,* 1 *− y*ˆ1)*.* Tomamos como es habitual *−* log(*·*) obteniendo

*N*

*—* log(*p*(*yc |* ***ω****,* **x**)) = *− y*1 log(*y*ˆ1) + *y*2 log(*y*2)

*i N*

= *− y*1 log(*y*ˆ1) + (1 *− y*1) log(1 *− y*ˆ1) (11)

*i*

##### Regularizacio´n

En esta secci´on, centraremos la atenci´on en el t´ermino **a priori**. Con es- te t´ermino imponemos un conocimiento a priori sobre los pesos de la red (los par´ametros de la funci´on) de manera que la red resultante est´e sesga- da por dicho conocimiento. Esto significa que si la informaci´on a priori que imponemos es c¸ierta”, los resultados esperados deber´an ser buenos; en caso contrario, no. Es importante pues, entender qu´e significa considerar que los pesos sean gausianos de media 0. Si pintamos la funci´on de distribuci´on que hemos impuesto para cada uno de los par´ametros que consideraremos i.i.d., tenemos:

(

2 *.*

1

*p*(*ω*) = *√*2*πσ*

2

exp

2

*−|ω|*2

2*σ*

*p*(*ω*)

*σ*2

*µ*2 = 0

Figura 2: Funci´on de distribuci´on Gausiana de un par´ametro *ω*

Es decir, en el proceso de optimizaci´on, los valores de finales de los par´ametros habr´an sido sesgados como si hubieran sido muestreados de la distribuci´on anterior, *ω p*(*ω*). De hecho, siguiendo ([8](#_bookmark247)), el valor que minimiza el t´ermino asociado al a priori es ***ω*** = 0, es decir, establecer todos los pesos a 0. Sin embargo, esto implicar´ıa un comportamiento desastroso en el objetivo de verosimilitud, por tanto, el proceso de minimizaci´on es un compromiso entre cumplir ambos t´erminos.

*∼*

En general, se suelen considerar redes con m´as par´ametros de los necesarios, es por ello que este tipo de a priori suele mejorar los resultados pues promueve que los pesos de una red sean cercanos a 0. No obstante, la elecci´on de este

a priori es completamente arbitraria y podr´ıamos pensar en qu´e propiedades o conocimiento tenemos sobre los valores ´optimos de una red.

## XVI Clase

#### Redes Neuronales (NN)

En esta secci´on nos vamos a centrar en qu´e son y qu´e propiedades tienen las redes neuronales. Inspiradas en la funci´on que desempen˜a una neurona cerebral, las neuronas o red de neuronas en computaci´on representan una **factorizaci´on** de funciones de un tipo concreto. Cada neurona tiene mu´lti- ples entradas a partir de las cuales toma la ”decisi´on”de *activarse* o no. Esto suele ser una combinaci´on lineal de las entradas. Por ejemplo, sean **x** R*m* y **w** R*m, b* R el vector de entrada y los pesos de la neurona 1, su salida lineal viene dada por:

*∈ ∈*

*∈*

*f*1 = **w***T* **x** + *b* = *w*1

*w*2 *· · · wm*

*x*1 

. + *b.*

]





*x*2



*xm*

Adema´s, se suele an˜adir una funci´on de activaci´on *σ*(*·*) como puede ser una tangente hiperb´olica tanh (*·*) : R *→* [*−*1*,* 1]; es decir, la salida lineal anterior, que puede tomar cualquier valor en R, se transforma al rango [*−*1*,* 1]

*f*1 = *σ* (**w***T* **x** + *b*



*x*1

*w*

1

*b*

*x*2

*w*2

*f*

.

*wm*

*xm*

Figura 3: Neurona

En la figura [3](#_bookmark253), existen *m* entradas y 1 salida. Si generalizamos a *m* entradas y *n* salidas, tenemos: **x** *∈* R*m*, *W ∈* R*n×m*, **b** *∈* R*n* y **f** *∈* R*n*. Es decir:



*b*2

*w*1*,*1 *w*1*,*2 *· · · w*1*,m*   *x*1

 *b*1 

**f** = *σ*



*w*2*,*1 *w*2*,*2 *· · · w*1*,m*   *x*2

 +  

 .

*wn,*1 *wn,*2 *· · · wn,m*

.

. .

.   . 

*xm*

 .



cuya reprensentaci´on gr´afica podemos ver en la figura [4](#_bookmark254).

*bn*



*b*1

*x*1

*w*1*,*1

*f*1

*w*2*,*1

*b*2

*w*1*,*2

*x*2

*w*2*,*2

*wn,*1

*f*2

*w*2*,m*

*wn,*2

.

.

*w*1*,m*

*bn*

*fn*

*wn,m*

*xm*

Figura 4: Red de una capa oculta: *m* entradas y *n* salidas

La noci´on de capa surge cuando las salidas de las *n* neuronas constituyen las entradas a otras neuronas. As´ı se construyen redes multi-capa de forma composicional. Por ejemplo, si consideramos la red de la figura [5](#_bookmark255), que tiene

5 entradas, 6 salidas y 8 neuronas *ocultas*, de forma matricial, su salida es:

(1)

(2)

*w*



1*,*1

*· · · w*1*,m*  

(1)

*w*

1*,*1

*· · · w*1*,m*

  *x*1

 *b*(1) 

 *b*(2) 

**f** = *σ*  . . .

(2)

.  *σ*  . . .

.   .

 +  .

 +  . 

. . .

1

1

(2)

(1)

. . .

  .

 . 

 . 

es decir,

(2)

*n,*1

*w*

*· · · wn,m*

(1)

*n,*1

*w*

*· · · wn,m xn*

(1)

*n*

*b*

(2)

*n*

*b*

**f** = *σ*(*W* (2)(*σ*(*W* (1)**x** + **b**(1))) + **b**(2)) = **f**2(**f**1(**x**)) = **f**2 *◦* **f**1(**x**)*.*

La caracter´ıstica principal de estas redes es su factorizaci´on: se suelen usar las m´ısmas funciones en cada capa, as´ı **f***n* = *σ*(*W* (*n*)**f***n−*1 + **b**(*n*)), nos proporciona una regla iterativa para crear redes cada vez m´as profundas (Deep Neural Network - DNN). El adjetivo *profundo - deep* suele atribu´ırsele a redes con 3 o m´as capas, criterio que es, por supuesto, arbitrario.

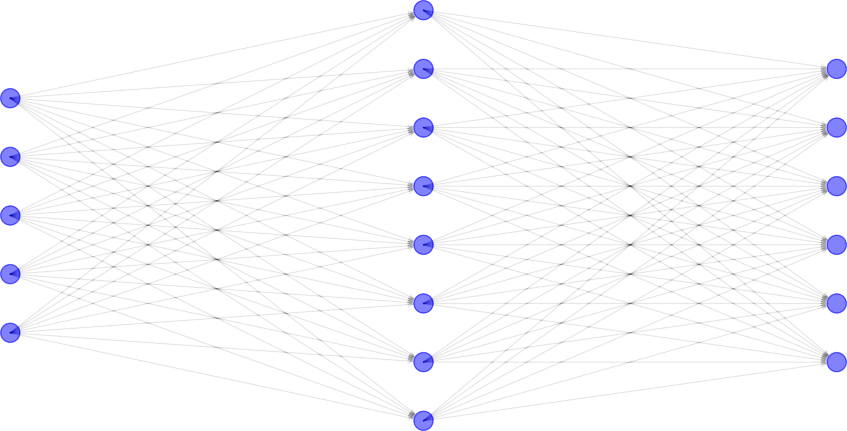


Figura 5: Topolog´ıa de una Red Neuronal con 1 capa oculta

Estas redes son tambi´en conocidas como Feedforward Networks por el hecho de que la informaci´on *fluye* de forma dirigida hacia delante. La informaci´on que recibe una capa proviene u´nicamente de la capa precedente. A partir de estas redes, aparecen otras en las que existe realimentaci´on y por lo tanto la informaci´on puede *viajar* hasta capas precedentes (Recurrent Neural Network

- RNN). Estos bucles de retro-alimentaci´on dan la posibilidad de introducir

bloques de memoria en las redes y por lo tanto describir *estados*; un ejemplo de ello son las LSTM (Long Short Term Memory).

