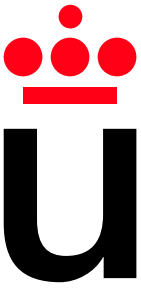


Universidad
Rey Juan Carlos

Escuela Técnica Superior de Ingeniería
Informática

Grado en Matemáticas

Curso 2023-2024

Trabajo Fin de Grado

INNOVACIÓN DOCENTE EN MATEMÁTICAS: ÁLGEBRA Y VISIÓN ARTIFICIAL

Autora: Marta Villaplana Fernández
Tutor: Emanuele Schiavi

Agradecimientos

A mi tutor, Emanuele.

A mi familia, en especial a mis padres y a mi hermana Virginia, que me han brindado con su apoyo y amor incondicionales durante todos los años de carrera y han sabido darme fuerzas cuando el cansancio, la inseguridad o la incertidumbre no me dejaban avanzar.

A mis amigos y compañeros de carrera, con los que tanto he compartido, por haberme escuchado y apoyado siempre.

Y a Alex, quien más me ha acompañado en este proceso, que gracias a su comprensión y a su confianza en mí me ha hecho el camino y la vida más fáciles.

Resumen

La mayoría de los planes de estudios universitarios del grado de Matemáticas tienden a enfocarse en la parte teórica con una gran cantidad de contenido abstracto y conceptual. Aunque la teoría resulta imprescindible para comprender los fundamentos de las matemáticas, la falta de aplicación práctica de estos conceptos en situaciones reales o problemas concretos puede resultar desmotivante para los estudiantes, lo que disminuye su interés y entusiasmo por la carrera, y, además, puede dificultar la transición de la universidad al mundo laboral al no disponer de la experiencia práctica necesaria.

El objetivo del presente trabajo es la realización de una propuesta de innovación docente por medio de la aplicación de la visión artificial para explicar conceptos matemáticos, fomentando, de esta forma, la motivación en el aprendizaje. Para ello, se ha planteado una propuesta de metodología basada en la impartición de seminarios docentes, la realización de prácticas grupales y la entrega de una práctica individual final. Con ello se pretende involucrar al alumno al hacerle partícipe del desarrollo de las aplicaciones de los conceptos que va adquiriendo.

En este trabajo se propone la Descomposición en Valores Singulares (SVD) como ejemplo modelo de un seminario a impartir durante un curso de Álgebra Lineal, con las prácticas correspondientes realizadas en Matlab, que se llevarían a cabo a lo largo del curso. La temática se enmarca dentro de la asignatura de Álgebra Lineal como una metodología integradora dentro de la propia metodología docente y evaluadora del curso. Se emplearían dos sesiones de dos horas cada una en las que se presentan las propiedades teóricas de esta importante técnica algebraica y tres de sus posibles aplicaciones, como son la compresión de imágenes digitales, el reconocimiento facial (eigenfaces) y la eliminación de ruido gaussiano (denoising).

Palabras clave:

- Visión Artificial
- Motivación en el aprendizaje
- Descomposición en Valores Singulares (SVD)
- Matlab

Abstract

Most undergraduate mathematics curricula primarily emphasize the theoretical aspects, with a substantial amount of abstract and conceptual content. While theory is crucial for grasping the fundamentals of mathematics, the absence of practical applications in real-world situations or concrete problem-solving can lead to reduced student motivation. This lack of practical context can dampen their enthusiasm for the subject and potentially hinder their transition from university to the workforce, as they may lack the practical experience required for real-world challenges.

The aim of this project is to promote teaching innovation by applying artificial vision to illustrate mathematical concepts, thereby enhancing student motivation in the learning process. To achieve this, we have designed a methodology centered around delivering teaching seminars, conducting group practices, and assigning a final individual project. The objective is to engage students actively by involving them in the practical application of the concepts they are acquiring.

In this project, Singular Value Decomposition (SVD) is presented as a model example in a seminar, accompanied by corresponding practical exercises conducted in Matlab. These exercises will span the duration of the course. The topic is situated within the broader subject of Linear Algebra, serving as an integral component of the course's methodology.

The theoretical properties of this significant algebraic technique, along with three of its potential applications, are covered in two sessions, each lasting two hours. These applications include digital image compression, facial recognition (eigenfaces), and the reduction of Gaussian noise (denoising).

Key words:

- Artificial Vision
- Motivation in learning
- Singular Values Descomposition (SVD)
- Matlab

Índice de contenidos

Índice de figuras

Índice de códigos

1 Antecedentes

La idea de este Trabajo de Fin de Grado surge a partir de la asignatura Ecuaciones en Derivadas Parciales (EDP) que el profesor Emanuele Schiavi, del Departamento de Matemáticas Aplicadas, imparte en el tercer curso del Grado de Matemáticas en la Universidad Rey Juan Carlos.

En la primera clase de la asignatura llevó a cabo una presentación del temario y de los contenidos que íbamos a ver durante el curso, pero, además, nos mostró algunas recientes aplicaciones de este tipo de ecuaciones. Una de ellas fue el procesamiento de imágenes digitales.

Con el uso del software Matlab se mostraron algunos ejemplos de aplicación de las técnicas de procesamiento de imágenes que consiguieron cautivar mi atención y mi curiosidad hasta el punto de convertirse en una de las asignaturas que más me gustó y en las que obtuve mejores resultados a lo largo de la carrera.

Fue por este motivo por el que decidí hacer mi TFG con el profesor Emanuele Schiavi como tutor contando, por tanto, con su experiencia docente en distintos grados y másters. Al tiempo, basándonos en mi propia experiencia y en mi perfil académico, estudiante del Doble Grado en Matemáticas y Educación Primaria, surgió la idea de realizar una propuesta de innovación docente utilizando la visión artificial como aplicación para entender conceptos matemáticos de álgebra, cálculo multivariable, ecuaciones diferenciales y cálculo numérico, fomentando así la motivación en el estudio y aprendizaje de las matemáticas.

Teniendo en cuenta las restricciones de espacio impuestas por el formato de un

Trabajo de Fin de Grado, las restricciones de tiempo para la impartición de la materia, así como la falta de experimentalidad en la asignación de créditos para los cursos troncales de matemáticas, la metodología propuesta se estructura en un caso modelo de aplicación en un contexto docente realista adaptado a las nuevas tecnologías. Siguiendo esta metodología es posible generalizar el diseño y desarrollo de seminarios y prácticas a otras asignaturas del currículum de matemáticas.

2

Objetivos

Los objetivos del presente TFG se introducen a continuación.

2.1. Objetivo general

El objetivo principal de este trabajo es realizar una propuesta de innovación docente por medio de la aplicación de la visión artificial para explicar conceptos matemáticos y formentar, de esta forma, la motivación en el aprendizaje.

La propuesta se enmarca en el currículum del grado de Matemáticas, pero se puede aplicar en otros contextos cuales Ingenierías e Informáticas y en estudios de Grado o Máster. En todos los casos el principio es motivar al alumno en el estudio de técnicas matemáticas mostrando aplicaciones relevantes en distintos campos científicos.

2.2. Objetivos específicos

Los objetivos específicos que se plantean para conseguir dicho fin son los siguientes:

1. Desarrollar conocimientos y competencias adquiridas en las asignaturas de Álgebra, Cálculo Multivariable, EDO, EDP y Métodos Numéricos.
2. Relacionar y aplicar conceptos y teorías matemáticas al tratamiento de imágenes digitales.
3. Realizar un puente entre la teoría, el cálculo simbólico, el cálculo numérico y la representación, aplicación y resolución de problemas. Validación e interpretación de resultados.
4. Introducir herramientas de cálculo simbólico y numérico de tipo Matlab y Octave durante las clases y en el trabajo autónomo o grupal. Ejecución de algoritmos y representación gráfica.
5. Caracterización de la Metodología docente: Seminarios (impartidos por el docente) y prácticas de ordenador (trabajo autónomo o grupal).
6. Caracterización de la Metodología de evaluación: Examen parcial, final y entrega de prácticas.

3

Introducción

3.1. La innovación educativa

La formación universitaria española tiene una tendencia demasiado teórica y poco práctica. Diferentes estudios realizados en los últimos años han comparado los puntos de vista centrales de la enseñanza universitaria española con los de universidades de otros países europeos, exponiendo la voluntad de una formación universitaria sustentada con modelos educativos más participativos, a la vez que se incrementen las actividades académicas de tipo práctico [3].

Actualmente la sociedad está inmersa en un proceso de cambio que está impulsado por lo que se conoce como Sociedad de la Información. Nuevas tecnologías, procesos y requerimientos han ido apareciendo y deben tenerse en cuenta en el mundo educativo, y más concretamente en la Universidad, para poder preparar a los alumnos para las transformaciones de la sociedad. Por ello, es necesario un cambio de paradigma educativo que apueste por una educación activa y más transversal en la que se haga uso de las nuevas tecnologías [3].

La innovación educativa debe ser un proceso permanente, creativo e intencional. Requiere la formulación de objetivos concretos sobre qué se quiere enseñar, cómo se quiere enseñar y para qué, que serán la base que permita saber los cambios que necesitan realizarse para lograr tales metas. Innovar en educación es mejorar la calidad de vida de las personas a través del desarrollo integral de sus capacidades. El director del Instituto Internacional de la Unesco para la Educación Superior en América Latina y el Caribe (IESALC), Francesc Pedró, señala: "Es innovación si añade valor al aprendizaje" [4].

La educación superior, especialmente en ingeniería o matemáticas, tiene como uno de sus objetivos primordiales preparar a los alumnos para que tengan capacidad sufi-

ciente para enfrentar los desafíos que su carrera profesional pueda ofrecerles. Por ello, como mencionan Feisel y Rosa [5], para que la práctica educativa sea buena, no solo debe trabajarse la resolución de problemas teóricos, sino que deben plantearse aplicaciones reales que contribuyan al desarrollo del pensamiento crítico y de la capacidad de resolución de problemas prácticos.

La formación de profesionales con capacidades y competencias en investigación e innovación es otra de las metas de la educación superior. Según Gil y Valdés [5], este objetivo puede alcanzarse a través de la realización de prácticas, ya que fomentan la autonomía y la transmisión del conocimiento.

En concreto, en la mayoría de cursos y asignaturas de los Grados o Másteres de Matemáticas, Ingenierías e Informáticas, se le da una gran importancia a la parte teórica, dejando en un segundo plano la práctica y los beneficios que esta supone.

Es por ello por lo que dentro de esta propuesta de innovación docente, tras ver a lo largo del curso problemas y ejemplos modelo sobre los contenidos, se pedirá elaborar una práctica en la que se propone el planteamiento, la resolución y validación de un problema novedoso y original que ilustre algún aspecto de la teoría y aplicaciones no cubierto en la colección de problemas.

Implícitamente la tarea requiere una primera fase de estudio y revisión del material que implica la adquisición de conocimientos. Se trata luego de replicar, a través del uso del cálculo simbólico y numérico, unos procesos físicos sobre la imagen que causan su transformación. La posibilidad de realizar un proceso físico sobre un objeto digital cuyo resultados se pueden visualizar en tiempo real es de por sí motivador para el alumno.

La búsqueda de originalidad y creatividad es también un ingrediente fundamental de la propuesta ya que permite evaluar habilidades que no se suelen medir en los procesos de aprendizaje y que son, sin embargo, de lo más preciado en mucho contextos laborales. Se trata por tanto de dar valor a la capacidad de desarrollo de una idea o propuesta.

Motivado por los ejemplos en los que se usan distintas técnicas matemáticas para conseguir distintos efectos sobre imágenes digitales, el alumno empieza a integrar e identificar la técnica con el proceso físico resultante sobre la imagen. Puede decirse que este marco de trabajo permite la obtención de competencias en el sentido de la definición proporcionada por la Sub. Dirección de Calidad de la Escuela de Master Oficial de la URJC:

....conocimientos son cosas que se aprenden en la asignatura, habilidades son cosas que se hacen con lo que se ha aprendido, competencias son cosas en las que se convierte uno cuando adquiere las dos anteriores.

3.2. Metodologías innovadoras

Diferentes investigaciones confirman que el uso de metodologías innovadoras en la práctica docente incrementa la motivación y participación de los estudiantes, refuerza

su autonomía, al impulsar el pensamiento creativo e innovador, y favorece el desarrollo de competencias transversales y profesionales [6].

3.2.1. Visual Thinking

La Sociedad del Conocimiento es aquella que se desarrolla y se transforma a través del saber humano y del uso de la información, que posee un marcado carácter visual y permite la interpretación y construcción del concepto del mundo que nos rodea. Cada vez es mayor la exposición y el consumo de imágenes y contenidos audiovisuales de los medios de comunicación, las redes sociales, la publicidad, etc. Como González-Alba y Cortés-González exponen "la aparente 'facilidad' y comodidad que conlleva la comprensión de imágenes hacen de ellas un medio útil, práctico y atractivo para transmitir ideas, información, sentimientos..., puesto que convierte a sus receptores en sujetos pasivos" [7].

Gracias a estas características las imágenes pueden emplearse como una herramienta didáctica dentro del ámbito educativo, transformando esta recepción pasiva haciendo de sus receptores sujetos activos. La necesidad de una mejora de la práctica docente y del aprendizaje en sí mismo ha favorecido la aparición de metodologías activas que giran en torno al protagonismo del alumno [7].

Estas metodologías buscan adaptarse a las nuevas tecnologías, el proceso de digitalización o las redes sociales, que forman parte de las transformaciones de la sociedad y los nuevos contextos educativos. Además, pretenden dejar atrás la metodología tradicional de enseñanza creando una nueva relación entre docente y estudiante [7].

El Visual Thinking o pensamiento visual es una técnica metodológica que, en palabras de Garbiñe Larralde [8], "supone la comprensión de una información mediante la visualización estructurada de sus partes". Urchegui et al. [7] lo define como "el razonamiento o conjunto de procesos cognitivos que realizamos de manera específica en torno a la información visual, con los que interpretamos la realidad y que nos conducen a la acción. Una acción que se articula con los lenguajes".

Esta herramienta metodológica tiene como objetivo la mejora de ideas o sistemas, lo que tiene una gran conexión con la innovación debido al constante trabajo con metas difusas o hipótesis dentro de este ámbito. Por ello, el pensamiento visual es una vía muy útil a través de la cual pueden explorarse nuevas áreas, contrastarse hipótesis y, como consecuencia, optimizar la toma de decisiones [6].

Según reflejan numerosos estudios, la información que se observa se retiene más fácilmente, lo que multiplica el factor de memorización con respecto a la información transmitida únicamente mediante la palabra. Otros de los beneficios del uso del Visual Thinking son el desarrollo del pensamiento visual como dimensión cognitiva, la mejora de la atención y la concentración, el incremento de la motivación e implicación en el proceso de aprendizaje al percibirlo desde una perspectiva más divertida o la capacidad de analizar ideas, definir nuevas conexiones o identificar problemas de forma más eficaz [7] [8].

3.2.2. Aprendizaje significativo

El aprendizaje significativo es una tipología y metodología del aprendizaje que puede definirse como aquel que permite construir el propio conocimiento y, además, dotarlo de significado [9].

La teoría del aprendizaje significativo de David Paul Ausubel considera que sólo se produce aprendizaje significativo cuando los conocimientos o la información que se trata de aprender se relacionan de forma no arbitraria y sustantiva con aspectos relevantes y preexistentes en la estructura cognitiva. "El aprendiz sólo aprende cuando encuentra sentido a lo que aprende". En ese sentido, un aprendizaje es significativo cuando aquel que aprende puede otorgarle una utilidad a la nueva información aprendida y relacionarla con un conocimiento previo [10].

Las actividades que le proporcionan experiencia al alumno son las que definen el proceso de aprendizaje significativo, pues es la propia experiencia la que es capaz de producir un cambio en los contenidos anteriores. Para que las actividades sean significativas el alumno debe disfrutar realizándolas, participar en ellas con interés y atención, trabajar con autonomía y seguridad, desafiar a sus propias habilidades y propiciar la creatividad y la imaginación [10].

Esos aspectos relevantes y preexistentes en la estructura cognitiva que se relacionan con los nuevos conocimientos en el proceso de aprendizaje significativo pueden ser, por ejemplo, un símbolo ya significativo, un concepto, una proposición, un modelo mental o una imagen [11].

Las imágenes pueden permitir crear conexiones entre los conocimientos previos y los nuevos aprendizajes. Fomentan, además, el interés y la atención y favorecen la imaginación y la creatividad. Es decir, son actividades significativas y su uso en la práctica educativa puede traer todos los beneficios del aprendizaje significativo.

Algunas de las múltiples ventajas del aprendizaje significativo son la mejora de los resultados académicos y de la calidad del sistema educativo, el incremento de la motivación y participación de los alumnos y mejora de su comportamiento, la realización personal del profesorado y del alumnado durante el aprendizaje o la contribución a un buen clima en el aula [9].

4

Metodología

En este capítulo se presenta la metodología de la propuesta de innovación docente. Esta pretende dar una respuesta a esa necesidad de cambio en el paradigma educativo universitario y, además, servir como una solicitud de un mayor nivel de experimentalidad en el grado de matemáticas.

La falta de aplicación práctica de los conceptos en situaciones reales o problemas concretos genera, no sólo desmotivación y falta de interés en los estudiantes, sino también una sensación de bajo dominio o, incluso, desconocimiento de los contenidos, lo que hace mucho más difícil la incorporación al mundo laboral.

Un mayor nivel de experimentalidad propiciaría una preparación más sólida para enfrentar desafíos matemáticos en la carrera profesional y un mayor desarrollo del pensamiento crítico y de la capacidad de resolución de problemas prácticos, a la vez que aumentaría la motivación del estudiante y, con ello, sus resultados.

La metodología se divide en docente y evaluadora, incluyendo en esa segunda clase la fase de reevaluación o Convocatoria Extraordinaria. En un contexto de Evaluación Continua la fase de reevaluación adquiere una importancia especial ya que permite la adquisición de conocimientos y habilidades (luego competencias) utilizando la retroalimentación del profesor. En el desarrollo de prácticas individuales el proceso de retroalimentación es particularmente efectivo al identificar los puntos débiles del trabajo.

4.1. Metodología docente

En este apartado se describe la propuesta de metodología docente para un curso que tiene como objetivo explicar a los estudiantes los contenidos de diferentes áreas de su carrera de forma que el aprendizaje se realice desde la motivación y que la comprensión de los conceptos tenga un carácter global.

Se asume que los alumnos tienen los conocimientos adquiridos en un tercer curso del Grado de Matemáticas. Sin embargo, los ejemplos y aplicaciones se pueden realizar en distintos contextos y disciplinas en donde las aplicaciones tienen un significado aún mayor al acercar al alumno a las investigaciones y desarrollos de los profesores.

La metodología que se propone consta de tres partes necesarias para la adquisición de competencias y se articula a través de distintas actividades docentes: las clases magistrales, los seminarios y las prácticas.

La primera parte, inicial, de la metodología se basa en la transferencia de los fundamentos teóricos a código, a través de la realización de seminarios impartidos por el docente donde se introduce al alumno al diseño de scripts de Matlab que automatizan la fase de cálculo proponiendo una salida gráfica (output) directamente interpretable. La segunda parte se realiza a través de prácticas grupales, que permiten afianzar los contenidos del seminario, a la vez que se favorece el trabajo en equipo, muy motivador para los alumnos. Finalmente, en una tercera parte, los conocimientos y habilidades se plasman en competencias específicas a través de la entrega de un trabajo original que el alumno sepa validar y, eventualmente, defender en una presentación oral.

Se presentan a continuación las distintas actividades docentes que conforman la estructura de la propuesta.

4.1.1. Seminarios (docentes)

En ellos se introducirá a los estudiantes diferentes conceptos y teorías matemáticas y, a su vez, se relacionarán y aplicarán a la visión artificial y el tratamiento de imágenes digitales. La aplicación concreta al marco de la visión y la participación de los alumnos de forma grupal permite fijar la atención del grupo favoreciendo el desarrollo de la sesión. Para ello, se utilizará el software Matlab, por lo que también se introducirán algunos de los comandos básicos, necesarios para la segunda parte de la propuesta.

Los seminarios, diseñados e impartidos por el profesor en el laboratorio de informática, sirven para introducir a los alumnos al uso del cálculo simbólico y numérico y mostrar (visualizar) la aplicación de los conceptos a casos prácticos. Suelen tener una duración de dos horas y se utilizan para descargar de excesivo contenido teórico algunas sesiones así como para afianzar conceptos previos. Digamos que los seminarios se sitúan en la fase de adquisición de conocimientos y permiten integrar el planteamiento del modelo con su resolución a través de las potente herramientas de cálculo simbólico y numérico de Matlab. Alternativamente y para doble grados de Informática

y Matemáticas sería igualmente posible trabajar en Python.

4.1.2. Prácticas (grupales)

Entramos ahora en la fase de adquisición de habilidades lo que, junto a los conocimientos, aportará competencias específicas al alumno. A la vez que los estudiantes vayan adquieriendo los conocimientos y competencias pertinentes, se les propondrá la realización de prácticas de ordenador. Estas consistirán en resolución de problemas en los que tengan que poner en práctica los conceptos introducidos y explicados durante los seminarios y experimentar por sí mismos las aplicaciones de estos.

El principal objetivo de estas prácticas es aumentar su motivación y, por tanto, propiciar que la adquisición de los aprendizajes sea más significativa. Las prácticas podrán ser realizadas de forma individual o grupal (en grupos de 2 o 3 estudiantes), dando preferencia a esta segunda estrategia de trabajo debido a sus beneficios en el proceso de enseñanza-aprendizaje del alumnado. El trabajo en equipo mejora la atención y la implicación, refuerza la autoestima y potencia la adquisición de conocimientos, al tener que coordinarse e intractuar dentro del grupo para alcanzar un mismo objetivo.

4.1.3. Entrega de un trabajo (individual)

Otra importante acción metodológica que tiene un profundo impacto en el desarrollo y evaluación de las competencias adquiridas por el alumno y en la mejora de resultados académicos consiste en reservar un porcentaje de la evaluación, que puede variar según el curso y el grado, para el desarrollo de una práctica completamente original en la que el alumno, con el conocimiento adquirido durante el curso y sin restricciones de tiempo, pueda mostrar su capacidad, originalidad y creatividad. De esta forma, se propone al alumno una metodología evaluadora que no está sesgada por factores externos (como el ruido o el calor) o internos (cansancio, fatiga, nerviosismo o estado emocional) del alumno durante la realización de un examen, sino que permite mostrar su verdadera capacidad en condiciones de trabajo óptimas, elegidas y gestionadas por el alumno.

Un factor importante del trabajo a presentar es que tiene que estar validado, es decir, el trabajo tiene que ser correcto -formalmente y matemáticamente- para lo cual se certificarán los resultados utilizando Matlab. Esta validación de resultados tiene un impacto profundo en la metodología de trabajo ya que introduce la idea fundamental de que cualquier propuesta tiene que estar validada siendo su exactitud lo que se espera del trabajo de un matemático.

En este sentido, la validación se ve como una parte del marco de trabajo de la matemática aplicada que se fundamenta en las fases de Modelado, Planteamiento del Problema, Aplicación del Modelo, Resolución simbólica o numérica y Validación de Resultados.

El trabajo permite medir muy fácilmente la motivación del alumno, lo que se refleja en propuestas interesantes, originales y elaboradas. Finalmente, destacar que se

favorece el uso del Latex para la redacción y formato de presentación del trabajo lo que acerca el alumno al formalismo y estilo típico de las matemáticas en revistas y artículos científicos. Se utiliza la plantilla del TFG, proporcionada por la Coordinación del Grado, para acercar el alumno a una redacción, limpia, clara, coherente y ordenada, lejos de las típicas hojas de examen con tachones y borrados que confunden al alumno y al profesor en su corrección y análisis de los errores cometidos.

4.2. Metodología evaluadora

4.2.1. Evaluación ordinaria continua

La evaluación será continua y progresiva durante el desarrollo de todo el curso. Constará de tres pruebas de dos tipos diferentes: dos controles y el desarrollo de una práctica. Estas pruebas serán reevaluables en la convocatoria extraordinaria y su nota mínima será un 3.

Primer control: Consiste en una prueba escrita, de evaluación continua, que se realizará al finalizar la primera parte del curso. Supondrá el 40 % de la asignatura.

Desarrollo de una práctica: Los alumnos deberán realizar una práctica original en la que combinarán los contenidos trabajados junto con Matlab, cálculo simbólico y numérico, y Latex, competencias necesarias de la asignatura. De esta forma, demostrarán su capacidad y dominio de la materia, a la vez que reflejan su motivación e interés por la asignatura. Esta práctica supondrá un 20 %.

Segundo control: Se trata de una segunda prueba escrita, también de evaluación continua, que tendrá lugar al finalizar la segunda parte del curso. Supondrá el 40 % de la asignatura.

4.2.2. Evaluación extraordinaria

Aquellos estudiantes que no consigan alcanzar la nota mínima de 5 sobre 10 deberán realizar una evaluación extraordinaria en la que reflejen el nivel de adquisición de las competencias del curso. En la convocatoria se evaluarán todas las pruebas en las que no se alcance la nota mínima, puesto que es un requisito necesario para aprobar. Si en ninguno de los dos controles se alcanza la nota mínima, se realizará un examen final en el que se evaluarán todos los contenidos del curso, correspondiéndole un 80 % de la nota.

4.2.3. La autoevaluación

La autoevaluación o *evaluación a la carta* es una potente herramienta para el docente que ha sido historicamente negada o, simplemente, no considerada. Sin embargo, mide la valoración del alumno con respecto al trabajo entregado. Lo importante es

que la autoevaluación realizada por el alumno *resuene* al docente para una correcta valoración del trabajo y de la madurez científica y matemática del alumno. Se trata de medir, por tanto, la confianza del alumno en el valor del trabajo presentado, siendo consciente del esfuerzo realizado y de las dificultades encontradas.

La distancia (o diferencia) entre una calificación objetiva, basada en la experienza docente, y la autoevaluación puede indicar la capacidad autocrítica del alumno.

4.3. Seminario Parte I: Descomposición por Valores Singulares (SVD)

En este apartado se propone y describe la primera parte de lo que será un ejemplo completo de uno de los seminarios que formarían parte del curso de Álgebra Lineal. Este servirá como modelo para el resto del curso. Otras propuestas, en la misma dirección, se pueden realizar en el contexto del Cálculo Multi Variable, las Ecuaciones en Derivadas Parciales y el Cálculo Numérico. Las características y los objetivos del seminario serán aquellos que han sido descritos en el apartado 4.1.1, adaptándose tanto al número de alumnos como a las características individuales del grupo. Para esta primera parte del seminario se destinará una sesión académica, cuya duración son dos horas.

En primer lugar, empezaremos con el desarrollo teórico del seminario, en el que se explica la técnica de Descomposición en Valores Singulares.

La Descomposición por Valores Singulares (SVD, del inglés *singular value descomposition*) es una técnica algebraica que tiene importantes aplicaciones en el campo del procesamiento de imágenes y, en general, en el análisis de grandes cantidades de datos (bigdata analysis).

Se trata de un tema que no se incluye en el temario de Álgebra Lineal representando un puente hacia el Álgebra numérica y las aplicaciones. Sin embargo, los fundamentos matemáticos necesarios para la comprensión del concepto son los típicos que aparecen en el curso de Álgebra Lineal, por lo tanto, representa un tema especialmente interesante para su desarrollo y aplicación a la Visión Artificial a través de la actividad didáctica del seminario y su implementación en las prácticas desarrolladas por los alumnos.

Los primeros resultados y propiedades matemáticas de la SVD aparecen en 1873 y 1874 y se deben a las contribuciones de varios matemáticos ilustres. Eugenio Beltrami (1835-1899), Camille Jordan (1838-1921) y James Joseph Sylvester (1814-1897) los obtuvieron a partir del álgebra lineal. Más tarde, Erhard Schmidt (1876-1959) y Hermann Weyl (1885-1955) los ampliaron a través del estudio de las ecuaciones integrales [12]. La demostración de la existencia de una descomposición SVD para cualquier matriz real o compleja es de Eckart y Joung, en 1936, [13].

El desarrollo, más reciente, de métodos numéricos eficientes para su cálculo se debe al trabajo de Golub (1965), lo que ha permitido el rápido desarrollo de las aplicaciones de la técnica a problemas de análisis de datos [14].

Una simple observación inicial justifica el interés, primero teórico y luego práctico, de la teoría de los valores singulares.

La teoría de los autovalores no puede aplicarse directamente sobre una matriz no cuadrada, por lo que no será posible calcular su espectro ni su factorización espectral. Sin embargo, la Descomposición por Valores Singulares ofrece una generalización del concepto de autovalor al caso de matrices no cuadradas. La generalidad de este resultado es máxima ya que permite trabajar (codificar y procesar) con imágenes de cualquier formato y tamaño adaptándose a todos los tipos de sistemas de visión artificial.

El siguiente teorema demuestra que cualquier matriz real admite una descomposición en valores singulares.

Teorema 1. *Sea $A \in \mathbb{R}^{m,n}$. Entonces existen matrices*

$$U \in \mathbb{R}^{m,m}, \qquad \Sigma \in \mathbb{R}^{m,n}, \qquad V \in \mathbb{R}^{n,n}$$

tales que

$$A_{m,n} = U_{m,m}\Sigma_{m,n}V_{n,n}^T$$

donde U y V son matrices ortogonales y Σ es una matriz diagonal no necesariamente cuadrada (ya que tiene las mismas dimensiones que A).

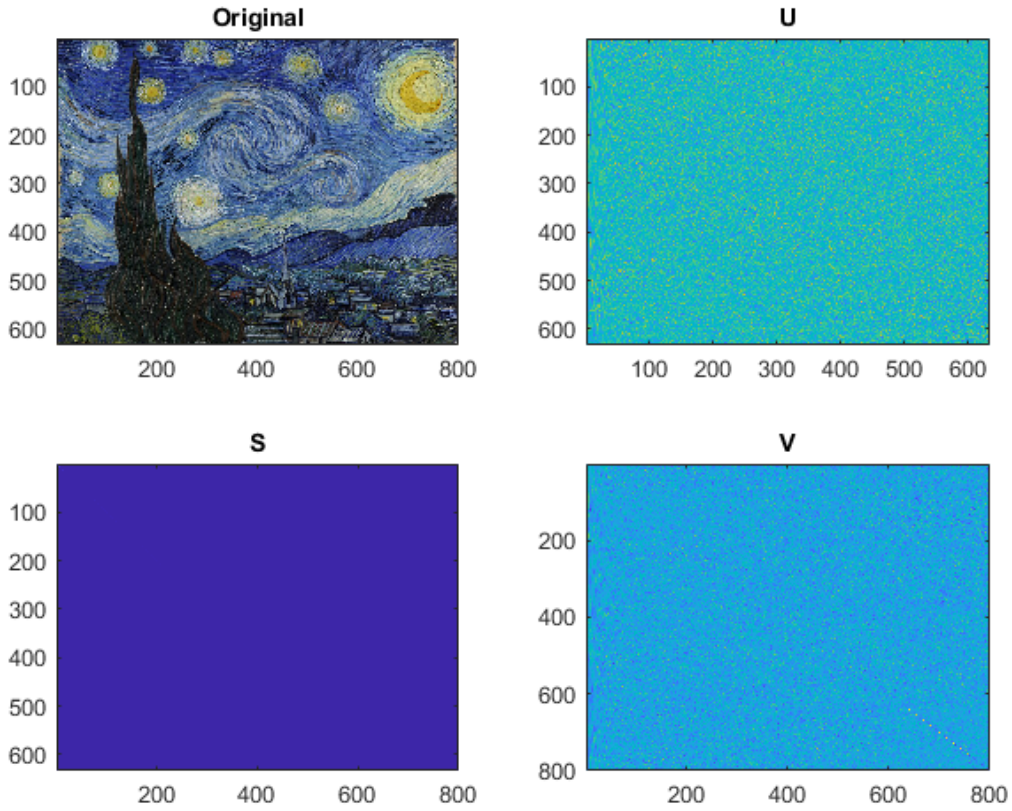


Figura 4.1: Descomposición en valores singulares (SVD) de una imagen del cuadro La noche estrellada de Van Gogh.