No olvidemos que el principal objetivo de usar una red neuronal es la de apro- ximar una funci´on, es decir, las NN son funciones de aproximaci´on universal. Prueba de ello es el siguiente teorema [[1](#_bookmark291)]:

**Theorem 1.** *”Sea σ una funci´on no constante, acotada y mon´otona crecien- te. Sea In el cubo unitario n-dimensional,* [0*,* 1]*n. El espacio de las funciones cont´ınuas en In es C*(*In*)*. Entonces, dado un ϵ >* 0 *y cualquier funci´on f C*(*In*)*, existe un entero positivo N tal que, podemos definir el sumatorio finito:*

*∈*

*N*

*F* (*x*) = *αiσ*(***w****T* ***x*** + ***b***)

*i*=1

*como aproximaci´on de la funci´on f ; i.e.*

*|F* (*x*) *− f* (*x*)*| < ε*

*para todo x ∈ In. Es decir, F* (*x*) *es densa en C*(*In*)*.”*

Sabemos, como demuestra George Cybenko en [[1](#_bookmark291)] en 1989 que estas funciones aproximan cualquier funci´on cont´ınua en un conjunto compacto. Es intere- sante ver que una de las condiciones sobre la funci´on de activaci´on *σ* es, entre otras, la de ser acotada, cosa que no ocurre con una funci´on de activaci´on muy usada: *ReLU* (*x*) = m´ax(0*, x*) (figura [6](#_bookmark256)).

3

2

1

0

*−*2 0 2

Figura 6: Funci´on de Activaci´on: *ReLU* (*x*)

## XVII Clase

#### Introduccio´n a Pytorch

Esta clase se desarrolla en el material de Pr´acticas.

## XVIII Clase

#### Redes Convolucionales(CNN)

Las redes convolucionales constituyen un elemento esencial en el Aprendizaje Automa´tico aplicado a la Visio´n Artificial y al Procesamiento de Imagen. Una de las primeras redes convolucionales de ´exito fue propuesta por LeCun et al. [[2](#_bookmark292)], aunque la primera red convolucional como tal se conoce como *Neocognitron* [[3](#_bookmark293)]. Estas redes, as´ı como el conocido Deep Learning hoy en d´ıa, viene fuertemente inspirado por unos trabajos en 1962 sobre el *campo receptivo* del sistema visual de los gatos: *Receptive fields of single neurones in the cat’s striate cortex”* [[4](#_bookmark294)]. Estos trabajos mostraban c´omo el sistema visual de los gatos era jer´arquico y se compon´ıa de grupos de neuronas que detectaban (activaban o no) caracter´ısticas b´asicas.

Es aqu´ı donde aparece la convoluci´on como operaci´on lineal que permite de- tectar patrones, especialmente u´tiles en im´agenes. En general, la convoluci´on se define como un operador lineal entre dos funciones *f* y *g*:

+*∞*

*f* (*x*) *∗ g*(*x*) =

*−∞*

*f* (*η*)*g*(*x − η*)*dη.*

*−*

De manera intuitiva podemos ver esta operaci´on como una medida de lo parecido que son ambas funciones, concretamente *f* (*x*) y *g*( *x*) (n´otese la inversi´on del eje *x* que se aplica para obtener la propiedad de comnutaci´on). De hecho, se puede definir la correlaci´on entre dos funciones como el producto de convoluci´on:

*f* (*x*) *∗ g*(*−x*) =

+*∞*

*f* (*η*)*g*(*η − x*)*dη.*

*−∞*

El equivalente discreto unidimensional es

+*∞*

*f* [*n*] *∗ g*[*n*] =

*h* =*−∞*

*f* [*h*]*g*[*n − h*]*,*

y el bidimensional (que aplicaremos a im´agenes):

+*∞*

*f* [*i, j*] *∗ g*[*i, j*] =

+*∞*

*f* [*k, l*]*g*[*i − k, j − l*]*.*

*k*=*−∞ l*=*−∞*

A pesar de c´omputo pueda parecer tedioso, y, por tanto, la interpretaci´on de dichas convoluciones, es bastante f´acil de entender con un ejemplo en el que las funciones son finitas (de soporte compacto). De hecho, en el caso 2D, podemos identificar *f* [*i, j*] como una imagen y *g*[*i, j*] como un filtro o patr´on que queremos encontrar, a modo de ventana deslizante, recorriendo pixel a pixel (elemento a elemento de la matriz) como podemos ver en la figura [7](#_bookmark261).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 5 | 2 | 2 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 4 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 2 | 2 | 5 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figura 7: Matrix de entrada, nu´cleo de convoluci´on y resultado.

Las redes convolucionales consisten pues en sustituir la funci´on de aproxima-

ci´on de cada neurona por la siguiente:

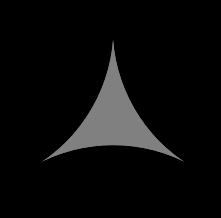
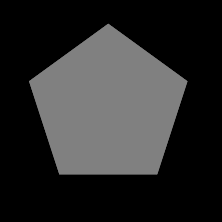
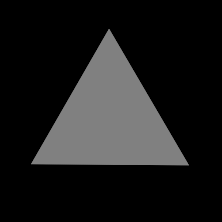
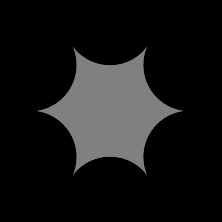
*f* = *σ* (*K ∗ X* + *B*)

donde ahora *X* R*m×n* es una imagen, *K* R*k×k* es un kernel de taman˜o *k k* y *B* R*m′×n′* es una matriz de sesgos. En la figura [8](#_bookmark262) se muestra una entrada (imagen) que se convoluciona con 4 *kernels* resultando en 4 *caracter´ısticas*. La salida de dicha capa de convoluci´on se obtiene sumando 4 matrices de sesgos (biases).

*× ∈*

*∈ ∈*

Kernels de Convoluci´on Caracter´ısticas *Biases*



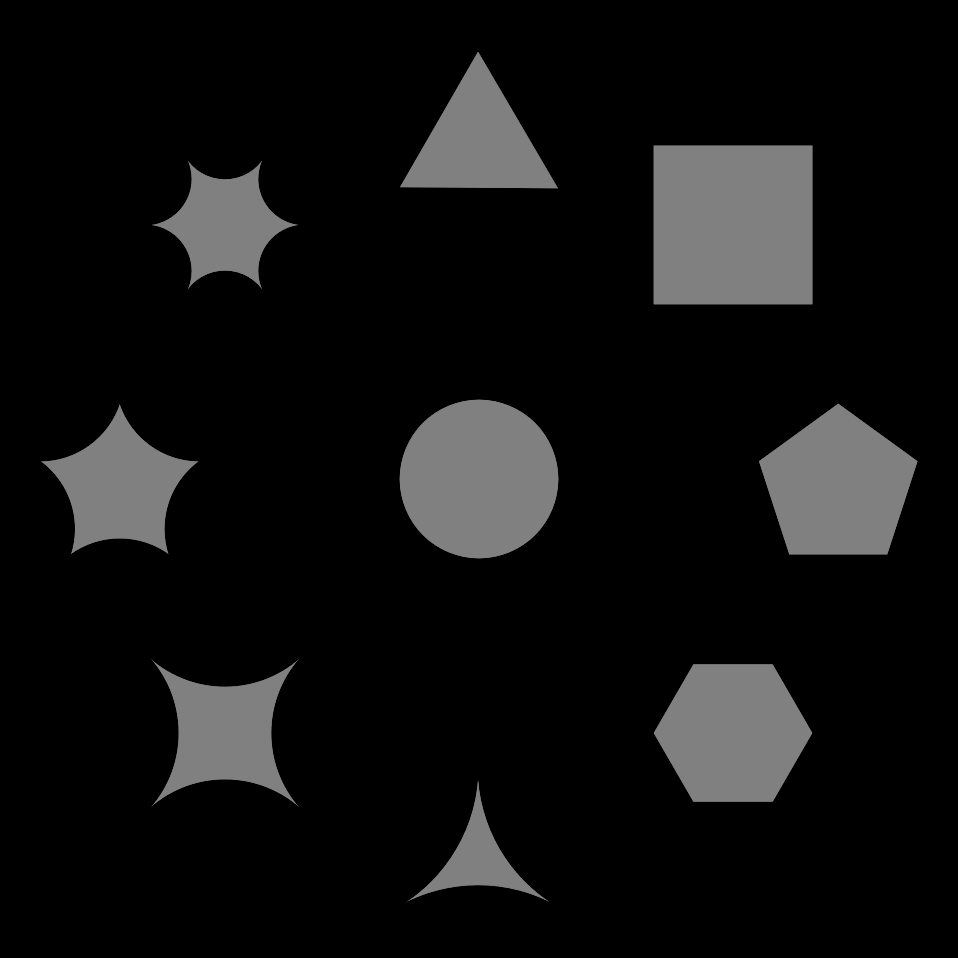
\* +

Figura 8: Una capa convolucional con 4 nodos (kernels).