El teorema general de Eckart y Joung, [13] vale para cualquier matriz cuadrada, real o compleja y la hipótesis de ortogonalidad se sustituye, en el caso complejo, por la de matrices unitarias[1].

[1]Aunque no parezca intuitivo pensar en imágenes complejas existen modalidades de imagen

Definición 1. *Los elementos de la diagonal de* Σ *son los Valores Singulares de la matriz A:*

$$\sigma_1 \geq ... \geq \sigma_r \geq 0$$

Referente a la unicidad de la descomposición SVD observamos que los valores singulares son únicos pero las matrices U y V no.

Definición 2. *Las columnas de U son los vectores singulares por la izquierda y las columnas de V son los vectores singulares por la derecha.*

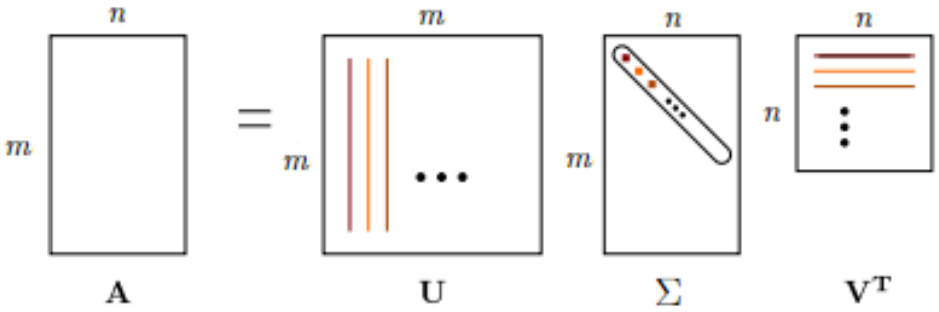


Figura 4.2: La descomposición en valores singulares (SVD). Cada valor singuar de Σ tiene asociado un vector singular por la izquerda en U y un vector singular por la derecha en V. *Fuente: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations, [1].*

En efecto, se verifica que:

$$A_{m,n}\mathbf{v}_{n,1} = \sigma\mathbf{u}_{m,1}$$

para todo vector singular $\mathbf{u} \in \mathbb{R}^m$ y $\mathbf{v} \in \mathbb{R}^n$. Esta descomposición en valores singulares puede interpretarse desde el punto de vista de una descomposición espectral.

El siguiente teorema tiene interés teórico y práctico al indicarnos el camino para el cálculo de los vectores singulares a través de los conocimientos adquiridos con el cálculo de los autovectores.

Teorema 2. *Sea* $A \in \mathbb{R}^{m,n}$ *y sean* $U \in \mathbb{R}^{m,m}$ *y* $V \in \mathbb{R}^{n,n}$ *matrices que verifican la descomposición matricial:*

$$A_{m,n} = U_{m,m}\Sigma_{m,n}V^T_{n,n}$$

médicas, del tipo imágenes de Resonancia Magnética, cuyos datos se adquieren en el espacio complejo de frecuencias de Fourier. Tras anti-trasformar estos datos *en crudo* al espacio de la imagen los datos siguen siendo complejos. Será sólo a la hora de generar la imagen para estudio y análisis médico cuando se considerará el módulo de la imagen lo que es necesariamente real.

Las columnas de U (vectores singulares por la izquierda) son los autovectores de la matriz

$$AA^T$$

Las columnas de V (vectores singulares por la derecha) son los autovectores de la matriz

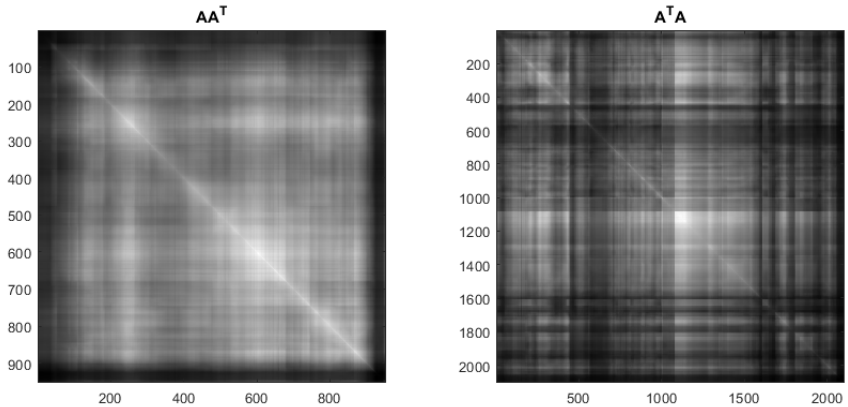
$$A^TA$$

En efecto, por la definición de la matriz traspuesta de un producto y por la propiedades de las matrices ortogonales podemos escribir

$$A^TA = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T$$

y deducir que la matriz V diagonaliza la matriz A^TA y por eso los vectores v_j son autovectores de A^TA. Razonando de forma análoga los alumnos tienen que resolver el siguiente ejercicio.

(a) Cuadro Guernica de Picasso. La imagen se codifica como una matriz A de dimensiones 953 × 2105.



(b) Matrices AA^T, de dimensiones 953 × 953, y A^TA, de dimensiones 2105 × 2105.

Figura 4.3: Representación de las matrices cuadradas y simétricas AA^T y A^TA.

Ejercicio 1. *Demostrar que la matriz U diagonaliza la matriz* AA^T.

Lo que lleva a las ecuaciones

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T$$

Utilizando las descomposiciones de A^TA y AA^T y observando que los autovalores de $\Sigma^T\Sigma$ y $\Sigma\Sigma^T$ son iguales, deducimos que las columnas de V son los autovectores de A^TA y las columnas de U son los autovectores de AA^T.

Sea $r = \text{rank(A)}$. La matriz A tendrá entonces r valores singulares no nulos (positivos). Los valores singulares no nulos de A son la raíz cuadrada de los autovalores de A^TA, que coinciden con los de AA^T. Es decir

$$\sigma_1^2 \geq ... \geq \sigma_r^2 > 0$$

son los autovalores no nulos de las matrices A^TA y AA^T.

A través del siguiente ejemplo los alumnos podrán observar paso a paso cómo se realiza de forma manual la descomposición SVD.

Ejemplo 1. *Dada la matriz*

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}_{3,2}$$

Calcular su Descomposición en Valores Singulares.

Calculamos

$$A^TA = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}_{2,2}$$

cuyos autovalores son

$$\lambda_1 = 3, \qquad \lambda_2 = 1$$

Los valores singulares de A son

$$\sigma_1 = \sqrt{3}, \qquad \sigma_2 = 1$$

y se tiene (observando que la fila de ceros da consistencia dimensional)

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix}_{3,2} = \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}_{3,2}$$

Los autovectores correspondientes de A^TA son

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Ya es posible construir la matriz V de la descomposición

$$V = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

Utilizando las relaciones

$$A\mathbf{v}_1 = \sigma_1 \mathbf{u}_1, \qquad A\mathbf{v}_2 = \sigma_2 \mathbf{u}_2$$

se calculan los vectores ortogonales

$$\mathbf{u}_1 = \frac{1}{\sqrt{6}} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \mathbf{u}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

Tras calcular los valores y vectores singulares podemos comprobar que se verifica la descomposición

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T$$

en donde los valores singulares *ponderan* la importancia relativa de cada sumando (modo). Se trata en realidad de un resultado más general y fundamental que se recoge en el Teorema 3.

Tras realizar una SVD en papel, un posible ejercicio consiste en calcular la SVD de una matriz dada mediante el uso de Matlab.

Ejercicio 2. *Dada la matriz*

$$A = \begin{pmatrix} 1 & -2 & 2 \\ -2 & 4 & -1 \\ -1 & -1 & 0 \\ 0 & 2 & 1 \end{pmatrix}_{4,3}$$

Calcular su Descomposición en Valores Singulares haciendo uso de Matlab.

En la figuras 4.4 y 4.5 se muestra una gráfica visual de cada una de las matrices resultantes de la descomposición SVD de la matriz A del Ejercicio 2. En la representación podemos observar que sólo hay 3 valores singulares positivos, luego $rank(A) = 3$. En la figura 4.4 la matriz diagonal Σ se ha complementado con una fila de ceros para tener las mismas dimensiones que la matriz de entrada A.

Sin embargo, en la figura 4.5 se aplica la opción *'econ'* de Matlab para calcular la SVD truncada, que nos da una versión compacta de la SVD. En este caso, la matriz diagonal Σ se construye únicamente con los valores singulares no nulos de A, eliminando la fila complementaria de ceros. Del mismo modo, podemos observar que en las columnas de la matriz U se ha eliminado la última de ellas, que se correspondía con un espacio vectorial complementario y ortogonal a las tres primeras columnas que abarca U.

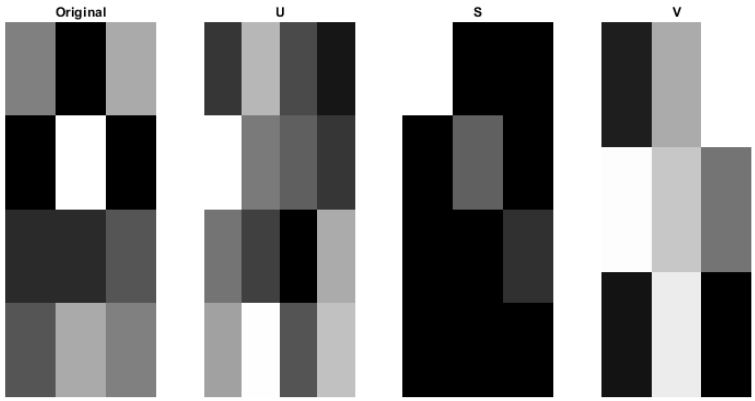


Figura 4.4: Matrices resultantes de la descomposición SVD de la matriz $A_{4,3}$ del *Ejercicio 2*. Las dimensiones son $U_{4,4}$, $\Sigma_{4,3}$, $V_{3,3}$.

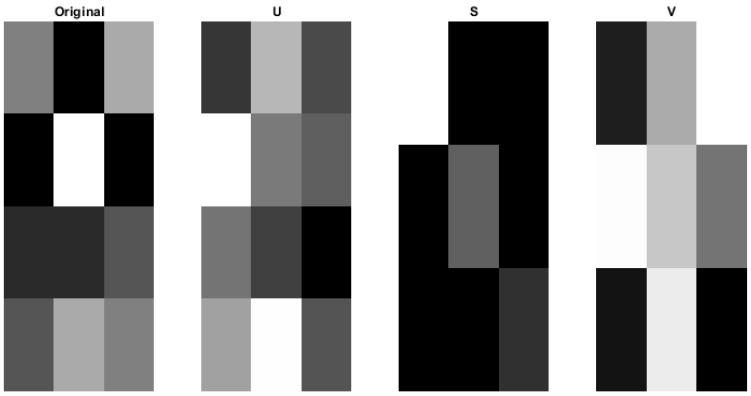


Figura 4.5: Matrices resultantes de la descomposición SVD aplicando la opción *'econ'* de la matriz A del *Ejercicio 2*. Las dimensiones son $U_{4,3}$, $\Sigma_{3,3}$, $V_{3,3}$. Se ha codificado la misma información pero eliminando la última columna de U y la última fila de Σ,

De forma más general, se tiene el siguiente resultado:

Teorema 3. *Sea $A \in \mathbb{R}^{m,n}$, y sean*

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$$

sus valores singulares (los no nulos son r). Sean $u_1, ..., u_r$ sus vectores singulares por la izquierda y $v_1, ..., v_r$ sus vectores singulares por la derecha. Entonces la factorización

$$A = U\Sigma V^T$$

se puede escribir en la forma:

$$A = \sigma_1 u_1 v_1^T + ... + \sigma_r u_r v_r^T \tag{4.1}$$

Además, se tiene:

1. *rank(A)* $= r \leq n$.
2. *La norma 2 de la matriz nos da el valor singular máximo*

$$||A||_2 = \sigma_1 \tag{4.2}$$

Si A es cuadrada (m=n) y no singular su condicionamiento es

$$cond(A) = ||A||_2||A^{-1}||_2 = \frac{\sigma_1}{\sigma_n} \tag{4.3}$$

3. *Los vectores* $u_1, ..., u_r$ *forman una base ortonormal de* $\mathcal{R}(A)$.
4. *Los vectores* $u_{r+1}, ..., u_n$ *forman una base ortonormal de* $\mathcal{N}(A^T)$.
5. *Los vectores* $v_1, ..., v_r$ *forman una base ortonormal de* $\mathcal{R}(A^T)$.
6. *Si* $r < n$ *los vectores* $v_{r+1}, ..., v_n$ *forman una base ortonormal de* $\mathcal{N}(A)$.

Más adelante, en las aplicaciones a la visión artificial, observaremos que, a la hora de realizar la reconstrucción de una imagen (matriz), el valor singular máximo (4.2) nos proporciona la información de baja frecuencia, es decir, la estructural. Detalles y texturas, asociadas a las altas frecuencias de la señal se codifican en los valores singulares positivos pero pequeños.

Por su parte, el condicionamiento de A (4.3) nos da información clave para la resolución numéricas de sistemas lineales y problemas de mínimos cuadrados asociados a regresiones lineales. Si hay un condicionamiento malo, es decir, muy grande, significa que ante pequeñas perturbaciones en el sistema, la solución diferirá mucho, perdiendo, por tanto, estabilidad en el sistema. Esta observación es clave a la hora de aplicar algoritmos de álgebra numérica para la resolución de sistemas lineales. Para condicionamiento bajo utilizaremos métodos directos (del tipo factorizaciones LU, QR) y para condicionamiento elevado usaremos algoritmos iterativos (del tipo Jacobi, Gauss-Seidel, Gradiente, Gradiente Conjugado) basados en el precondicionamiento de la matriz del sistema.

En relación al Teorema 3 se les plantea a los alumnos el siguiente ejercicio donde las propiedades 3, 4, 5 y 6 del teorema permiten el análisis de sistemas lineales asociados a datos desconocidos.

Ejercicio 3. *Sea dada la matriz del ejemplo*

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}_{3,2}$$

Calcular la dimensión y una base de los espacios $\mathcal{N}(A) \in \mathbb{R}^2$, $\mathcal{R}(A) \in \mathbb{R}^3$, $\mathcal{N}(A^T) \in \mathbb{R}^3$, $\mathcal{R}(A^T) \in \mathbb{R}^2$. *Clasificar las aplicaciones asociadas* f_A, f_{A^T} *en inyectivas, suprayectivas y biyectivas. Determinar una base de estos subespacios vectoriales. Utilizar la caracterización de los espacios núcleo e imagen de las aplicaciones asociadas a las matrices A y*

A^T para estudiar la existencia y unicidad de los sistemas $A_{3,2}x_{2,1} = b_{3,1}$, $A^T_{2,3}x_{3,1} = \hat{b}_{2,1}$ siendo b y $\hat{b}$ vectores de datos desconocidos.

Finalmente, tras haber entendido el proceso de descomposición aplicado a una matriz de pequeñas dimensiones, se propone a los alumnos que realicen el siguiente ejercicio en el que tendrán que utilizar el cálculo simbólico para calcular, de forma automática, la descomposición SVD de una matriz de grandes dimensiones.

Ejercicio 4. *Dada una matriz de dimensiones muy grandes. Calcular su descomposición en Valores Singulares haciendo uso de Matab.*

Para generar una matriz de gran tamaño en Matlab puede utilizarse el comando **rand(n,m)**, que devuelve una matriz de n por m números aleatorios.

Propiedades SVD

A partir de los trabajos de los autores mencionados en el apartado anterior, vamos a ver algunas de las propiedades fundamentales de la Descomposición en Valores Singulares. Estas son la Aproximación de matrices de bajo rango (4.3.1), la Pseudo-Inversa de Moore-Penrose (4.3.2) y el Análisis de Componentes Principales (PCA) (4.3.3).

La explicación de cada una de estas propiedades será llevada a cabo a continuación de los fundamentos teóricos de la SVD, por lo que seguirán las mismas caracterísiticas y objetivos del seminario.

4.3.1. Aproximación de matrices de bajo rango

Una de las importantes propiedades que posee la SVD es la capacidad de proporcionar una aproximación de una matriz mediante otra matriz de menor rango. La SVD, en particular, ofrece la mejor aproximación, en términos de la norma de Frobenius y de mínimos cuadrados, de cualquier matriz por otra de las mismas dimensiones, pero de menor rango [15].

Esto es así ya que, si queremos aproximar lo mejor posible una matriz A por una matriz de rango k, la idea sería mantener los k datos "más importantes"para dar la mejor representación de la matriz original. Y eso es precisamente lo que hace la SVD [1].

Como hemos visto en 4.1, la factorización $A = U\Sigma V^T$, con $rank(A) = r$, es equivalente a la expresión

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

donde la SVD expresa a la matriz A como una combinación lineal no negativa de r matrices de rango 1 (los productos externos de los vectores singulares izquierdo y derecho), ponderadas por los valores singulares [16].

Luego, si mantenemos los primeros k términos, la aproximación de rango k propuesta es:

$$\hat{A}_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

donde $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_k \geq 0$. Como suma de k matrices de rango 1, es evidente que la matriz A tiene rango k [16].

Una forma equivalente de ver la aproximación de rango k es

$$\hat{A}_k = U_k \Sigma_k V_k^T$$

donde al computar la SVD, únicamente mantenemos las primeras k filas de V^T (k vectores singulares por la derecha), las k primeras columnas de U (k vectores singulares por la izquierda) y los k primeros valores singulares de Σ [1]. Esta descomposición se conoce como SVD truncada.

En Matlab, esta descomposición se realiza mediante los siguientes comandos:

Código 4.1: Descomposición SVD truncada

```
1 [U, S, V] = svd(A);
2 Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
```

El siguiente resultado, conocido como el teorema de Eckart-Young, justifica que la aproximación de bajo rango anterior A_k es la mejor aproximación de A en términos de la norma de Frobenius y de la norma espectral [17].

Teorema 4. *Para cualquier matriz $A \in \mathbb{R}^{m,n}$ con $rank(A) = r$ y $B \in \mathbb{R}^{m,n}$ con $rank(B) \leq k$, $1 \leq k \leq r$*

$$A_k = \arg \min_{\{B \in \mathbb{R}^{m,n} : rank(B) \leq k\}} \|A - B\|_F \tag{4.4}$$

donde $\arg\min \|A - B\|_F$ denota la distancia mínima de la norma de Frobenius.

La optimalidad de la aproximación mediante SVD de la matriz A se interpreta en la forma:

$$\sqrt{\sum_{i \leq k} \sigma_i^2} \doteq \|A - A_k\|_F \leq \|A - B\|_F$$

$$\sigma_{k+1} \doteq \|A - A_k\|_2 \leq \|A - B\|_2$$

donde A_k es la aproximación de rango k derivada de la SVD de A [16].

Se dice que la matriz reconstruida A_k es óptima (en sentido de mínimos cuadrados) para matrices de rango k porque satisface la siguiente condición

$$\|A - A_k\|_F^2 = Tr\{(A - A_k)(A - A_k)^T\} = \min_X \|A - X\|_F^2$$

para el conjunto de matrices X de rango menor o igual que k. La calidad de la reconstrucción viene dada por la relación entre los primeros k valores propios (es decir, los valores singulares al cuadrado) y la suma de todos los valores propios. Esta cantidad se interpreta como la proporción reconstruida o la varianza explicada, y corresponde a la inversa de la cantidad minimizada por la ecuación anterior. La calidad de la reconstrucción también puede interpretarse como el coeficiente de correlación al cuadrado entre la matriz original y su aproximación [15].

4.3.2. Pseudo-Inversa de Moore-Penrose

Los sistemas lineales y su resolución eficiente representan una parte fundamental de las matemáticas que se utiliza en una amplia variedad de campos, incluyendo la física, la ingeniería, la estadística, la economía y la informática.

Dado un sistema lineal de ecuaciones $Ax = b$. Si A es una matriz cuadrada e invertible (es decir, su determinante es distinto de cero), entonces existe una solución única x para cada b. Sin embargo, cuando A es singular o rectangular puede haber una, ninguna o infinitas soluciones, dependiendo del vector b y de los espacios de columnas y filas de A [18].

Consideramos el sistema *indeterminado*, donde $A \in \mathbb{R}^{m,n}$ y $m \ll n$, de manera que el número de ecuaciones es menor que el de incógnitas. Es probable que este tipo de sistema tenga rango de columna completo, ya que tiene muchas más columnas que las necesarias para una base linealmente independiente. En general, si un sistema tiene rango de columna completo, entonces hay infinitas soluciones x para cada b. El sistema se denomina *indeterminado* porque no hay suficientes valores en b para determinar unívocamente la x de dimensión superior.

Del mismo modo, consideramos el sistema *sobredeterminado*, donde $m \gg n$, de modo que hay más ecuaciones que incógnitas. Esta matriz no puede tener un rango de columna completo, por lo que está garantizado que hay vectores b que no tienen solución x. De hecho, sólo habrá solución x si b está en el espacio de columnas de A, i.e., $b \in col(A)$.

En el caso *sobredeterminado*, en general, no existe solución y el objetivo para poder resolver el sistema es encontrar la "mejor"solución que minimice la diferencia entre Ax y b, es decir, que minimice el error cuadrático $||Ax-b||_2^2$. Esta es la denominada solución de *mínimos cuadrados*. Observar que la solución de mínimos cuadrados también minimiza $||Ax-b||_2$. En el caso *indeterminado*, cuando existen infinitas soluciones, para resolver el sistema debemos encontrar la solución x con norma mínima $||x||_2$ de modo que $Ax = b$, que es la denominada solución de *norma mínima* [18].

Para resolver este tipo de problemas lineales resulta muy útil la aplicación de la

Pseudo-Inversa de Moore-Penrose. Este concepto generaliza la noción habitual de inversa de una matriz cuadrada, permitiendo su aplicación a matrices cuadradas singulares o incluso a matrices no cuadradas. Se define por

Definición 3. *Sea $A \in \mathbb{R}^{m,n}$. Entonces existe la matriz Pseudo-Inversa de Moore-Penrose, denotada por $A^\dagger$ y definida por*

$$A^\dagger = \lim_{\epsilon \to 0} (A^T A + \epsilon I_n)^{-1} A^T = \lim_{\epsilon \to 0} A^T (A A^T + \epsilon I_n)^{-1} \tag{4.5}$$

Un algoritmo muy práctico para calcular la Pseudo-Inversa se basa en la fórmula

$$A^\dagger = V \Sigma^\dagger U^T \tag{4.6}$$

donde U, Σ y V son las matrices proporcionadas por la SVD de A y $\Sigma^\dagger$ es la pseudo-inversa de la matriz diagonal Σ, que se construye tomando el recíproco de los elementos no nulos de Σ y después calculando la traspuesta de la matriz resultante.

Si A es cuadrada y no singular (luego invertible) se recupera la matriz inversa: $A^{-1} = A^\dagger$. Para las matrices que no son de rango máximo, el paso al límite es necesario para el cálculo de la pseudo-inversa. Para las matrices de rango máximo se tiene:

Teorema 5. *Sea $A \in \mathbb{R}^{m,n}$ una matriz de rango máximo, es decir, $rank(A) = \text{mín}\{m, n\}$. Si $rank(A) = n$ podemos calcular la Pseudo-Inversa mediante la fórmula*

$$A^\dagger = (A^T A)^{-1} A^T$$

Si $rank(A) = m$ podemos calcular la Pseudo-Inversa mediante la fórmula

$$A^\dagger = A^T (A A^T)^{-1}$$

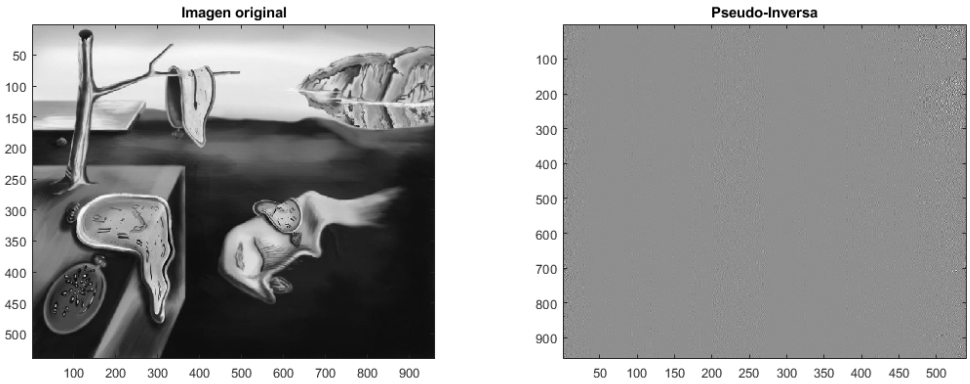


Figura 4.6: Representación del cálculo de la Pseudo-Inversa de una imagen del cuadro La persistencia del tiempo de Dalí.

Si $A \in \mathbb{R}^{m,n}$ no es una matriz de rango máximo, es decir, $rank(A) < \text{mín}\{m, n\}$, entonces hay que usar la fórmula (4.5) de la definición. En todo los casos se puede

utilizar el algoritmo (4.6) basado en la SVD y que está implementado en el comando **pinv.m** de Matlab. El interés de la definición radica en que introduce claramente la técnica de regularización de problemas mal planteados.

4.3.3. Análisis de Componentes Principales (PCA)

La técnica de análisis de componentes principales es debida a Hotelling (1933), [19] y se basa en resultados previos sobre problemas de mínimos cuadrados y regresiones lineales de K. Pearson (1901), [20]. Aplicaciones recientes en el campo de la Inteligencia Artificial y las redes neuronales son, entre otras, el reconocimiento automático de caras de Gan et al., (2015), [21] y el análisis de datos multivariables de alta dimensión en la red NetPCA de Codesido et al. (2021), [22]. Estos desarrollos recientes demuestran la efectividad y generalidad de la técnica en la reducción dimensional de problemas de alta dimensión.

El Análisis de Componentes Principales (PCA, del inglés *principal component analysis*) es una de las herramientas más importantes del álgebra lineal aplicada. Este método estadístico se utiliza en aprendizaje automático no supervisado para reducir la dimensionalidad durante el análisis de grandes cantidades de datos a la vez que conserva su información. Permite entender las tendencias, patrones y relaciones entre los datos que están oscurecidos o que son confusos en su representación original.

Este algoritmo puede utilizarse para reducir significativamente el tiempo de ejecución de los algoritmos de aprendizaje de características no supervisados (*unsupervised feature learning algorithm*).

Para poner a los alumnos en contexto y hacerles entender de manera sencilla en qué consiste el análisis de componentes principales utilizaremos el ejemplo del problema de física del movimiento de un muelle ideal que aparece en el Tutorial sobre el Análisis en Componentes Principales [2].

El sistema de este problema consiste en una bola de masa m unida a un muelle sin masa ni rozamiento. La bola se suelta a una pequeña distancia del equilibrio, es decir, el muelle se estira. Como el muelle es "ideal" oscila indefinidamente a lo largo del *eje* x alrededor de su equilibrio con una frecuencia determinada. Este es un problema estándar en física en el que el movimiento a lo largo de la dirección x se resuelve mediante una función explícita del tiempo. En otras palabras, la dinámica subyacente puede expresarse en función de una única variable x.

Suponiendo que no sabemos qué ejes y dimensiones son importantes medir, decidimos medir la posición de la bola en un espacio tridimensional. Para ello, colocamos tres cámaras alrededor del sistema. A 200 Hz, cada cámara graba una imagen que indica la posición bidimensional de la bola (su proyección). Como tampoco sabemos cuáles son los ejes reales "x", "y" y "z", elegimos tres ejes de cámara $\rightarrow a$, $\rightarrow b$, $\rightarrow c$ en ángulos arbitrarios respecto al sistema, que puede que ni siquiera sean de 90^{o}. Después de grabar durante varios minutos con las cámaras, nos preguntamos cómo pasar de este conjunto de datos a una ecuación simple de x.

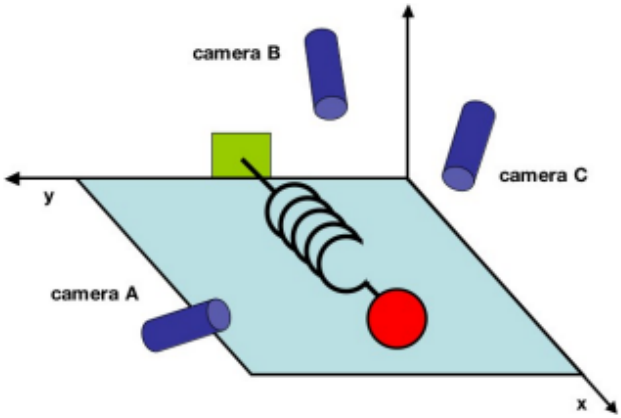


Figura 4.7: Representación del ejemplo del problema de física del movimiento de un muelle ideal. *Fuente: A tutorial on principal component analysis, [2].*

Podríamos habernos limitado a medir la posición a lo largo del eje x con una sola cámara. Pero esto no es lo que pasa en el mundo real, ya que muchas veces no sabemos qué mediciones reflejan mejor la dinámica del sistema. Además, hay que tener en cuenta el problema del ruido. En nuestro ejemplo habría que tener en cuenta el aire, la imprefección de las cámaras o el rozamiento en un muelle no ideal. El ruido contamina nuestros datos perturbando aún más la dinámica.

El objetivo del PCA es calcular la base más significativa que permita reexpresar mejor un conjunto de datos ruidosos.