#### Convoluci´on como Operaci´on Matricial

El producto de convoluci´on se puede expresar como multiplicaci´on de una matriz por un vector tal que:

(*K ∗ X*)(:) = *W* **x** (12)

donde **x** es la vectorizaci´on de la matriz *X* (imagen), *W* es la matriz co- rrespondiente al kernel *K* y (:) denota la vectorizaci´on de una matrix por filas. Veamos un ejemplo de c´omo construir *W* a partir de un kernel *K*. Sean *K ∈* R3*×*3, *X ∈* R4*×*4

 *k*

*K* =

*k*2*,*1 *k*2*,*2 *k*2*,*3

3*,*1

*k*

1*,*1

*k*1*,*2

*k*

3*,*2

*k*1*,*3 

*k*

3*,*3

 *x*1*,*1 *x*1*,*2 *x*1*,*3 *x*1*,*4 

 

*x*2*,*1 *x x x*21*,*2 2*,*3 2*,*4



*, X* =

*.*

*x*3*,*1 *x*3*,*2 *x*3*,*3 *x*3*,*4

*x*4*,*1 *x*4*,*2 *x*4*,*3 *x*4*,*4



La dimensi´on de la salida de la convoluci´on es: (3 + 4 1*,* 3 + 4 1) = (6*,* 6). La vectorizaci´on de *X* en **x** se hace an˜adiendo ceros para que el vector de entrada tenga el mismo nu´mero de filas que la salida: 6 *×* 6 = 36

*— −*

 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | *x*1*,*1 | *x*1*,*2 | *x*1*,*3 | *x*1*,*4 |
| 0 | 0 | *x*2*,*1 | *x*21*,*2 | *x*2*,*3 | *x*2*,*4 |
| 0 | 0 | *x*3*,*1 | *x*3*,*2 | *x*3*,*3 | *x*3*,*4 |
| 0 | 0 | *x*4*,*1 | *x*4*,*2 | *x*4*,*3 | *x*4*,*4 |

 

*X* =





entonces **x** = *X*(:)

**x** = [0*, · · · ,* 0*, x*1*,*1*, x*1*,*2*, x*1*,*3*, x*1*,*4*,* 0*, · · · ,* 0*, x*4*,*1*, x*4*,*2*, x*4*,*3*, x*4*,*4]*T ∈* R36

De la misma forma vectorizamos el kernel *K* con ceros an˜adidos

 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *k*1*,*1 | *k*1*,*2 | *k*1*,*3 | 0 | 0 | 0 |
| *k*2*,*1 | *k*2*,*2 | *k*2*,*3 | 0 | 0 | 0 |
| *k*3*,*1 | *k*3*,*2 | *k*3*,*3 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

 

*K* =

como **k**:

**k** = [*k*1*,*1*, k*1*,*2*, k*1*,*3*,* 0*,* 0*,* 0*, k*2*,*1*, k*2*,*2*, k*2*,*3*,* 0*,* 0*,* 0*, k*3*,*1*, k*3*,*2*, k*3*,*3*,* 0*, · · · ,* 0]*T ∈* R36

Por u´ltimo, construimos la matriz *W*

 *k*1 *k*2 *k*3 *k*4 *· · · k*36 

*W* = 













0 *k*1 *k*2 *k*3 *· · · k*35

0 0 *k*1 *k*2 *· · · k*34

0 0 0 *k*1 *· · · k*33

. . . .

.

.

.

.



0 0 0 0 *· · · k*1

con la que podemos obtener la equivalencia de la ecuaci´on ([12](#_bookmark264)).

En resumen, la convoluci´on es una operaci´on lineal equivalente a una mul- tiplicaci´on matricial y en consecuencia, cada nodo de una red convolucional puede sustituirse por una red neuronal. La raz´on de usar las convoluciones en vez de redes neuronales para la aplicaci´on de filtros es la carga computacio- nal y requerimiento en memoria. Mientras un kernel de convoluci´on tiene *k × k* coeficientes y se aplica independientemente del taman˜o de la entrada

*X* R*m,n*, la matriz *W* que debemos construir tiene dimensiones dependien- tes de la imagen: (*m n m n*) (*k k*).

*× × × ≫ ×*

*∈*

Un nodo de convoluci´on es equivalente a una red neuronal en la que la mayor´ıa de los coeficientes valen 0. Este hecho constituye un *apriori* que se impone directamente y viene motivado por las propiedades de localidad que existen en las im´agenes, principalmente: un p´ıxel de una imagen est´a *relacionado* con otros si estos se encuentran en una regi´on cercana (que se har´a coincidir con el taman˜o del filtro).

## XIX Clase

#### Introduccio´n a Pytorch II

Esta clase se desarrolla en el material de Pr´acticas.

## XX Clase

#### Descenso de Gradiente - Backpropagation

Hemos visto hasta ahora que una red neuronal tiene por objetivo aproximar *cualquier* funci´on (bajo ciertas condiciones). Tenemos una factorizaci´on de funciones que nos garantiza dicha aproximaci´on con una precisi´on arbitraria y un nu´mero finito de nodos. Sin embargo, la demostraci´on de aproximaci´on universal nada nos dice sobre c´omo encontrar los coeficientes que aproximan la funci´on que buscamos. Sin p´erdida de generalidad tomaremos como ejem- plo el problema de regresi´on ([8](#_bookmark247)) en el que consideraremos maximizar solo el t´ermino de verosimilitud (Maximum Likelihood). El problema de minimiza- ci´on que tenemos es

***ω****∗* = arg m´ın *C*(***ω***) = arg m´ın 1 *||***y**(*i*) *−* **f**(***ω****,* **x**(*i*))*||*2*.* (13)

*N*

***ω***

***ω***

*N*

2

*i*

Para ello es necesario encontrar las condiciones de optimalidad de primer orden (Euler-Lagrange):

*∂C*(***ω***) = 0

*∂****ω***

Si consideramos ***ω*** como un vector en el que se almacenan todos los pesos de nuestra red *f* (***ω****,* **x**), entonces lo que necesitamos calcular es el gradiente de *C*(***ω***) con respecto de cada uno de los par´ametros:

es decir,

*∇*

*∂C*(***ω***) =

*∂****ω***

***ω****C*(***ω***)

*∇****ω****C*(***ω***)*k*

= *∂C*(***ω***) *.*

*∂ω*

*k*

Veamos un ejemplo con una red sencilla:

*z*(1) = *w*(1)*x*1 + *w*(1)*x*2 + *b*(1)

1 1*,*1 1*,*2 1

*a*(1) = *σ*(*z*(1))

1 1

(1)

*z*(2) = *w*(2)*a*(1) + *w*(2)*a*(1) + *w*(2)*a*(1) + *b*(2)

*a*1 1

1*,*1 1

1*,*2 2

1*,*3 3 1

*a*(2) = *σ*(*z*(2))

*x z*(1) = *w*(1)*x*1 + *w*(1)*x*2 + *b*(1)

1 1

(2)

1 2 2*,*1

2*,*2 2 *a*1

(1)

(2)

*a*(1) = *σ*(*z*(1))

*L*(***ω***) = (*y*1 *− a*1 )2 + (*y*2 *− a*2 )2

2 2

(1)

*a*2

*z*(2) = *w*(2)*a*(1) + *w*(2)*a*(1) + *w*(2)*a*(1) + *b*(2)

*L*(***ω***)

*a*(2) = *σ*(*z*(2))

2

2*,*1

1

2*,*2

2

2*,*3

3

2

*x z*(1) = *w*(1)*x*1 + *w*(1)*x*2 + *b*(1)

2 2

(2)

2 3 3*,*1

3*,*2 3 *a*2

*a*(1) = *σ*(*z*(1))

3 3

(1)

*a*

3

Esta es la representaci´on habitual de una red, donde cada salida *a*(*L*) de la capa *L* representa la activaci´on no lineal de la entradas por sus pesos. Para explicar el algoritmo de back-propagation, se suelen obviar estas activaciones para que el c´alculo se simplifique. No obstante, el procedimiento queda igual

*i*

si separamos en 2 nodos las salidas lineales *z*(*L*) de las activaciones *a*(*L*) =

*i*

*i*

*σ*(*z*(*L*)). Es decir

*i*

(2)

*w*(1)

1*,*1

*w*(1)

1*,*2

*b*(1)

1

*w*(2)

1*,*1

*w*(2)

1*,*2

*w*(2)

1*,*3

*b*(2)

1

*w*(1)

2*,*1

*w*(1)

2*,*2

*b*(1)

2

*w*(2)

2*,*1

*w*(2)

2*,*2

*w*(2)

2*,*3

*b*(2)

2

*w*(1)

3*,*1

*w*(1)

3*,*2

*b*(1)

3

*z*(1)

1

*a*(1)

1

*x*1

*z*(2)

1

*z*(1)

2

*a*(1)

2

*x*2

*z*(2)

2

*z*(1)