En el ejemplo anterior, cada muestra temporal será una muetra individual del conjunto de datos. La cámara A registrará una posición correspondiente en un momento dado de la bola (x_A, y_A). Por lo que la muestra en este ejemplo se puede expresar como un vector columna de 6 dimensiones donde cada cámara contribuye con una proyección bidimensional de la bola:

$$\overrightarrow{X} = \begin{pmatrix} x_A \\ y_A \\ x_B \\ y_B \\ z_A \\ z_B \end{pmatrix}$$

Grabando la posición de la pelota durante 10 minutos a 120 Hz, registraríamos $10 \times 60 \times 120 = 72000$ de esos vectores.

En términos abstractos, cada muestra de X es un vector m-dimensional generado por alguna base ortonormal. De álgebra lineal sabemos que todos los vectores de medida forman combinaciones lineales de este conjunto de vectores de base de longitud unitaria. Por ello, el PCA parte del supuesto de linealidad y se limita a reexpresar los datos como una combinación lineal de sus vectores base de longitud unitaria.

Para saber cuál es esta base ortonormal, supongamos en el ejemplo del muelle que

hemos reunido los datos sólo mirando la cámara A. Una elección ingenua de la base sería $\{(1,0),(0,1)\}$, pero podríamos elegir otra como $\{(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}),(\frac{-\sqrt{2}}{2},\frac{-\sqrt{2}}{2})\}$ o cualquier otra rotación. La razón de la elección de la primera base es el método con el que se recopilan los datos, pues para registrar la posición $(2,2)$ en nuestras cámaras, significa 2 unidades hacia arriba y 2 unidades hacia la izquierda en la ventana de la cámara.

Generalizando al caso m-dimensional podemos construir una matriz identidad $m \times m$

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdot & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdot & 1 \end{pmatrix} = I$$

donde cada fila es un vector básico ortonormal b_i con m componentes. Este sería el punto de partida ya que todos los datos han sido registrados en este base y, por tanto, pueden expreserse de forma trivial como combinación lineal de $\{b_i\}$.

Ahora, para describir la base matemática del proceso utilizamos el Tutorial de la Universidad de Stanford [23] y el Tutorial sobre el Análisis en Componentes Principales de Jonathon Shlens [2].

Considerando el conjunto de datos $X = \{x_1, ..., x_m\}$ como una matriz $m \times n$, donde cada muestra (dato) x_i tiene n componentes describiendo así n características, es decir, $x^{(j)} \in \mathbb{R}^n$. Sea Y otra matriz $m \times n$ relacionada por una transformación lineal P. Y es una representación del conjunto de datos original de la forma

$$PX = Y$$

que representa un cambio de base. P es una matriz que tranforma X en Y y, geométricamente, se trata de una rotación y un estiramiento que transforma X en Y. Las filas de P, $\{p_1, ..., p_m\}$ son un conjunto de nuevos vectores base para expresar las columnas de X. Es decir,

$$PX = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} \begin{pmatrix} x_1 \cdots x_n \end{pmatrix}$$

$$Y = \begin{pmatrix} p_1 \cdot x_1 & \cdots & p_1 \cdot x_n \\ \vdots & \ddots & \vdots \\ p_m \cdot x_1 & \cdots & p_m \cdot x_n \end{pmatrix}$$

Cada coeficiente y_i es un producto punto de x_i con la fila correspondiente en P. Es decir, $y_j^{(j)}$ es una proyección sobre la j-ésima fila de P. Los vectores fila $\{p_1, ..., p_m\}$, que son ortonormales, serán las componentes principales en X.

Para preparar los datos de manera que la PCA funcione calculamos la media por filas

$$\mu_i = \frac{1}{m} \sum_{i=1}^{n} x_i^{(j)}$$

para cada ejemplo del dataset:

$$x^{(j)} = (x_1^{(j)}, ..., x_i^{(j)}, ...x_m^{(j)})' \in \mathbb{R}^m$$

y se la restamos al dataset para normalizarlo

$$x_i^{(j)} = x_i^{(j)} - \mu_i$$

y obtener un nuevo dataset de media cero, donde las distintas características tengan variaciones similares entre sí.

Calculamos la matriz de Covarianza del dataset mediante la fórmula

$$C_X = \frac{1}{n-1} XX^T$$

La forma matricial XX^T, en ese orden, compone el valor deseado para el elemento (i, j)-ésimo de C_X. Concretamente, el elemento (i, j)-ésimo de C_X es el producto punto entre el i-ésimo vector del tipo de muestra con el j-ésimo vector del tipo de muestra.

La matriz de covarianza C_X es una matriz cuadrada, simétrica y semi-definida positiva $m \times m$. Los términos diagonales de C_X son la varianza de las variables particulares del dataset y los términos no diagonales, su covarianza. C_X recoge los valores de correlación entre todos los pares de muestras posibles, reflejando el ruido y la redundancia de los datos.

Los objetivos son minimizar la redundancia, medida por la covarianza, y maximizar la señal, medida por la varianza. Por definición, las covarianzas son no negativas, por lo que la covarianza mínima es cero. Si llamamos C_Y al resultado de manipular la matriz C_X, lo que buscaremos será que C_Y sea diagonal.

Escribiendo C_Y en términos de la variable de elección P tenemos

$$\begin{aligned} C_Y &= \frac{1}{n-1} YY^T \\ &= \frac{1}{n-1} (PX)(PX)^T \\ &= \frac{1}{n-1} PXX^T P^T \\ &= \frac{1}{n-1} P(XX^T)P^T \\ C_Y &= \frac{1}{n-1} PAP^T \end{aligned}$$

Hemos definido una nueva matriz $A = XX^T$, donde A es simétrica. Por la teoría de Diagonalizabilidad y Descomposición Espectral (ver anexos A.1.12 y A.1.14) sabemos que por ser simétrica puede escribirse como

$$A = EDE^T \tag{4.7}$$

donde D es una matriz diagonal formada por los autovalores $\{\lambda_1, ..., \lambda_m\}$ de A y E es

una matriz ortogonal de los vectores propios de A dispuestos en columnas.

Seleccionamos la matriz P de forma que cada fila p_i sea un vector propio de XX^T. Por esta selección, $P \equiv E^T$. Sustituyendo en la ecuación 4.7 se tiene $A = P^T DP$. Con esta relación y sabiendo que $P^{-1} = P^T$, se tiene

$$\begin{aligned} C_Y &= \frac{1}{n-1} PAP^T \\ &= \frac{1}{n-1} P(P^T DP)P^T \\ &= \frac{1}{n-1} (PP^T)D(PP^T) \\ &= \frac{1}{n-1} (PP^{-1})D(PP^{-1}) \\ C_Y &= \frac{1}{n-1} D \end{aligned}$$

Es evidente que la elección de P diagonaliza C_Y, que era el objetivo del PCA. Resumiendo los resultados del PCA en las matrices P y C_Y:

- Los componentes principales de X son los autovectores de XX^T (o filas de P).
- El i-ésimo elemento diagonal de C_Y es la varianza de X a lo largo de p_i.

En la práctica, para calcular el PCA de un conjunto de datos X simplemente hay que calcular los vectores propios de la matriz de covarianza C_X.

Es habitual ordenar los autovectores en función del autovalor asociado [24]. Es decir, siendo $\{\lambda_1, ..., \lambda_m\}$ los autovalores de C_X y $\{p_1, ..., p_m\}$ sus autovectores. El espectro es no negativo, $\lambda_i \geq 0$ y los autovalores están ordenados de mayor a menor:

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$$

El primer autovalor, λ_1, es el autovalor principal y corresponde a p_1, el autovector principal.

Otra forma de calcular la PCA es aplicando la descomposición en valores singulares debido a la estrecha relación que existe entre ambos métodos. Digamos que la SVD proporciona el marco teórico para el cálculo y aplicación de la PCA en análisis estadístico de datos. Lo veremos a continuación.

SVD y PCA

Definimos una nueva matriz Y de $n \times m$ en la forma

$$Y = \frac{1}{\sqrt{n-1}} X^T$$

donde cada columna de Y tiene media cero. La definición de Y se justifica analizando Y^TY.

$$\begin{aligned} Y^TY &= (\frac{1}{\sqrt{n-1}}X^T)^T(\frac{1}{\sqrt{n-1}}X^T) \\ &= \frac{1}{n-1}X^{TT}X^T \\ &= \frac{1}{n-1}XX^T \\ Y^TY &= C_X \end{aligned}$$

Por construcción Y^TY es igual a la matriz de covarianza de X. Como sabemos que los componentes principales de X son los autovectores de C_X, si calculamos la SVD de Y, las columnas de la matriz V contienen los autovectores de $Y^TY = C_X$. Por tanto, las columnas de V son los componentes principales de X.

V genera el espacio de filas de $Y \equiv \frac{1}{\sqrt{n-1}}X^T$. Por tanto, V debe generar el espacio de columnas $\frac{1}{\sqrt{n-1}}X$. Podemos concluir que encontrar los componentes principales equivale a encontrar una base ortonormal que genere el espacio de columnas de X.

Si el objetivo final es encontrar una base ortonormal para el espacio generado por las columnas de X, entonces podemos calcularla directamente sin construir Y. Por simetría, las columnas de U producidas por las SVD de $\frac{1}{\sqrt{n-1}}X$ también deben ser los componentes principales.

Otra forma de apreciar la similitud entre el PCA y la SVD es a partir de la ecuación 4.7. Supongamos que $\{\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m\}$, los componentes principales, se refieren a los primeros r vectores columna de E, y los asociados $\{\lambda_1, ..., \lambda_r\}$ son varianzas o potencias de los componentes principales.

Por similitud, pero sin pérdida de generalidad, podemos suponer que X tiene media cero a lo largo de todas las filas. Utilizando la descomposición SVD de X y la ortonormalidad de V, es fácil demostrar que E se puede calcular a partir de la SVD de X:

$$XX^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma V^TV\Sigma U^T = U\Sigma^2U^T$$

Es decir, $E = U$ y $D = \Sigma^2$ o $\lambda_i = \sigma_i^2$.

Como puede observarse, hay una estrecha relación entre la SVD y la PCA puesto que el análisis de componentes principales de una matriz de datos $X \in \mathbb{R}^{n,m}$ puede reducirse a calcular la SVD de X [1].

Los autovectores (ordenados por columnas) de C_X construyen la matriz ortogonal U de la Factorización SVD de X. Los autovectores constituyen una nueva base en la que representar los datos.

El resultado del PCA, fijado un k, es simplemente considerar los k autovectores superiores de la matriz de covarianza XX^T, es decir, retener las primeras k componentes principales, pues son las que tienen asociada una varianza mayor y, por tanto, quienes

contienen una información más interesante. Precisamente la SVD de X entrega esos autovectores, que son las primeras k filas de V^T [1].

4.4. Seminario Parte II: Aplicaciones SVD

En esta segunda parte del seminario se les mostrará a los alumnos algunas de las importantes aplicaciones de la Descomposición en Valores Singulares. Entre ellas, la Compresión de imágenes digitales (4.4.1), el Reconocimiento facial (Eigenfaces) (4.4.2) y la Eliminación de ruido (Denoising) (4.4.3).

De nuevo, esta segunda parte del seminario seguirá las características y objetivos descritos en la sección 4.1.1 y se llevará a cabo en otra sesión de duración de dos horas.

4.4.1. Compresión de imágenes digitales

La compresión de imágenes digitales es una de las técnicas más importantes hoy en día, ya que permite resolver uno de los mayores problemas actuales: reducir el espacio de almacenamiento ocupado por una imagen sin que esta pierda calidad [25].

Mediante esta técnica pueden almacenarse y transmitirse los datos de manera eficiente así como ahorrar memoria, ancho de banda y costes. Básicamente hay una redundancia en la codificación de los datos, en los valores entre píxeles y también visual.

La compresión de imágenes se clasifica en dos tipos: compresión de imágenes con pérdidas (*lossy compression*) y compresión de imágenes sin pérdidas (*lossless compression*). En este caso veremos la compresión con pérdidas, ya que son las técnicas mayormente utilizadas para la compresión de imágenes, siendo la SVD una de ellas. El énfasis de esta técnica está en la observación de que en una imagen hay información redundante que puede no ser almacenada para permitir una tasa de transmisión de datos más elevada.

Se le introduce a los alumnos, a continuación, los comandos de Matlab necesarios para leer y calcular la SVD de una imagen, que después será comprimida.

Tras leer una imagen con el comando **imread** y almacenarla en una variable, $A \in \mathbb{R}^{m,n}$, pasamos los datos enteros a datos de coma flotante (típico en Matlab) mediante el comando A = **im2double**(A). Realizamos la descomposición SVD mediante el comando [U,S,V] = **svd**(A). Fijado el número de valores singulares, digamos r, se construye una versión comprimida X de la imagen original A de la forma:

$$A_{red} = U(:, 1:r) * S(1:r, 1:r) * V(:, 1:r)^T$$

Sabemos que el rango de una matriz singular es igual al número de valores singulares distintos de cero. Luego, el rango de la matriz reducida que se ha construido es exactamente r, el número de valores singulares retenidos. Una cuantificación de la

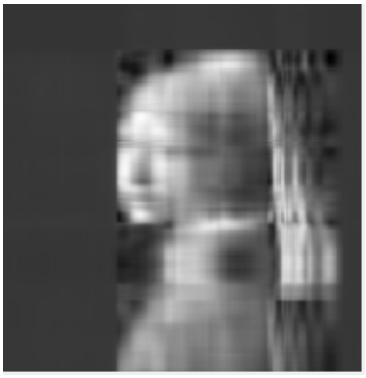
compresión obtenida se puede dar definiendo el **ratio de compresión** c_r, que nos da el porcentaje de elementos retenidos en la versión reducida de la imagen:

$$c_r = \frac{r(m+n+1)}{mn}$$

Se les mostrará a los alumnos a través del siguiente ejemplo cómo comprimir una imagen para diferentes valores de r utilizando la descomposición SVD.

Ejemplo 2. *Sea dada una imagen. Se pide calcular una descomposición SVD de la imagen y una versión reducida de la misma utilizando r = 5, 20 y 100 modos de la descomposición. Representar las imágenes valorando visualmente la calidad de la reducción dimensional.*

(a) Imagen original



(b) Imagen reducida de rango r = 5 y 1.81 %

(c) Imagen reducida de rango r = 20 y 7.23 %

(d) Imagen reducida de rango r = 100 y 36.16 %

Figura 4.8: Compresión de la imagen del cuadro La joven de la perla de Johannes Vermeer mediante una descomposición SVD para diferentes valores del rango. Se detalla el porcentaje de datos que se almacena en cada imagen para cada valor.

En este ejemplo observamos cómo es la reconstrucción de la imagen al pasar por diferentes valores del rango, mejorando la aproximación al aumentar dichos valores.

El código empleado para leer la imagen original, calcular su descomposición SVD, aproximar la imagen para los distintos valores del rango y representar tanto la imagen original como las aproximaciones es el siguiente:

Código 4.2: SVD para compresión de una imagen en escala de grises

```
A = imread('.\imagenes\jovendelaperla.jpg');
X = im2double(rgb2gray(A));
nx = size(X,1); ny = size(X,2);
%% imagen original
figure,
subplot(2,2,1)
imagesc(X),  axis off
colormap gray
title('Original');
%% SVD Approximations
[U,S,V] = svd(X,'econ'); % econ remove the zero
   singular values
plotind = 2;
for r = [5 20 100]; % truncation value
    Xapprox = U(:,1:r)*S(1:r,1:r)*V(:,1:r)';
    subplot(2,2,plotind), plotind = plotind + 1;
    imagesc(Xapprox), axis off, colormap gray;
    title(['r=',num2str(r,'%d'),', ',num2str(100*r*(nx+
       ny)/(nx*ny),'%2.2f'),
    'storage'])
end
```

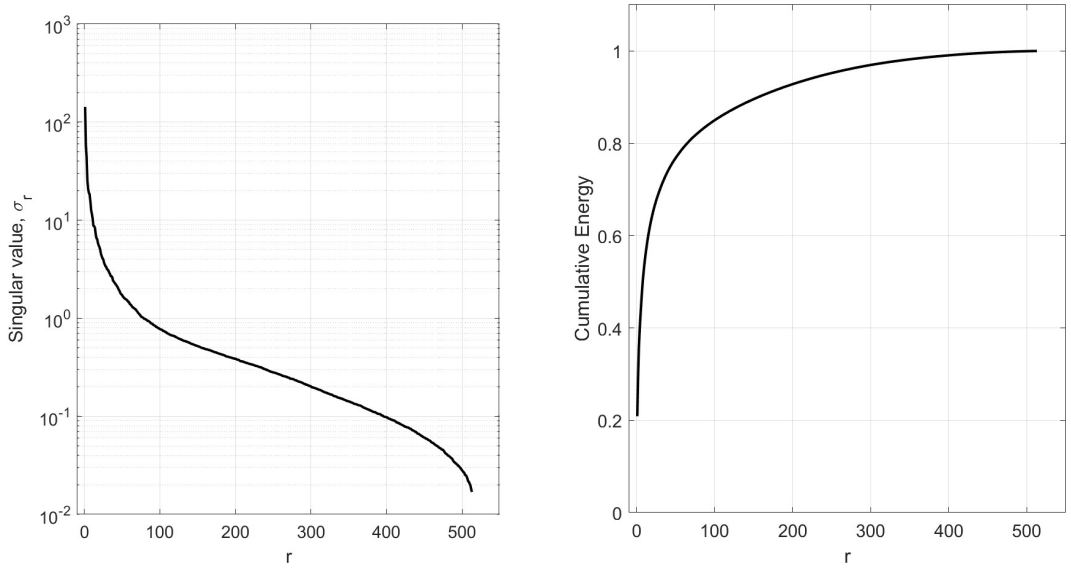
Observando la figura 4.8, en el caso $r = 5$ podemos decir que hay mucha pérdida de información en la compresión, pero realmente ya es posible percibir las características básicas de la imagen y hacerse una idea de lo que está representando con sólo 5 modos [18].

A medida que se aumentan los valores de r a 20 y 100 se obtienen características más sutiles y detalladas de la imagen. En la imagen de rango $r = 20$ se puede ver claramente lo que representa la imagen, aunque con baja resolución. Sin embargo, con 100 modos la representación de la imagen se aproxima muy fielmente a la imagen original [18].

Por otro lado, se recoge de manera explícita cuántos datos está almacenando cada una de las imágenes para los diferentes valores de r. Podemos observar que únicamente el 36.16 % de los datos de la imagen A son los que deben almacenarse para mantener las 100 primeras columnas de U, la primera submatriz de 100×100 de Σ y las primeras 100 columnas de V.

Utilizando el comando **diag(S)** se trazan los elementos de la diagonal de Σ para representarlos en un gráfico tomando el logaritmo de esos valores singulares. Además, vamos a representar la suma acumulativa, es decir, la suma de los primeros r valores

singulares dividida entre la suma de todos los valores singulares, utilizando el comando **cumsum(diag(S))/sum(diag(S))**, para ver cuánta energía o información hay en los primeros modos en comparación con todos los modos [18].



(a) Gráfica de la escala logarítmica de los valores singulares de la imagen.

(b) Gráfica de la energía acumulada al aumentar los modos.

Figura 4.9: Gráficas de la escala logarítmica de los valores singulares de la imagen y de la energía acumulada al aumentar los modos.

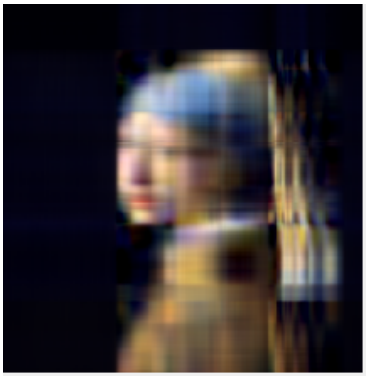
En la gráfica 4.9a podemos observar una caída muy pronunciada en la magnitud de los valores singulares que después se estabiliza. El primer grupo de valores singulares es el que mantiene la mayor parte de la información en la matriz de datos, lo que refleja que gran parte de los valores singulares de baja energía puedan desecharse, manteniendo la información de la matriz original. Esta gráfica da la información sobre cuántos modos se necesitan para mantener la estructura de los datos [18].

La gráfica 4.9b refleja cuánta energía acumulada se mantiene según el número de modos utilizados. Se observa que tan sólo con el primer modo se obtiene aproximadamente el 40 % de la información en esa matriz de datos [18].

A partir del análisis anterior se observa que a medida que r aumenta, la calidad de la imagen mejora, pero al mismo tiempo también aumenta la cantidad de memoria necesaria para almacenar la imagen. Por ello, el valor óptimo de r debe seleccionarse de forma que no se degrade la calidad de la imagen y, al mismo tiempo, se reduzca el espacio de almacenamiento ocupado por la imagen. Por lo tanto, la selección del valor r desempeña un papel crucial en el rendimiento de la técnica de Descomposición de Valores Singulares [25].

Se les mostrará a los alumnos, a continuación, cómo realizar la compresión de una imagen digital en color, para lo que se utilizará el ejemplo anterior con la imagen del cuadro de La joven de la perla en color.

(a) Imagen original en color



(b) Imagen reducida de rango r = 5 y 1.81 %

(c) Imagen reducida de rango r = 20 y 7.23 %

(d) Imagen reducida de rango r = 100 y 36.16 %

Figura 4.10: Compresión de una imagen del cuadro La joven de la perla de Johannes Vermeer en color mediante una descomposición SVD para diferentes valores del rango. Se detalla el porcentaje de datos que se almacena en cada imagen para cada valor.

Para realizar la descomposición SVD sobre una imagen en color separamos los canales RGB para después realizar la descomposición en cada uno de ellos. Una vez leída la imagen y almacenada en una variable A, podemos obtener los canales RGB de la imagen con un bucle que almacene cada canal en una variable. Fijando el número de valores singulares r para realizar la aproximación comprimida de la imagen, aplicamos la descomposición SVD sobre cada canal y se reconstruye la nueva imagen comprimida, almacenándola en una nueva variable X. El código empleado para ello es el siguiente:

Código 4.3: SVD para compresión de una imagen en color

```
%% Separacion de canales
channels = cell(1,3);
for i = 1:3
    channels{i} = A(:,:,i);
end
%% SVD
```

```
nx = size(A,1); ny = size(A,2);
X = zeros(nx,ny,3);
for i = 1:3
    [U, S, V] = svd(channels{i});
    X(:,:,i)=U(:,1:r)*S(1:r,1:r)*V(:,1:r)';
end
```

Tras esta primera aplicación de la SVD se les mostrará a los alumnos el código necesario para realizar una animación que se recoge en un vídeo (por ejemplo, una combinación lineal convexa de imágenes) y ejemplificar, de esta forma, el proceso matemático. Esto les permitirá entender mejor la evolución del proceso de la SVD al percibirlo de manera visual sobre una imagen digital.

Al final del seminario se le propondrá a los estudiantes una práctica en la que tendrán que cambiar las imágenes y generar otros vídeos para las aplicaciones propuestas. En ella, como se ha mencionado en la sección 4.1.3 se valorará la creatividad, el desarrollo, la originalidad y la calidad del vídeo generado.

El código empleado para esta aplicación de la SVD se muestra a continuación:

Código 4.4: Animación SVD

```
A=imread('./imagenes/dali.jpg');
A=rgb2gray(A);
A=im2double(A);
[U,S,V]=svd(A);
%%  animacion
figure
colormap gray
subplot(1,3,1);
imagesc(A)
axis off
title('original')
r=rank(A);
videoFile = VideoWriter('videoSVD.avi');
videoFile.FrameRate = 10; % Velocidad de fotogramas (
   frames per second)
open(videoFile);
for t=1:1:r
    U_red=U(:,1:t);
    S_red=S(1:t,1:t);
    V_red=V(:,1:t)';
    X_red = U_red*S_red*V_red;
    subplot(132);
    imagesc(X_red)
    axis off
    title(['t=',num2str(t)]);
```

```
25      F=getframe(gcf);
26      writeVideo(videoFile,F);
27      drawnow;
28      pause(0.1)
29      B=A-X_red;
30      subplot(133)
31      imagesc(B,[0,1])
32      axis off
33      title('residuo')
34  end
35  close(videoFile);
```

4.4.2. Reconocimiento facial (Eigenfaces)

Una de las aplicaciones más llamativas de la SVD es el denominado *Eigenfaces Example*. Este problema consiste en aplicar PCA basado en la SVD a una gran biblioteca de imágenes faciales para extraer las correlaciones más dominantes entre las imágenes. Esta descomposición da como resultado un conjunto de caras propias (*eigenfaces*) que definen un nuevo sistema de coordenadas. Las imágenes pueden representarse en estas coordenadas tomando el producto punto con cada uno de los componentes principales [18].

El término *eigenface* fue introducido y estudiado por primera vez por Sirovich y Kirby en 1987 [26] y, más tarde, fue ampliado [27]. Su aplicación al reconocimiento facial automatizado fue presentada por Turk y Pentland en 1991 [28]. Eigenface es el concepto fundamental utilizado para el reconocimiento facial. Varios algoritmos de preprocesamiento, como la corrección gamma para el preprocesamiento de la iluminación, se basaron en él para mejorar la precisión [29].

Para describir el Problema de Eigenfaces se utiliza el Tutorial de la Universidad de Washington de los autores Steven L. Brunton y J. Nathan Kutz [18].

En él se utiliza la base de datos de rostros ampliada de Yale B [30], que consiste en imágenes de 38 individuos (28 de la base de datos ampliada y 10 de la base de datos original) bajo 9 poses y 64 condiciones de iluminación (ver figura 4.11b). Esas imágenes han sido recortadas y alineadas de manera que las cejas, ojos, narices y bocas estén en una posición similar en las imágenes del conjunto de datos y que cada una de ellas tenga 192 píxeles de alto y 168 píxeles de ancho. Cada una de las imágenes faciales ha sido remodelada en un gran vector columna con $192 \times 168 = 32256$ elementos para construir una gran matriz de caras. Se utilizan las 36 primeras personas de la base de datos como datos de entrenamiento para el ejemplo de eigenfaces y se retienen a dos personas como conjunto de prueba.

Una vez descargados los datos de la base de datos y cargados en Matlab a través del comando **load allFaces.mat**, tenemos una matriz de gran tamaño llamada *faces* formada por 2410 imágenes individuales de rostros humanos dispuestas en los vectores

(a) Representación en una única imagen de las 36 primeras personas.

(b) Ejemplo de las 64 imágenes de una persona específica.

Figura 4.11: Representación de algunas de las caras de la base de datos de rostros ampliada de Yale B.

columna de 32256 elementos (es decir, $faces \in \mathbb{R}^{32256,2410}$). Hay un bloque de la matriz para la persona uno, otro para la persona dos y así sucesivamente hasta completar las 38 caras que contiene la matriz.

Vamos a crear una matriz de todas las personas que queremos utilizar en el modelo, ya que sólo vamos a entrenar la SVD para las primeras 36 caras. Veremos si esas eigenfaces, que se almacenarán el matriz U de la SVD, se pueden utilizar para representar la cara de otra persona que no estaba en el conjunto de datos inicial.

El código para crear y representar la matriz de los 36 rostros es el siguiente:

Código 4.5: Representar la matriz de los 36 rostros

```
allPersons = zeros(n*6,m*6);
count = 1;
for i=1:6 % 6 x 6 grid of faces
    for j=1:6
        allPersons(1+(i-1)*n:i*n,1+(j-1)*m:j*m) ...
            = reshape(faces(:,1+sum(nfaces(1:count-1)))
               ,n,m);
        count = count + 1;
    end
end
figure(1), axes('position', [0 0 1 1]), axis off
imagesc(allPersons),
colormap gray
```

Para preparar los datos para calcular la SVD se calcula la cara promedio. Tomamos el promedio de todas las columnas de la matriz de las 36 caras y lo restaremos a cada vector columna. Los vectores de la imagen con la media restada se apilan horizontalmente como columnas en la matriz de datos X. Calculamos entonces la SVD (de tamaño económico) de la matriz con la media restada X, obteniendo como resultado el PCA. Las columnas de la matriz U son las eigenfaces y estas pueden remodelarse para ser nuevamente imágenes 192×168, es decir, para que tengan de nuevo la forma de una cara.

Código 4.6: Calcular las eigenfaces de los datos de entrenamiento con la media restada.

```
% Utilizamos las primeras 36 personas para los datos de
    entrenamiento
trainingFaces = faces(:,1:sum(nfaces(1:36)));
avgFace = mean(trainingFaces,2); % size n*m by 1;

% Calculo de eigenfaces sobre los datos de
   entrenamiento con la media restada
X = trainingFaces-avgFace*ones(1,size(trainingFaces,2))
   ;
[U,S,V] = svd(X,'econ');
```

En la figura 4.12 se muestran las primeras 64 caras propias, que son las primeras 64 columnas de la matriz U de la descomposición SVD. Para mostrar las eigenfaces, cada una de esas columnas de U se ha remodelado en una matriz que se parece a la forma de una cara. Es decir, esos vectores de 32256 elementos se remodelan en una imagen y cada una de esas imagenes se trazan a lo largo de las filas de una matriz. Lo que se ve en la figura son las 64 primeras caras propias, pero hay tantas caras propias como columnas de datos (2410 eigenfaces).

Código 4.7: Representar primeras 36 eigenfaces

```
EigenFaces = zeros(n+8,m+8);
count = 1;
for i=1:8
    for j=1:8
        EigenFaces(1+(i-1)*n:i*n,1+(j-1)*m:j*m) ...
            = reshape(U(:,count),n,m);
        count = count + 1;
    end
end
figure(1), axes('position',[0 0 1 1]), axis off
imagesc(EigenFaces),
colormap gray;
```

En la imagen se perciben estructuras muy interesantes. Las dos primeras eigenfaces contienen aquello que todos los rostros tienen en común: dos ojos, nariz y boca.

En las siguientes caras propias se empiezan a observar características diferentes en la iluminación y en la estructura, que aportan aquellos rasgos que diferencian los rostros humanos unos de otros. Por tanto, cada vector columna en X es solo una combinación lineal de estas caras propias.

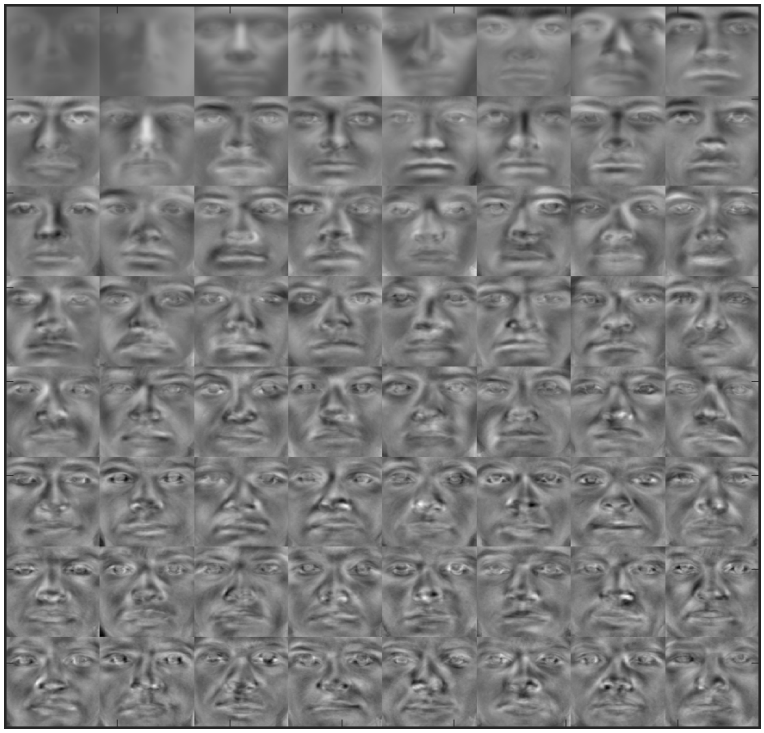


Figura 4.12: Representación de las primeras 64 eigenfaces resultantes al aplicar la descomposición SVD.