3

*a*

(1)

3

*a*

1

*L*(***ω***)

(2)

*a*

2

El vector de pesos es

***ω*** = (*ω*1*, ω*2*, ω*3*, ω*4*, ω*5*, ω*6*, ω*7*, ω*8*, ω*9*, ω*10*, ω*11*, ω*12*, ω*13*, ω*14*, ω*15*, ω*16*, ω*17)*T*

= (*w*(1)*, w*(1)*, w*(1)*, w*(1)*, w*(1)*, w*(1)*, b*(1)*, b*(1)*, b*(1)*, w*(2)*, w*(2)*, w*(2)*, w*(2)*, w*(2)*, w*(2)*, b*(2)*, b*(2))*T ,*

1*,*1

1*,*2

2*,*1

2*,*2

3*,*1

3*,*2 1 2 3

1*,*1

1*,*2

1*,*3

2*,*1

2*,*2

2*,*3 1 2

entonces

*∇****ω****C*(***ω***) =

*∂C*(***ω***) *,*

*∂ω*1

(

*∂C*(***ω***)

*∂C*(***ω***)

*∂ω*2

*∂C*(***ω***)

*∂w*

*, · · · ,*

*∂C*(***ω***) *T*

*∂ω*17

*∂C*(***ω***) *T*

2

= (1) *,*

*∂w*1*,*1

(1) *, · · · ,*

1*,*2

*∂b*(2)

donde

*∂C*(***ω***) = *∂* 1

*N*

*||***y**(*i*) *−* **f**(***ω****,* **x**(*i*))*||*2

*∂ωk*

*∂ω N*

*i*

*k*

*∂* 1

=

*∂ω N*

*k*

*i j*

2

2

*N*

*n*

*y*(*i*) *− fj*(***ω****,* **x**(*i*)) *.*

*j*

Por simplicidad y sin p´erdida de generalidad asumiremos que tan solo existe 1 ejemplo de entrenamiento (*N* = 1). Puesto que tenemos 2 salidas, la funci´on de coste ser´a

*L*(***ω***) = (*y*1 *− f*1(***ω****,* **x**)) + (*y*2 *− f*2(***ω****,* **x**))

2 2

o equivalentemente,

*L*(***ω***) = *y*1

*— a*(2) 2 + *y*

*— a*(2) 2 *.*

Ahora debemos calcular el gradiente con respecto a cada par´ametro. Por ejemplo, si consideramos el par´ametro *ω*10 = *w*(2), aplicando la regla de la

1

2

1*,*1

2

cadena tenemos:

(2)

*w*(1)

1*,*1

*w*(1)

1*,*2

*b*(1)

1

*w*(2)

1*,*1

*w*(2)

1*,*2

*w*(2)

1*,*3

*b*(2)

1

*w*(1)

2*,*1

*w*(1)

2*,*2

*b*(1)

2

*w*(2)

2*,*1

*w*(2)

2*,*2

*w*(2)

2*,*3

*b*(2)

2

*w*(1)

3*,*1

*w*(1)

3*,*2

*b*(1)

3

*z*(1)

1

*a*(1)

1

*x*1

*z*(2)

1

*z*(1)

2

*a*(1)

2

*x*2

*z*(2)

2

*z*(1)

3

*a*

(1)

3

*a*

1

*L*(***ω***)

(2)

*a*

2

*∂L*(***ω***)

*∂* (***ω***) *∂a*(2)

=

1

*L*

*∂* (***ω***) *∂a* (2) 0

+

2

*L*

*∂w*(2)

*∂a*(2) *∂w*(2)

*∂ a*(2) *∂w*(2)

1*,*1

1 1*,*1

2 1*,*1

*∂ L*(***ω***) *∂a*(2) *∂z*(2)

= 1 1

*∂a*(2) *∂z*(2) *∂w*(2)

1 1 1*,*1

= *−*2 *y*1 *− a*(2) *σ′* (*z*(2))*a*(1)

1

1

1

donde

*∂L*(***ω***) = *∂ y*

*— a*(2) 2 = *−*2 *y*

*— a*(2)

*∂a*(2)

1

1

*∂a*(2)

*∂a*(2) 1 1 1 1

*∂ ′*

1 =

1

*σ*(*z*(2)) = *σ* (*z*(2))

*∂z*(2)

1

*∂z*(2)

1

*∂w*

(2)

1*,*1

*∂z*(2) 1 1

*∂*

=

*w*(2)*a*(1) + *w*(2)*a*(1) + *w*(2)*a*(1) + *b*(2)

= *a*(1)

*∂w*

(2)

1*,*1

1*,*1

1

1*,*2

2

1*,*3

3

1

1

De la misma forma, derivamos ahora con respecto a *ω*1 = *w*(1):

1*,*1

(2)

*w*(1)

1*,*1

*w*(1)

1*,*2

*b*(1)

1

*w*(2)

1*,*1

*w*(2)

1*,*2

*w*(2)

1*,*3

*b*(2)

1

*w*(1)

2*,*1

*w*(1)

2*,*2

*b*(1)

2

*w*(2)

2*,*1

*w*(2)

2*,*2

*w*(2)

2*,*3

*b*(2)

2

*w*(1)

3*,*1

*w*(1)

3*,*2

*b*(1)

3

*z*(1)

1

*a*(1)

1

*x*1

*z*(2)

1

*z*(1)

2

*a*(1)

2

*x*2

*z*(2)

2

*z*(1)

3

*a*

(1)

3

*a*

1

*L*(***ω***)

(2)

*a*

2

*∂L*(***ω***)

= 1 + 2

*∂L*(***ω***) *∂a*(2)

*∂L*(***ω***) *∂a*(2)

(1)

*∂w*

1*,*1

*∂a*(2)

(1)

1*,*1

*∂a*(2)

(1)

1*,*1

*∂L*(***ω***) *∂a*(2) *∂z*(2) *∂L*(***ω***) *∂a*(2) *∂z*(2)

1

*∂w*

2

*∂w*

= 1 1 + 2 2

*∂a*(2) *∂z*(2) *∂w*(1) *∂a*(2) *∂z*(2) *∂w*(1)

1 1 1*,*1 2 2 1*,*1

*∂L*(***ω***) *∂a*(2) *∂z*(2) *∂a*(1) *∂L*(***ω***) *∂a*(2) *∂z*(2) *∂a*(1)

= 1 1 1 + 2 2 1

*∂a*(2) *∂z*(2) *∂a*(1) *∂w*(1) *∂a*(2) *∂z*(2) *∂a*(1) *∂w*(1)

1 1 1

1*,*1

2 2 1

1*,*1

*∂L*(***ω***) *∂a*(2) *∂z*(2) *∂a*(1) *∂z*(1) *∂L*(***ω***) *∂a*(2) *∂z*(2) *∂a*(1) *∂z*(1)

= 1 1 1 1 + 2 2 1 1

*∂a*(2) *∂z*(2) *∂a*(1) *∂z*(1) *∂w*(1) *∂a*(2) *∂z*(2) *∂a*(1) *∂z*(1) *∂w*(1)

1 1 1 1

1*,*1

2 2 1 1

1*,*1

*∂L*(***ω***) *∂a*(2) *∂z*(2) *∂L*(***ω***) *∂a*(2) *∂z*(2) *∂a*(1) *∂z*(1)

=

1 1 + 2 2

1 1

*∂a*(2)

*∂z*(2) *∂a*(1)

*∂a*(2)

*∂z*(2) *∂a*(1)

*∂z*(1) *∂w*(1)

1 1 1

2 2 1

1 1*,*1

= *−*2 f *y*1 *− a*(2) *σ′* (*z*(2))*w*(1) + *y*2 *− a*(2) *σ′* (*z*(2))*w*(1) *σ′* (*z*(1))*x*1

1

1

1*,*1

2

2

2*,*1

1

donde

*∂L*(***ω***) = *∂ y*

*— a*(2) 2 = *−*2 *y*

*— a*(2)

*∂a*(2)

1

1

*∂a*(2)

*∂a*(2) 1 1 1 1

*∂ ′*

1 =

1

*σ*(*z*(2)) = *σ* (*z*(2))

*∂z*(2)

1

*∂z*(2)

1

*∂z*(2) 1 1

*∂*

=

*w*(2)*a*(1) + *w*(2)*a*(1) + *w*(2)*a*(1) + *b*(2)

= *w*(2)

*∂a*(1)

1*,*1

1

1*,*2

2

1*,*3

3

1

1*,*1

*∂L*(***ω***) = *∂ y*

1

1

*∂a*(1)

*— a*(2) 2 = *−*2 *y*

*— a*(2)

*∂a*(2)

2

2

*∂a*(2)

*∂a*(2) 2 2 2 2

*∂ ′*

2 =

2

*σ*(*z*(2)) = *σ* (*z*(2))

*∂z*(2)

2

*∂z*(2)

2

*∂z*(2) 2 2

*∂*

=

= *w*(2)

*∂a*(1)

1

*∂a*(1)

*w*(2)*a*(1) + *w*(2)*a*(1) + *w*(2)*a*(1) + *b*(2)

*∂a*(1)

*∂*

1

2*,*1 1 2*,*2 2

*′*

2*,*3 3 2

2*,*1

1 =

1

*σ*(*z*(1)) = *σ* (*z*(1))

*∂z*(1)

1

*∂z*(1)

*∂z*(1) 1 1

*∂*

1 = *w*(1)*x*1 + *w*(1)*x*2 + *b*(1) = *x*1

(1)

*∂w*

*∂w*

1*,*1

(1)

1*,*1

1*,*1

1*,*2 1

#### Forma Matricial

A la vista de los resultados anteriores, podemos ver que para carcular cada derivada parcial con la que construir el gradiente asociado a cada peso, ne- cesitamos evaluar primero las salidas de los nodos asociados a una entrada **x** (forward pass) y luego completar con ´estos, los c´alculos restantes (backward pass). El c´omputo de todos los gradientes, que ser´an necesarios para apli- car el esquema de descenso, puede expresarse de forma matricial. Para ello podemos apoyarnos en el c´alculo de matrices Jacobianas. Una matriz Jaco- biana se compone de las derivadas parciales de primer orden de una funci´on **f** : R*m →* R*n*. Es decir, **f**(**x**) con **x** *∈* R*m*, tiene como matriz Jacobiana:





*∂f*1

*∂x*1



*∂f*2

*∂f*1

*∂x*2

*∂f*2

*∂f*1

*· · · ∂xm*

*∂f*2 

*J* (**x**) = 

**f**

*∂x*1

*∂f*3

*∂x*2 *· · ·*

*∂f*3

*∂xm*

*∂f*3 

*∂x*1

 .

.



*∂x*2

.

.

*· · ·*

. . .

*∂xm*

. 

.



*∂fn*

*∂x*1

*∂fn*

*∂x*2

*∂fn*

*· · · ∂xm*

Ahora calcularemos el Jacobiano entra cada capa de la red neuronal (consi- derando tambi´en como capa las etapas de activaci´on):

*JL*(**a**

(2)) = f

*∂J*

*∂a*(2)

1

*∂J*

*∂a*(2)

2

*, J***a**(2) (**z**(2)) = 







*∂a*(2)

*∂z*(2)

1

1

*∂a*(2)

2

1

*∂z*(2)

*∂a*(2)

*∂z*(2)

1

2





*,*



*∂a*(2)

2

2

*∂z*(2)

 *∂z*

(2)

1

(2)

1

*∂z*

(2)

1

*∂z*

*∂a*(1)

 *∂z*(1)

*∂a*(1)

*∂z*(1)

1

2

*∂a*(1)

*∂z*(1) 

(1)

 *∂a*(1)

*∂a*(1)

*∂a*(1) 

(1)

*∂a*(1)

*∂a*(1)

*∂a*(1) 

*J***z**(2) (**a** ) =



1

*∂z*(2)

2

*∂a*(1)

1

2

*∂z*(2)

2

*∂a*(1)

2

3

*∂z*(2)

2

*∂a*(1)

3

*, J* (1) (**z**

) = 2

*∂z*(1)

1

1



1







*∂ a*(1)

3

*∂z*(1)

1

2

*∂z*(1)

2

*∂ a*(1)

3

*∂z*(1)

2

2

(1)

1

3

3



*∂z* 



*∂a*(1)

3

*∂z*(1)

3

y tambi´en los Jacobianos con respecto a los par´ametros:

 **a**

 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *∂z*(2) |  | *∂z*(2) |  | *∂z*(2) |  | *∂z*(2) |  | *∂z*(2) |  | *∂z*(2) |  | *∂z*(2) |  | *∂z*(2)  1 |
| *∂w*(2)  *∂z*(2) |  | *∂w*(2)  *∂z*(2) |  | *∂w*(2)  *∂z*(2) |  | *∂b*(2)  *∂z*(2) |  | *∂w*(2)  *∂z*(2) |  | *∂w*(2)  *∂z*(2) |  | *∂w*(2)  *∂z*(2) |  | *∂b*(2)  2  *∂z*(2)  2 |

*J***z**(2) (***ω***

(2)) = 

1

1*,*1

2

(2)

*∂w*

1*,*1

1

1*,*2

2

(2)

*∂w*

1*,*2

1

1*,*3

2

(2)

*∂w*

1*,*3

1

1

2

*∂b*(2)

1

1

2*,*1

2

(2)

*∂w*

2*,*1

1

2*,*2

2

(2)

*∂w*

2*,*2

1

2*,*3

2

2

(2)

*∂w*

2*,*3

*∂b*(2)

 *,*

*J***z**(1) (***ω***

(1)

) = 





*∂z*(1)

(1)

1*,*1

1

*∂w*

*∂z*(1)

2

(1)

1*,*1

*∂w*

*∂z*(1)

3

(1)

*∂w*

1*,*1

*∂z*(1)

(1)

1*,*2

1

*∂w*

*∂z*(1)

2

(1)

1*,*2

*∂w*

*∂z*(1)

3

(1)

*∂w*

1*,*2

*∂z*(1)

*∂b*(1)

*∂z*(1)

1

1

2

*∂b*(1)

*∂ z*(1)

1

3

*∂b*(1)

1

*∂z*(1)

(1)

2*,*1

1

*∂w*

*∂z*(1)

2

(1)

2*,*1

*∂w*

*∂z*(1)

3

(1)

*∂w*

2*,*1

*∂z*(1)

(2)

2*,*2

1

*∂w*

*∂z*(1)

2

(2)

2*,*2

*∂w*

*∂z*(1)

3

(2)

*∂w*

2*,*2

*∂z*(1)

*∂b*(1)

*∂z*(1)

1

2

2

*∂b*(1)

*∂z*(1)

2

3

*∂b*(1)

2

*∂z*(1)

(1)

3*,*1

1

*∂w*

*∂z*(1)

2

(1)

3*,*1

*∂w*

*∂z*(1)

3

(1)

*∂w*

3*,*1

*∂z*(1)

(1)

3*,*2

1

*∂w*

*∂z*(1)

2

(1)

3*,*2

*∂w*

*∂z*(1)

3

(1)

*∂w*

3*,*2

*∂z*(1)

*∂b*(1)

*∂z*(1)

1

3

2

 *.*



*∂b* 

(1)

*∂z*(1)

3

3



*∂b*(1)

3

No´tese que, en las capas de activaci´on, no existen dependencias entre salidas, es decir, podemos apriori imponer ceros en esas derivadas:

 *∂a*(2) 

0

*J***a**(2)

(**z**(2)) = 

1

*∂z*(2)

1

(2) 

0 *∂a*2

*∂z*(2)

2

 *∂a*(1) 

(1)

0

1

*∂z*(1)

1



0 0

*∂a*(1) 

*J***a**(1) (**z** ) = 0





2

*∂z*(1)

2

(1) 

 *∂z*(2)



1

(2)

1*,*1

1

(2)

1*,*2

1

(2)

1*,*3

(2)

*∂w*

*∂z*(2)

*∂w*

*∂z*(2)

*∂w*

0 0 *∂a*3

*∂z*(1)

3

*∂z*(2) 

1

*∂b*(2)

0 0 0 0



*,*



1

**z**

*J* (2) (***ω***

) =



0 0 0 0

2

2

2

2

*∂z*(2)

(2)

*∂w*

2*,*1

*∂z*(2)

(2)

*∂w*

2*,*2

*∂z*(2)

(2)

*∂w*

2*,*3

*∂z*(2)

*∂b*(2)

2

 *∂z*(1)

1

*∂w*(1)

*∂z*(1)

1

*∂w*(1)

*∂z*(1) 

(1)



1*,*1

1*,*2

1

*∂z*(1)

*.*

1

*∂b*(1)

0 0 0 0 0 0

*∂z*(1)

*∂z*(1) 

*J***z**(1) (***ω*** ) =



0 0 0

2

*∂w*(1)



2

*∂w*(2)

2

*∂b*(1)

0 0 0

 2*,*1

2*,*2 2

(1)

0 0 0 0 0 0 *∂z*3

3*,*1

*∂z*3

*∂b*(1)

*∂w*

(1)

(1)

*∂w*

(1) 

3

Finalmente, podemos calcular el gradiente con respecto a todos los par´ame- tros de la siguiente manera:

*∂z*3

(1)