A continuación, veremos que si desechamos muchas de estas eigenfaces, incluso si solo conservamos los primeros 100 en lugar de los 2410, podemos hacer una buena aproximación de los rostros de nuestras personas individuales únicamente a partir de una combinación lineal de algunas de estas caras propias. Esa información de la combinación lineal está en ΣV^T, ya que cada columna de esta parte de la descomposición SVD va a ser la mezcla de las caras propias y va a dar como resultado la persona uno, la persona dos y así sucesivamente.

La cara promedio compuesta por cada una de las imagenes de la base de datos se muestra a continuación (ver figura 4.13). Podemos observar que prácticamente las sombras se han eliminado al haberse promediado. Los comandos para cacular y representar la cara promedio son:

Código 4.8: Representar cara promedio.

```
%% Representar cara promedio
figure, axes('position',[0 0 1 1]), axis off
imagesc(reshape(avgFace,n,m)),
colormap gray;
```

Figura 4.13: Representación de la cara promedio compuesta por cada una de las imagenes de la base de datos.

Trazamos los valores singulares en una escala logarítmica utilizando el comando **semilogy(diag(S))**. En la gráfica 4.14, el eje x representa el número de modos que podrían mantenerse (es decir, el rango de truncamiento), cuyo valor máximo es 2410 por ser el número de columnas de X, y el eje y representa la magnitud del logaritmo del valor singular en la diagonal de Σ.

Observamos en la gráfica un comportamiento interesante, ya que se ralentiza lentamente y, después, cae de golpe a una precisión de 10^{-16}. Esto se debe a que si mantuviésemos absolutamente todas las eigenfaces obtendríamos una aproximación de precisión de máquina de la matriz de datos X con un error 0 o lo más cercano que un ordenador puede representar. Podemos ver también lo esperable en una distribución de valores singulares y es que los primeros valores singulares son los que llevan la mayor parte de la energía, en este caso, la mayor varianza en las caras.

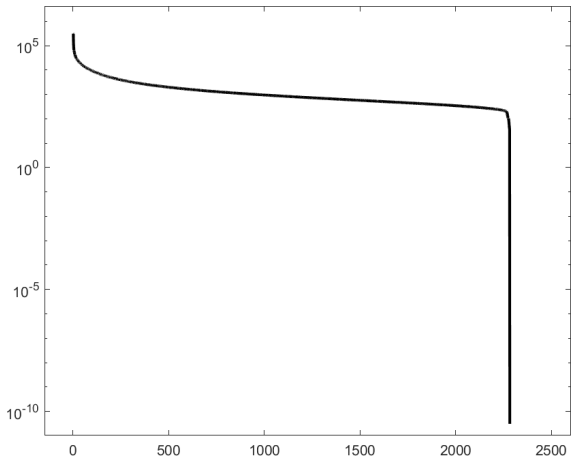


Figura 4.14: Gráfica de la escala logarítmica de los valores singulares de la matriz de rostros X.

Ahora vamos a ver cómo podemos usar la biblioteca de eigenfaces U para aproximar un nuevo rostro humano que no estaba en los datos de entrenamiento. Para ello, vamos a tomar la cara de esta nueva persona 37, que será un vector columna x de 32256

dimensiones, y vamos a proyectarlo en el espacio de las eigenfaces U, tomando las primeras r caras propias. La proyección es la siguiente:

$$\tilde{x}_{test} = \tilde{U_r}\tilde{U_r}^T x_{test}$$

Se obtienen, con el producto $\tilde{U_r}^T x_{test}$, los coeficientes de la persona X en el sistema de coordenadas de U, es decir, la mezcla de las primeras caras propias de la nueva persona. Y tomando una combinación lineal de U_r se reconstruye una aproximación de X en ese espacio.

Veremos cómo se reconstruye la cara de esta nueva persona si mantenemos diferentes valores de eigenfaces ($r = 25, 50, 100, 200, 400, 800$ y 1600) y comprobaremos cómo de precisa es esta reconstrucción.

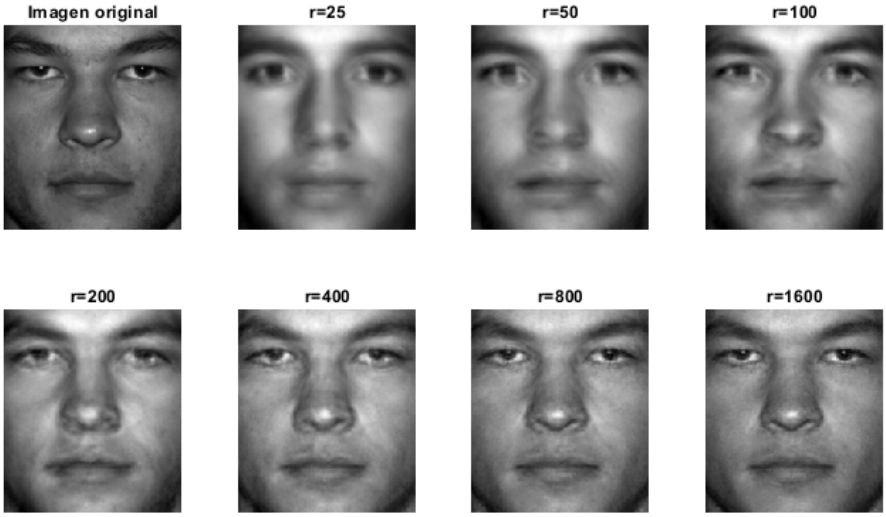


Figura 4.15: Representación de la aproximación de un rostro fuera de los datos de entrenamiento tomando eigenfaces de diferentes valores r.

En la figura 4.15 podemos ver que rápidamente las aproximaciones convergen a la cara original. Es importante recalcar que no utilizamos las imágenes de esta persona en los datos de entrenamiento, sino que estos fueron entrenados con las primeras 36 personas. A partir del rango $r = 200$ observamos que los rasgos y estrucutras empiezan a tener un gran parecido y aumentando a 400 u 800 ya podría decirse realmente quién es esta persona. De 32256 píxeles, sólo necesitamos los primeros 400 bits de información para poder distinguir la huella digital única del rostro de esta persona dentro de la biblioteca de datos.

El código de Matlab para representar la aproximación del rostro bajo diferentes valores de r es el siguiente:

Código 4.9: Reconstrucir las eigenfaces de la imagen fuera de los datos de entrenamiento

```
%% Reconstruccion de las eigenfaces de la imagen fuera
   de los datos de entrenamiento
testFace = faces(:,1+sum(nfaces(1:36))); % primera cara
    de la persona 37

subplot(2,4,1)
imagesc(reshape(testFace,n,m)), colormap gray, axis off
title('Imagen original');
count = 1;
testFaceMS = testFace - avgFace;
for r=[25 50 100 200 400 800 1600]
    count = count + 1;
    subplot(2,4,count)
    reconFace = avgFace + (U(:,1:r)*(U(:,1:r)'*
       testFaceMS));
    imagesc(reshape(reconFace,n,m)), colormap gray,
       axis off
    title(['r=',num2str(r,'%d')]);
end
```

Es muy interesante observar que el espacio ortogonal de caras propias no sólo es útil para representar rostros humanos, sino que también puede utilizarse para aproximar imágenes muy diferentes. Las columnas de la matriz U son ortonormales y si mantenemos suficientes de ellas abracaríamos un subespacio vectorial de 2410 dimensiones del espacio de imagen de 32256 dimensiones correspondiente a rasgos espaciales amplios, suaves y no localizados, como las mejillas, la frente, la boca, etc.

Para ello, vamos a tomar la imagen de un perro y vamos a mostrar a los alumnos qué ocurre si la aproximamos utilizando las eigenfaces.

Observamos en la figura 4.16 que en los rangos 25, 50 y 100 sigue pareciéndose más a un rostro humano. Sin embargo, conforme se va aumentando el rango de la aproximación cada vez más, se observa que empieza a apreciarse la forma de un perro y aunque la aproximación no es perfecta, con 1600 modos se puede percibir claramente la imagen de este perro concreto. Debido a que este espacio es ortogonal, es capaz de capturar muchas características que pueden utilizarse para reconstruir otro tipo de imágenes distintas a caras humanas.

Al igual que en la compresión de imágenes, tras esta aplicación de la SVD se le propondrá a los alumnos la elaboración de una animación para las eigenfaces.

El código empleado para esta aplicación de la SVD se muestra a continuación:

Código 4.10: Animación Eigenfaces

```
figure
colormap gray
```

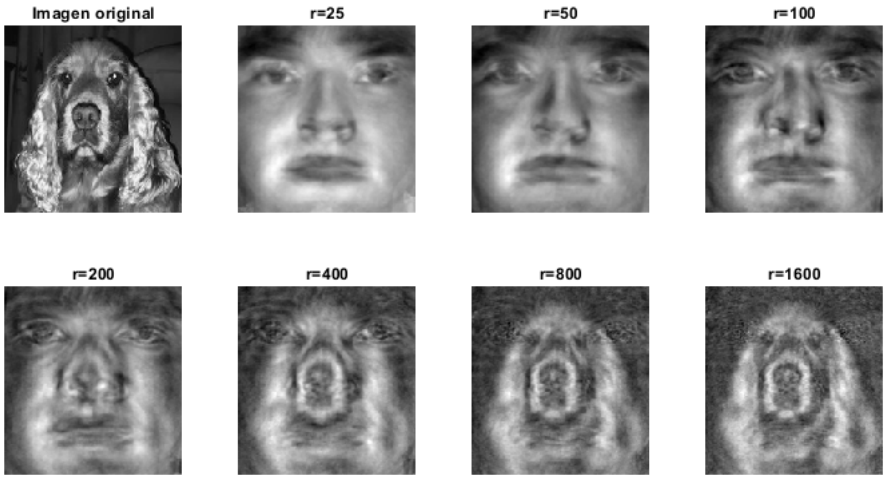


Figura 4.16: Representación de la aproximación de una imagen de un perro usando eigenfaces.

```
subplot(1,3,1);
imagesc(reshape(testFace,n,m))
axis off
title('original')
r=1600;
videoFile = VideoWriter('videoEigenfaces.avi');
open(videoFile);
for t=1:1:r
    U_red=U(:,1:t);
    reconFace = avgFace + (U_red*(U_red'*testFaceMS));
    subplot(1,3,2);
    imagesc(reshape(reconFace,n,m))
    axis off
    title(['t=',num2str(t)]);
    F=getframe(gcf);
    writeVideo(videoFile,F);
    drawnow;
    pause(0.0001)
    B=(reshape(testFace,n,m))-reshape(reconFace,n,m);
    subplot(1,3,3)
    imagesc(B,[0,1])
    axis off
    title('residuo')
end
close(videoFile);
```

4.4.3. Eliminación de ruido

En numerosos contextos, tales como la mejora visual, la extracción de rasgos distintivos y el reconocimiento de objetos, la eliminación de ruido es un paso fundamental dentro del procesamiento de imágenes, ya que permite una mayor precisión de la imagen en los pasos posteriores del procesamiento. El objetivo de la eliminación de ruido es reconstruir la imagen original con la mayor precisión posible a partir de una versión ruidosa, conservando sus características esenciales, como bordes y texturas [31].

La descomposición en valores singulares (SVD) es uno de los métodos mayormente utilizados en la eliminación de ruido de imágenes.

Se les mostrará a los alumnos el siguiente ejemplo para que observen cómo la SVD puede utilizarse para la eliminación de ruido en un caso en el que se utilizan varios receptores para recoger una señal correlacionada en presencia de ruido [32].

Ejemplo 3. *Supongamos que 5 sensores reciben una sinusoide* $\cos(\omega t)$ *y cada sensor obtiene 100 muestras. Como las ubicaciones de los sensores son diferentes, cada uno puede recibir:*

$$x_i(t) = A_i \cos(\omega t + \Phi_i) + n_i(t), \quad i = 1, ..., 5, t = 1, ..., 100$$

Introduciendo ruido en los datos, se pide calcular una nueva versión de los mismos aplicando reducción de ruido a través del cálculo de la descomposición SVD truncada. Estudiar la calidad de la reconstrucción aplicando los medidores MSE, PSNR y SNR.

Como una sinusoide puede representarse como una combinación lineal de dos sinusoides de la misma frecuencia, por ejemplo, $x_3(t) = \alpha_1 x_1(t) + \alpha_2 x_2(t)$ y $2\cos(\omega)\cos(\omega(t-1)) = \cos(\omega t) + \cos(\omega(t-2))$, la matriz $X \in \mathbb{R}^{5,100}$ construida a partir de la salida del sensor como filas es de rango 2 en ausencia de ruido de medición.

A través del siguiente código creamos y representamos en Matlab los 5 sensores con sus 100 muestras:

Código 4.11: Crear y representar los 5 sensores y sus 100 muestras

```
N=100;
w=0.3;
x1=sin(w.*(1:N));
x2=sin(w.*(1:N)+0.5);
x3=sin(w.*(1:N)+1);
x4=sin(w.*(1:N)+1.5);
x5=sin(w.*(1:N)+2);
subplot(5,1,1)
plot(x1,'r.')
subplot(5,1,2)
plot(x2,'m.')
subplot(5,1,3)
plot(x3,'g.')
```

```
subplot(5,1,4)
plot(x4,'c.')
subplot(5,1,5)
plot(x5,'b.')
```

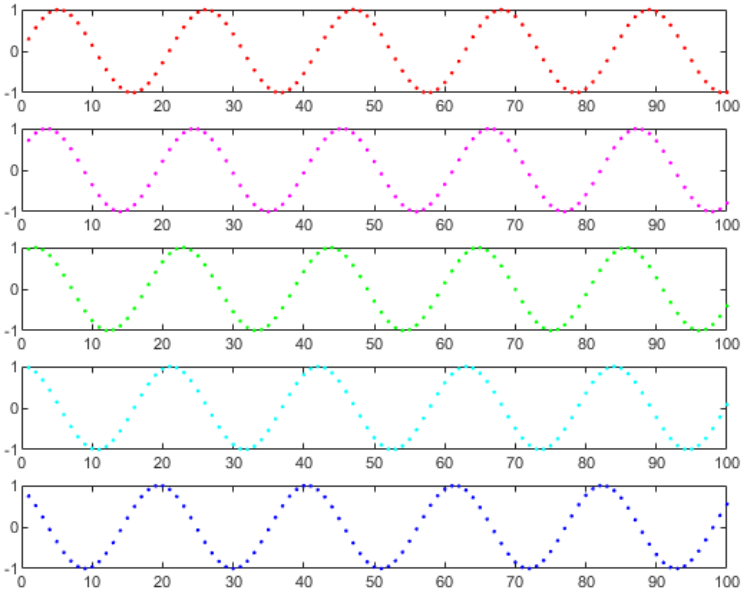


Figura 4.17: Gráfica de los 5 sensores que reciben una sinusoide $\cos(\omega t)$, obteniendo cada uno 100 muestras.