3*,*2

*JL*

(***ω***(1))*T* l

(*JL*

(**a**(2))*J*

**a**(2)

(**z**(2))*J*

**z**(2)

(***ω***(2)) *T* l

*∇****ω****L*(***ω***) =

*JL*(***ω***(2))*T*

= (*J*

(**a**(2))*J*

**a**(2)

*L*

(**z**(2))*J*

**z**(2)

(***ω***(2))*J*

**z**(2)

(**a**(1))*J*

**a**(1)

(**z**(1))*J*

**z**(1)

(***ω***(1)) *T*

Con este gradiente podemos finalmente aplicar el esquema de descenso de gradiente tal que:

***ω****t*+1 = ***ω****t − lr · ∇****ω****t L*(***ω****t*)*.*

## XXI Clase

#### Optimizadores

Existen distintos tipos de optimizadores para resolver problemas de optimi- zaci´on en Deep Learning. A menudo, decidir cu´al es el m´as apropiado para nuestro problema no es trivial o simplemente no existen fundamentos te´oricos para tomar una decisi´on. Los optimizadores o algoritmos de optimizaci´on, a menudo se clasifican segu´n 2 criterios. El primero, dependiente del nu´mero de ejemplos a considerar en cada iteraci´on. As´ı tenemos:

* + 1. Batch (Gradient Descent): se consideran todos los ejemplos del conjun- to de entrenamiento en cada iteraci´on.

*∂C*(***ω***) = *∂* 1

*∂****ω***

*∂****ω*** *N*

*N*

*||***y**(*i*) *−* **f**(***ω****,* **x**(*i*))*||*2

2

*i*

* + 1. Mini-Batch (Gradient Descent): se consideran subconjuntos del con- junto de entrenamiento en cada iteraci´on.

*∂C*(***ω***) = *∂* 1

*∂****ω***

*∂****ω*** *B*

*B*

*||***y**(*i*) *−* **f**(***ω****,* **x**(*i*))*||*2

2

*i*

donde *B* es un subconjunto de *N* .

* + 1. Stochastic (Gradient Descent) - SGD: se considera s´olo 1 ejemplo del conjunto de entrenamiento en cada iteraci´on.

(*i*) (*i*) 2

*∂C*(***ω***) = *∂*

*∂****ω***

*∂****ω***

*||***y** *−* **f**(***ω****,* **x** )*||*

En la figura anterior podemos tener un intuici´on del comportamiento de cada t´ecnica. Mientras que el Batch GD es m´as estable y tiende al m´ınimo local, el SGD tiene un caracter m´as .exploratorio”. Por otro lado, el Mini-batch GD es un compromiso entre ambos enfoques. El Batch GD tiene propiedades de convergencia cuando el funcional a optimizar es convexo pero es lento y requiere bastante memoria. Es SGD resulta ser mejor en la pr´actica cuando el objetivo a minimizar es no convexo pues es capaz de evitar m´ınimos locales. Adema´s require poca memoria y puede ser usado para un entrenamiento online. El Mini-batch GD, de nuevo es un compromiso entre ambos, siendo una de las t´ecnicas m´as usadas.

2

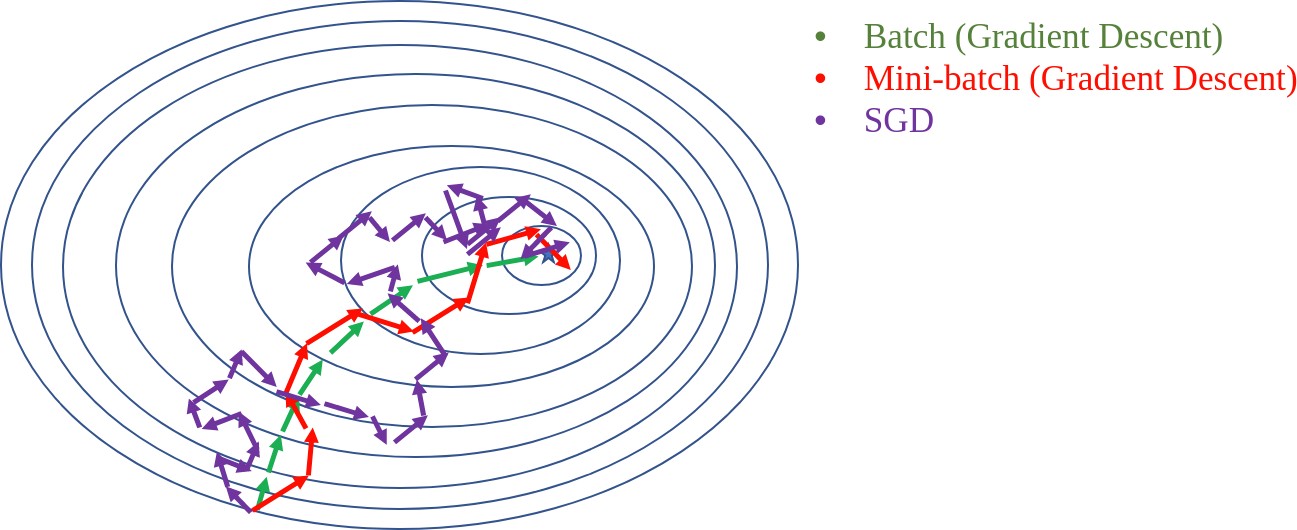


Figura 9: Comparaci´on: Batch, Mini-batch y Stochastic Gradient Descent

En segundo lugar, otra clasificaci´on u´til para entender los m´etodos de op- timizaci´on, se hace atendiendo a las posibles modificaciones del gradiente una vez calculado. Principalmente dichas modificaciones afectan a la tasa de aprendizaje o *learning rate*, adapt´andolo a los cambios en el gradiente. Destacamos los siguientes:

1. AdaGrad: este algoritmo adapta el *learning rate* de forma inversamente proporcional a la energ´ıa de los gradientes anteriores:

**Set**: *lr, δ* = 1*e* 7

*−*

**Initialize**: **r** = 0

**while** *not converged* **do**

*∂C*(***ω***)

**g** *←*

*∂****ω***

**r** *←* **r** + **g** *⊙* **g**

***ω*** *←* ***ω*** *− lr* 1*√ ⊙* **g**

*δ*+

**r**

##### end

**Algorithm 1:** AdaGrad

Tiene buenas propiedades de convergencia cuando la optimizaci´on es convexa pero en la pr´actica sufre de un desvanecimiento excesivo del *learning rate*.

1. RMSProp: este algorithmo modifica el anterior AdaGrad para mejorar el comportamiento en funcionales no convexos. Para ello, introduce un cambio en la acumulaci´on del gradiente con un filtro de media m´ovil:

**Set**: *lr, δ* = 1*e* 6*, ρ*

*−*

**Initialize**: **r** = 0

**while** *not converged* **do**

*∂C*(***ω***)

**g** *←*

*∂****ω***

**r** *← ρ***r** + (1 *− ρ*)**g** *⊙* **g**

***ω*** = ***ω*** *− lr√* 1 *⊙* **g**

*δ*+**r**

##### end

**Algorithm 2:** RMSProp

1. Adam: este algoritmo extiende a su vez el anterior incluyendo 2 momen- tos. Mientras que el RMSProp s´olo contempla el momento de segundo orden (sobre **g g**), con decaimiento exponencial, Adam an˜ade el mo- mento de primer orden sobre las propias variaciones del gradiente.

*⊙*

**Set**: *lr, δ* = 1*e* 8*, β*1 = 0*.*9*, β*2 = 0*.*999

*−*

**Initialize**: **r** = 0*,* **s** = 0

**while** *not converged* **do**

*∂C*(***ω***)

**g** *←*

*∂****ω***

*t ← t* + 1

**s** *← β*1**s** + (1 *− β*1)**g**

**r** *β*2**r** + (1 *β*2)**g g**

*← − ⊙*

***s*ˆ s**

*← t*

1*−***r** *β*1

***r*ˆ** = 1*−βt*

2

***ω*** = ***ω*** *− lr δ*+

*√*

***s*ˆ**

##### end

***r*ˆ** *⊙* **g**

**Algorithm 3:** Adam

Finalmente podemos encontrar otras clasificaciones para algoritmos de opti- mizaci´on que no ´solo utilizan las condiciones de optimalidad de primer orden, es decir, las primeras derivadas con respecto a la funci´on de coste, sino las segundas derivadas tambi´en. Estos m´etodos de segundo orden, como el m´eto- do de Newton, el Broyden–Fletcher–Goldfarb–Shanno (BFGS) o el L-BFGS requieren el c´omputo de las matrices Hessianas y que ´estas sean semidefi- nidas positivas. Si bien estos m´etodos son muy potentes, requieren mucha memoria y acaban siendo impracticables en Deep Learning cuando el orden de par´ametros a optimizar asciende a millones.

#### Inferencia Bayesiana

Desde un punto de vista Bayesiano, el problema de ML que consiste en en- contrar una funci´on **f** : R*m* R*n* **Y** = **f**(**X**) se puede abordar apoy´andonos en la regla de Bayes:

*→ |*

p(*M|D*) =

p(*M, D*)

(14)

p(*D*)

donde *M* representa el modelo y *D* los datos. Es decir, queremos definir la probabilidad condicional de obtener el modelo *M* dados los datos *D*. Los datos o evidencia son *D* = (**X***,* **Y**) y el modelo *M* = **F**. De esta manera podemos definir las distribuciones siguientes:

**x** *∼* p(**X**)

**y** *∼* p(**Y**)

**f** *∼* p(**F**)

donde **x***,* **y***,* **f** son muestras de cada una de sus respectivas distributiones. Es decir, **X** e **Y** son distribuciones de datos mientras que **F** es una distribuci´on de funciones.

Re-escribimos ([14](#_bookmark273)) como

p(**F***,* **X***,* **Y**)

p(**Y** *|* **F***,* **X**)p(**F***,* **X**)

p(**Y** *|* **F***,* **X**)---p(---**X**---*|* **---**

p(**F** *|* **X***,* **Y**) =

=

p(**X***,* **Y**)

=

p(**X***,* **Y**)

p(**Y** *|*

**X**) p(**X**)

**F**)p(**F**)

(15)

donde p(**X** *|* **F**) = p(**X**) porque los datos **X** no dependen de **F**, obteniendo:

p(**F X***,* **Y**) = p(**Y** *|* **F***,* **X**)p(**F**) *.* (16)

*|*

p(**Y** *|* **X**)

Una alternativa que nos facilita el c´omputo de la probabilidad a posteriori y la visualizaci´on del sistema, es considerar dicho sistema como un *Modelo Gr´afico Ac´ıclico Dirigido* (Directed Acyclic Graph Models - DAG en ingl´es). En estos modelos tambi´en conocidos como *Bayesian Networks* o *Belief Net- works*, se utilizan las propiedades de dependencia entre nodos para calcular r´apidamente la probabilidad conjunta del sistema.

**X**

**F**

**Y**

Figura 10: Modelo Gr´afico Ac´ıclico Dirigido

En nuestro caso, p(**F***,* **X***,* **Y**) se obtendr´ıa como el producto de todos los nodos condicionados a sus nodos *padre*:

Nodo **X**: no tiene nodos padre *→* p(**X**) Nodo **F**: no tiene nodos padre *→* p(**F**)

Nodo **Y**: tiene 2 nodos padre *→* p(**Y** *|* **F***,* **X**)

resultando

p(**F***,* **X***,* **Y**) = p(**Y** *|* **F***,* **X**)p(**X**)p(**F**)*.*

Adema´s, el denominador de ([15](#_bookmark274)) p(**X***,* **Y**) que corresponde a la probabilidad de los datos, usando la regla de Bayes a su vez, tendr´ıa 2 equivalencias:

p(**X***,* **Y**) = p(**Y** *|* **X**)p(**X**)

p(**X** *|* **Y**)p(**Y**)

donde es f´acil ver la correcta por sus dependencias en el grafo de la Figura ([11](#_bookmark278)).

Una forma de interpretar la equaci´on ([16](#_bookmark275)) es la siguiente: *buscamos una dis- tribuci´on de funciones* ***F*** *(modelos) que, con mucha propabilidad, haya gene- rado las salidas* ***Y****, con las entradas* ***X*** *(evidencia)*. Una vez obtenidas estas distribuciones, podemos usar la Inferencia Bayesiana para estimar salidas (prediciones)de datos que no han sido vistos *Dtest* = (**X***test,* **Y***test*):

**x***test ∼* p(**X***test*)

**y***test ∼* p(**Y***test*)

usando para ello la *probabilidad a posteriori* ([14](#_bookmark273)) en la siguiente *ecuaci´on de predicci´on*:

p(**y***test |* **x***test,* **X***,* **Y**) = p(**y***test |* **F**)p(**F** *|* **x***test,* **X***,* **Y**)d**F** (17) y a continuaci´on, muestreando de dicha distribuci´on:

**y***test ∼* p(**y***test |* **x***test,* **X***,* **Y**)

o, alternativamente, hallando su valor esperado Ep

**y***test*

[**y***test*]. N´otese que para

muestrear de esta distribuci´on, es a su vez necesario muestrear la distribuci´on

a posteriori del modelo ([16](#_bookmark275)):

**f** *∼* p(**F** *|* **X***,* **Y**)*.*

Con todos esos modelos **F** integraremos en ([17](#_bookmark276)) cada una de sus correspon- dientes predicciones para obtener **y***test*. Adem´as, vamos a caracterizar cada uno de estos modelos de forma param´etrica, es decir, vamos a imponer que todas las posibles funciones en **F** dependen de unos par´ametros **W** que tam- bi´en son una distribuci´on:

**f**(**W**) *∼* p(**F**)

**w** *∼* p(**W**) de manera que ([17](#_bookmark276)) se transforma en:

p(**y***test |* **x***test,* **X***,* **Y**) = p(**y***test |* **F**)p(**F** *|* **x***test,* **W**)p(**W** *|* **X***,* **Y**)d**F**d**W***.*

(18)

En este punto se suele hacer una simplificaci´on cuando queremos usar redes neuronales m´as adelante. Asumiremos que solo tenemos un modelo **f**(*W* ) cuyos par´ametros s´ı son muestras de una distribuci´on. De esta forma tenemos

un u´nico modelo a nivel de arquitectura, pero cada instanciaci´on de sus

par´ametros nos dar´a una funci´on distinta. Es habitual que estas instancias de un modelo, se les llame tambi´en modelos. Integrando en **F** = **f** tenemos

p(**y***test |* **x***test,* **X***,* **Y**) = p(**y***test |* **x***test,* **W**)p(**W** *|* **X***,* **Y**)d**W** (19) donde ahora el a posteriori que necesitamos para obtener

**y***test ∼* p(**y***test |* **x***test,* **X***,* **Y**)

es p(**W** *|* **X***,* **Y**) y el modelo gr´afico asociado:

**X**

**W**

**Y**

Figura 11: Modelo Gr´afico Ac´ıclico Dirigido (**f**-param´etrica)

Sin embargo, no siempre es posible evaluar estas distribuciones desde un punto de vista anal´ıtico ni computacional. Debemos pues buscar una soluci´on para evaluarlas f´acilmente. Una de estas t´ecnicas se conoce como Inferencia Variacional que veremos en la secci´on [7.4](#_bookmark280).

#### Par´ametros Deterministas vs Estoc´asticos

Es importante entender la diferencia que supone tener una funci´on cuyos par´ametros son deterministas y otra en la que son distribuciones. Para ilus- trar un poco m´as el problema empezaremos por diferenciar:

* + 1. Una funci´on param´etrica con par´ametros deterministas
    2. Una funci´on param´etrica con par´ametros estoc´asticos

##### f con par´ametros deterministas

En este caso, el problema es simple. Se trata de encontrar los par´ametros o´ptimos **W***∗* de **f** que minimizen el error entre la predicci´on **y**ˆ = **f**(**W***,* **x**) y la etiqueta real **y**:

**W***∗* = arg m´ın *C*(**Y***,* **Y**ˆ ) = 1

**W**

*N*

*L*(**y**(*i*) *−* **f**(**W***,* **x**(*i*)))*.* (20)

*i*

Dependiendo de qu´e **f**(**W***,* ) usemos, los parametros ser´an unos u otros. En Deep Learning, la estructura de **f** es una red neuronal. En este caso, el enfoque Bayesiano no parece ser necesario pues el objetivo es obtener los par´ametros de una funci´on para que transforme los datos de entrada en los de salida. No obstante, este enfoque no proporciona ideas o pistas sobre c´omo debe ser **f** ni qu´e funci´on de *p´erdidas L* usar. Por ejemplo:

*·*

**f**(**W***,* **x**) = **Wx**

donde **W***∗* = (*w*1*∗, w*2*∗*)*T* son, despu´es de optimizar: (*w*1*∗, w*2*∗*)*T* = (2*.*1*,* 1*.*0)*T* . De manera que si **x***test* = (1*,* 3)*T* , entonces:

*−*

**y**ˆ = **f**(**W***∗,* **x**

*test*

) = 2*.*1 *−*1*.*0 ] *·* 1 l = *−*0*.*9

##### f con par´ametros estoc´asticos (distribuciones)

3

En este otro caso, bastante m´as interesante, no buscamos unos par´ametros sino distribuciones. Es decir que cada vez que evaluemos **W**, obtendremos valores distintos pues cada uno de los par´ametros de **W** son distribuciones.

Por ejemplo, para evaluar la misma funci´on anterior

**f**(**W***,* **x**) = **Wx***,*

debemos primero muestrear de p(**W**), i.e., **w** = (*w*1 p(**W**1)*, w*2 p(**W**2))*T* . Si suponemos que los par´ametros siguen todos alguna distribuci´on particular como una normal p(**W**) = *N* (0*, σ*I) y son independientes entre s´ı, entonces cada vez que evaluemos *w*1*, w*2 tendremos valores centrados en 0 y con des- viaci´on t´ıpica 1. Esto se traduce en que la salida **f** tambi´en es una variable aleatoria y consideraremos pues su valor esperado:

*∼ ∼*

**y**ˆ = Ep [**f**(**W***,* **x**)] = Ep [**Wx**]

**W W**

que se puede aproximar usando un nu´mero de muestras *t* = 1*, ..., T* de p(**W**)

E[**Wx**] *≈* 1 **w x** = 1 *w*

*T*

*t*

*T*

1

*t*

*t*

*w* ] *·* 1 l

= 1 ( 2*.*22 *−*0*.*96 ]

*T*

¿

2

*t*

3

1

*·* 1 l + *...* + 1*.*99 *−*1*.*06 ]

*·* 1 l)

de tal forma que l´ım*t→∞ t* **w***t***x** = E[**Wx**]. Esta aproximaci´on que converge al valor esperado en el l´ımite es conocida como *integraci´on de Montecarlo*.

3

3

*T*

#### Inferencia Variacional

Hemos visto que lo fundamental para hacer predicciones (inferencia) desde el punto de vista Bayesiano es el muestreo de distibuciones. Adem´as, el nu´mero de muestras que deben realizarse para obtener resultados correctos es mu- chas veces impracticable. Pasamos pues a aproximar dichas distribuciones por distribuciones variacionales. Es decir, sustituiremos las distribuciones de las que es muy dificil muestrear por funciones param´etricas que lo sean muy f´acilmente y que adem´as **¡no dependan de los datos!** Dichas funciones var´ıan en funci´on de sus par´ametros, que se conocen como par´ametros *varia- cionales*. En particular queremos encontrar una distribucion variacional que aproxime la probabilidad a posteriori q***θ*** (**W**) p(**W X***,* **Y**), pues es la que necesitamos para predecir segu´n ([17](#_bookmark276)).

*≈ |*

Llegados a este punto es importante entender que los par´ametros **W** de los que depende **f**, son los par´ametros del modelo (distribuciones) y, por otro lado, que ***θ*** son los par´ametros variacionales con los que queremos aproximar p(**W** *|* **X***,* **Y**). Es decir, son par´ametros distintos que relacionaremos m´as adelante. Una vez obtenida q***θ*** (**W**) de la que podemos muestrear f´acilmente **w***t ∼* q***θ*** (**W**), la equaci´on ([18](#_bookmark277)) se aproxima como

p(**y***test |* **x***test,* **X***,* **Y**) *≈* p(**y***test |* **x***test,* **W**)q***θ*** (**W**)d**W** (21)

El problema reside ahora en encontrar q***θ*** (**W**). Una soluci´on es definir una medida entre ambas distribuciones, la real p(**W** *|* **X***,* **Y**) y la aproximada q***θ*** (**W**), como por ejemplo la Divergencia Kullback-Leibler (KL), y minimizar dicha medida. Esto nos lleva a un problema de optimizaci´on tal que:

*θ∗* = arg m´ın KL(q***θ*** (**W**)*||*p(**W** *|* **X***,* **Y**)) (22)

*θ*

Este problema resulta ser equivalente a maximizar la conocida ELBO (Evi- dence Lower Bound). Para comprobarlo, aplicamos la desigualdad de Jensen:

log(p(**X***,* **Y**)) = log ( p(**X***,* **Y***,* **W**)*d***W**

= log (

**W**

p(**X,Y,W**) q(**W**) *d***W**

q(**W**)

= log (Eq p(**X***,* **Y***,* **W**) l *≥* Eq log (p(**X***,* **Y***,* **W**) l

**W**

q(**W**) q(**W**)

de manera que la cota inferior de log(p(**X***,* **Y**)) es

ELBO =

**W**

q(**W**) log p(**X,Y,W**) *d***W***.* (23)

q(**W**)

(

Desarrollando la expresi´on de la Divergencia KL [22](#_bookmark281) tenemos:

KL(q (**W**)*||*p(**W***|***Y***,* **X**)) = q (**W**) log ( q*θ* (**W**)

*θ θ*

*d***W**

**W** p(**W***|***X***,* **Y**)

= *−*  q (**W**) log (p(**W***|***X***,* **Y**) *d***W**

**W**

*θ*

q*θ* (**W**)

= *−*

q (**W**) log ( p(**X***,* **Y***,* **W**) *d***W**

= *−* q (**W**) log (p(**X***,* **Y***,* **W**) *d***W**

**W**

*θ*

q*θ* (**W**)

**W**

p(**X***,* **Y***,* **W**)*d***W**

*θ*

q*θ* (**W**)

**W**

+ q*θ* (**W**) log (p(**X***,* **Y**) *d***W**

= *−* q (**W**) log (p(**X***,* **Y***,* **W**) *d***W** + log (p(**X***,* **Y**) *.*

**W**

*θ*

q*θ* (**W**)

**W**

y finalmente, introduciendo ([23](#_bookmark282) ), obtenemos: KL(q*θ* (**W**)*||*p(**W***|***Y***,* **X**)) = *−*ELBO + log (p(**X***,* **Y**) *.*

Obviando el t´ermino de la Evidencia log(p(**X***,* **Y**)) pues no depende de *θ* y cambiando el signo a la equaci´on, tenemos:

donde

arg m´ın *J*(*θ,* **W**)

*θ*

*J*(*θ,* **W**) = *−* q (**W**) log (p(**X***,* **Y***,* **W**) *d***W**

**W**

*θ*

q*θ* (**W**)

= *−*

q (**W**) log (p(**Y***|***X***,* **W**)p(**X***|***W**)p(**W**) *d***W**

*θ*

**W**

= *−*

q*θ* (**W**)

q*θ* (**W**) log (p(**Y***|***X***,* **W**) *d***W**

+ q (**W**) log (q*θ* (**W**) *d***W**

**W**

*θ*

p(**W**)

**W**

= *−* q*θ* (**W**) log (p(**Y***|***X***,* **W**) *d***W** + KL(q*θ* (**W**)*||*p(**W**)

**W**

Finalmente, la expresi´on a minimizar se compone de dos t´erminos, donde el primero corresponde a la verosimilitud y el segundo har´a las veces de t´ermino de regularizaci´on si asumimos que:

KL(q*θ* (**W**)*||*p(**W**)) = *λ||θ||*2*.*

#### Dropout

Dropout es una t´ecnica muy utilizada en Deep Learning que tiene como objetivo principal reducir el sobreajuste en la red. El Dropout es considerado por tanto una t´ecnica de regularizaci´on. Su aplicaci´on es simple. En cada iteraci´on de la etapa de entrenamiento, algunos coeficientes de la red son anulados, es decir, reemplazados por 0. El criterio de anulaci´on es aleatorio, en concreto, existe una probabilidad *p* de ser anulados. Esto proporciona en cada iteraci´on del entrenamiento una configuraci´on distinta de la red, en realidad, una sub-red. De forma intuitiva, podemos imaginar que en cada iteraci´on estamos entrenando un modelo distinto. La ventaja es por tanto, tener distintos modelos entrenados con el mismo fin, y por lo tanto se pueden aplicar t´ecnicas de votaci´on u otras para mejorar los resultados. Sin embargo, estos modelos no son del todo independientes, pues comparten pesos. Lejos de ser esto una desventaja, se aproveca este hecho para hacer una aproximaci´on

de una salida combinada, sin m´as que desactivar el Dropout y usar todos los pesos de la red.

Una manera de interpretar el Dropout es considerar que los pesos de la red, no son escalares, sino distribuciones, en concreto una distribuci´on de *Bernoulli*(*p*), con probabilidad *p*.

## XXII Clase

#### II Pr´actica en Pytorch.

##### 8.1.1. Denoising con TV

Esta clase se desarrolla en el material de Pr´acticas.

## XXIII Clase

#### III Pr´actica en Pytorch.

Esta clase se desarrolla en el material de Pr´acticas.

## XXIV Clase

#### IV Pr´actica en Pytorch.

Esta clase se desarrolla en el material de Pr´acticas.

## Referencias

1. George Cybenko. Approximation by superpositions of a sigmoidal fun- ction. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
2. Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
3. Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.
4. David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat’s striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.