A continuación, contruimos la matriz $X \in \mathbb{R}^{5,100}$ formada por los 5 sensores como filas y cuyo rango es igual a 2:

Código 4.12: Construir matriz de sensores

```
X=[x1; x2; x3; x4; x5;];
rango=rank(X)
```

Sin embargo, en la práctica, el ruido está presente. Por ello, vamos a introducir ruido gaussiano de media cero distribuido con potencia $E\{n_i^2\} = 0{,}2^2 = 0{,}4$ a los datos sin procesar. En Matlab lo realizamos a través del siguiente código:

Código 4.13: Introducir ruido gaussiano

```
%% Datos ruidosos sin procesar
sigma=0.2
x1n=sin(w.*(1:N))+sigma*randn(1,N); % desviacion
   estandar 0.2
subplot(5,1,1)
plot(x1n,'r.')
```

```
x2n=sin(w.*(1:N)+0.5)+sigma*randn(1,N);
subplot(5,1,2)
plot(x2n,'m.')
x3n=sin(w.*(1:N)+1)+sigma*randn(1,N);
subplot(5,1,3)
plot(x3n,'g.')
x4n=sin(w.*(1:N)+1.5)+sigma*randn(1,N);
subplot(5,1,4)
plot(x4n,'c.')
x5n=sin(w.*(1:N)+2)+sigma*randn(1,N);
subplot(5,1,5)
plot(x5n,'b.')
```

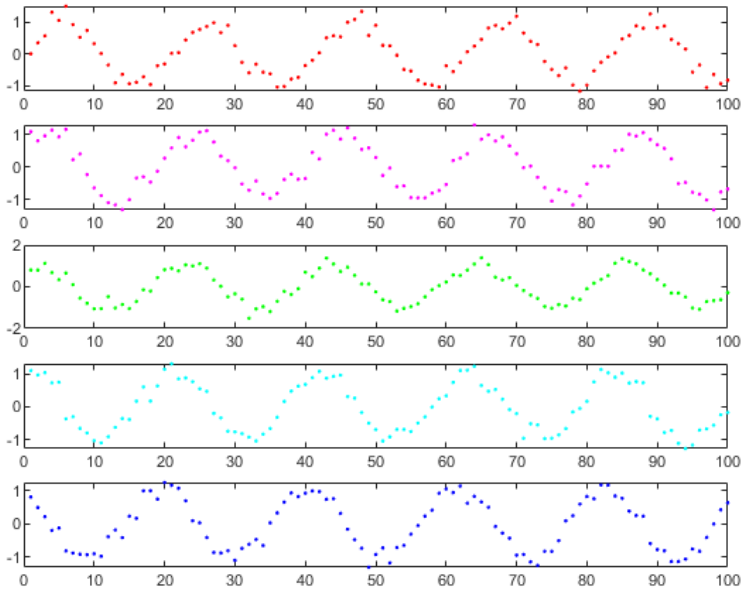


Figura 4.18: Gráfica de los 5 sensores tras haber sido contaminados con ruido.

Creamos una nueva matriz X_n formada por los 5 sensores con ruido añadido como filas y cuyo rango es igual a 5, y definimos la potencia del ruido:

Código 4.14: Crear matriz de sensores con ruido añadido y definir potencia de ruido

```
%% Datos ruidosos y potencia del ruido
Xn=[x1n; x2n; x3n; x4n; x5n];
rangon=rank(Xn)
noise_power=sigma^2
```

Una de las medidas de calidad que vamos a utilizar para medir la precisión del modelo es el error cuadrático medio (MSE, del inglés *Mean Squared Error*), que mide la cantidad de error en los modelos estadísticos, evaluando la diferencia cuadrática media entre los valores observados y previstos. El error cuadrático medio se define como

$$MSE = \frac{\sum_{n=1}^{N}(x_n - \hat{x}_n)^2}{N}$$

donde x_n es el valor real y $\hat{x}_n$ es el valor estimado de x_n. Cuando un modelo no tiene errores, el MSE es igual a cero. A medida que aumenta el error del modelo, el MSE aumenta su valor [32].

Consideramos los datos ruidosos no procesados como los valores estimados. Calculando el MSE:

$$MSE = \frac{\sum_{i=1}^{5}\sum_{t=1}^{100}(x_i(t) - \hat{x}_i(t))^2}{500} = 0{,}0413$$

que coincide con el valor de la potencia de ruido.

Para calcular la métrica MSE incial en Matlab, definimos la variable **mse_in** y calculamos el MSE mediante el comando **immse(X, Xn)**, que da como resultado:

```
The initial mean-squared error is 0.0413
```

Otros medidores de calidad del modelo son la PSNR y la SNR. La proporción máxima de señal a ruido o PSNR (del ingés, *Peak Signal-to-Noise Ratio*) mide la relación entre la máxima potencia posible de una señal y la potencia del ruido que afecta a su representación. La PSNR se define a partir de la MSE, como

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right)$$

siendo MAX el valor máximo de intensidad que aparece en la imagen [33].

Por su parte, la relación señal-ruido o SNR (del ingés, *Signal-to-Noise Ratio*) es una medida que compara el nivel de una señal deseada con el nivel de ruido de fondo presente. La SNR se define como

$$SNR = 10\log_{10}\left(\frac{\text{Potencia de la señal}}{\text{Potencia del ruido}}\right)$$

Para calcular estas dos métricas, utilizamos el siguiente código en Matlab:

Código 4.15: Calcular PSNR y SNR

```
%% Metricas PSNR, SNR
[peaksnr_in, snr_in] = psnr(X,Xn);
fprintf('\n The initial Peak-SNR value is %0.4f',
   peaksnr_in);
```

```
fprintf('\n The initial SNR value is %0.4f \n', snr_in)
   ;
\end{verbatim}
obteniendo como resultado
\begin{verbatim}
 The initial Peak-SNR value is 13.8356
 The initial SNR value is 11.1685
```

Calculamos ahora una nueva versión de los datos (X_out) aplicando reducción del ruido a través del cálculo de la descomposición SVD truncada de X con $r = 2$, ya que sabemos que el rango de X es igual a 2.

Código 4.16: SVD truncada para la reducción del ruido

```
%% Denoised data con SVD truncada
[U,S,V] = svd(Xn);
r=2;
X_out= U(:,1:r)*S(1:r,1:r)*V(:,1:r).';
```

Repetimos el cálculo del MSE con los datos sin ruido

Código 4.17: Calcular MSE de los datos sin ruido

```
%% Metrica MSE de salida
mse_out = immse(X, X_out);
fprintf('\n The final mean-squared error is %0.4f\n',
   mse_out);
```

y obtenemos

```
The final mean-squared error is 0.0188
```

Repetimos también los cálculos de la PSNR y SNR:

Código 4.18: Calcular PSNR y SNR de los datos sin ruido

```
%% Metricas PSNR, SNR de salida
[peaksnr_out, snr_out] = psnr(X,X_out);
fprintf('\n The final Peak-SNR value is %0.4f',
   peaksnr_out);
fprintf('\n The final SNR value is %0.4f \n', snr_out);
```

obteniendo como resultado

```
The final Peak-SNR value is 17.2550
The final SNR value is 14.4027
```

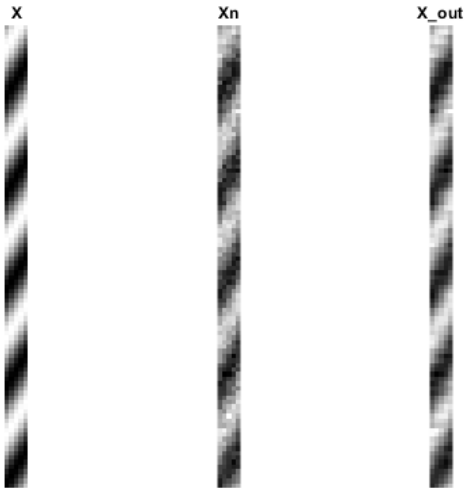


Figura 4.19: Representación de la matriz de datos original (X), la matriz de datos contaminados con ruido (Xn) y la matriz resultante al aplicar reducción de ruido mediante SVD (X_out).

Comparando las medidas MSE, PSNR y SNR iniciales y finales, calculamos las ganancias obtenidas:

Código 4.19: Calcular ganancias obtenidas en MSE, PSNR y SNR

```
1 %% Ganancias en MSE, PSNR y SNR
2 ganancias=[mse_in-mse_out, peaksnr_out-peaksnr_in,
    snr_out-snr_in]
```

que son:

```
0.0225    3.4193    3.2342
```

A partir de estos resultados, se puede demostrar claramente el rendimiento de la eliminación de ruido y sabemos cuánto ruido se reduce en términos de MSE.

Debido al ruido aleatorio, se obtendrán resultados numéricos ligeramente diferentes en cada simulación. Además, cabe mencionar que no podremos obtener un MSE cero mediante la eliminación de ruido porque los componentes truncados del SVD.

Como vimos en el apartado 4.3.1, sabemos que la SVD truncada es la mejor matriz de bajo rango en el sentido de mínimos cuadrados. En la figura 4.20 se representan los valores del sensor x1 en color azul, los valores de x1n en color rojo y en amarillo la aproximación dada por X_out.

Seguidamente, mostraremos a los alumnos otro ejemplo de la aplicación de la SVD en la eliminación de ruido sobre una imagen digital.

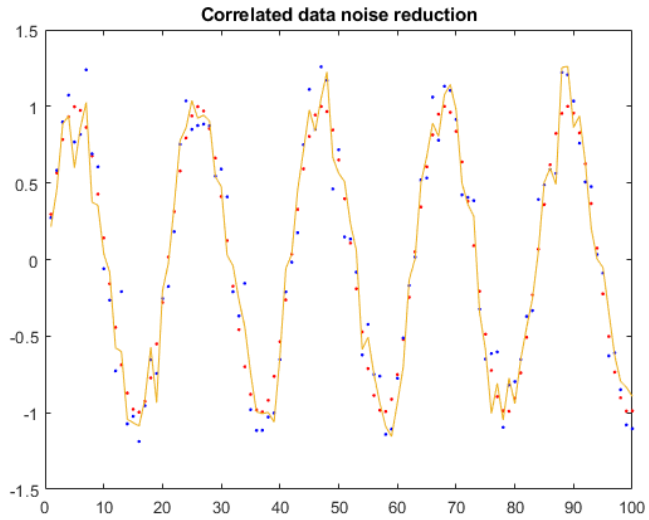


Figura 4.20: Representación de la aproximación del sensor x1 en el sentido de mínimos cuadrados para la norma de Frobenius.

Ejemplo 4. *Sea dada una imagen originalmente sin ruido. Se pide introducir ruido gaussiano sobre la imagen digital y, mediante la aplicación de la SVD truncada, reconstruir la imagen a partir de su versión ruidosa. Valorar la calidad de la reconstrucción y observar cómo el ruido introducido queda parcialmente eliminado, recuperando en gran medida la imagen original.*

Una vez cargada la imagen en Matlab, introducimos ruido gaussiano mediante el comando **imnoise(I,'gaussian', 0, 0.01)**, que introduce ruido blanco gaussiano de media 0 y con varianza de 0.01 [34]. En la figura 4.21 podemos ver la imagen original y la versión resultante tras haber introducido ruido.

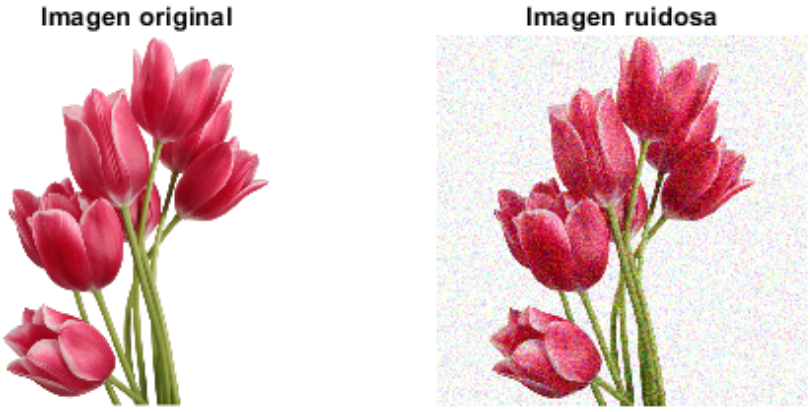


Figura 4.21: Representación de la imagen original (izquierda) y de su versión ruidosa (derecha).

Calculamos la PSNR para medir la relación señal-ruido máximo de la versión ruido-

sa con respecto a la imagen original, es decir, para evaluar la calidad de la imagen tras la introducción de ruido en comparación con la imagen original. Para ello utilizamos la función **PSNR_V(I,Inoise)**, creada en Matlab mediante el siguiente código:

Código 4.20: Función PSNR_V

```
function psnr=PSNR_V(I1, I2)

if (size(I1)~=size(I2))
    error('Las imagenes tienen que tener el mismo
       tamanio');
end
if ((I1-I2)==0)
    error('Las imagenes son iguales');
end

maximo=max(I1(:));
N=size(I1,1);
M=size(I1,2);
P=size(I1,3);

if (P==1)
    MSE=(1/(M*N))*sum(sum((I1-I2).^2));
else
    MSE=(1/(M*N*P))*sum(sum(sum((I1-I2).^2)));
end

psnr=20*log10(maximo/sqrt(MSE));

end
```

El resutado obtenido tras aplicar la PSNR entre la imagen ruidosa y la imagen original es:

```
PSNR I-Inoise = 21.8782
```

La imagen original tiene unas dimensiones de 850×850. Para reconstruir la versión comprimida, primero, vamos a calcular la descomposición SVD, tanto de la imagen original como de su versión ruidosa.

A continuación, vamos a reconstruir la imagen manteniendo unicamente los $k = 100$ primeros modos de la imagen. Para ello, empleamos el siguiente código, que aproxima la imagen al valor k al mantener los primeros k valores singulares de la matriz diagonal Σ (en nuestro código de Matlab S). Además, se calcula el error relativo en términos de la norma de Frobenius y la señal de ruido PSNR:

Código 4.21: Reconstruir la imagen utilizando SVD

```
[m,n,k]=size(I);
K = 1:100;

%Separa los canales RGB
channelsI = cell(1,3);
channelsIn = cell(1,3);
for i = 1:3
channelsI{i} = I(:,:,i);
channelsIn{i} = Inoise(:,:,i);
end

for i = 1:length(K)
     for j = 1:3
         %SVD
         [U, S, V] = svd(channelsI{j});
         [Un, Sn, Vn] = svd(channelIn{j});

         %Mantiene unicamente k modos de
         Sk=S;
         Snk=Sn;
         Sk(K(i):end,K(i):end)=0;
         Snk(K(i):end,K(i):end)=0;

         %Recupera la imagen
         Irecover(:,:,j)=U*Sk*V';
         Inoiserecover(:,:,j)=Un*Snk*Vn';
    end

    %Error relativo con la norma de frobenius
    E(i)=norm(I-Irecover,'fro')/norm(I,'fro');
    En(i)=norm(Inoise-Inoiserecover,'fro')/norm(I,'fro'
       );

    %PSNR
    PSNRorirecover(i)= psnr(I,Inoiserecover);
    PSNRnoiserecover(i)= psnr(Inoise,Inoiserecover);
end
```

Representamos el error relativo en términos de la norma de Frobenius, calculado mediante el código anterior, entre la imagen original y su aproximación comprimida y la versión ruidosa y su aproximación, en la figura 4.22. Recordamos que la definición de la norma de Frobenius se encuentra en el teorema 4 de la sección 4.3.1.

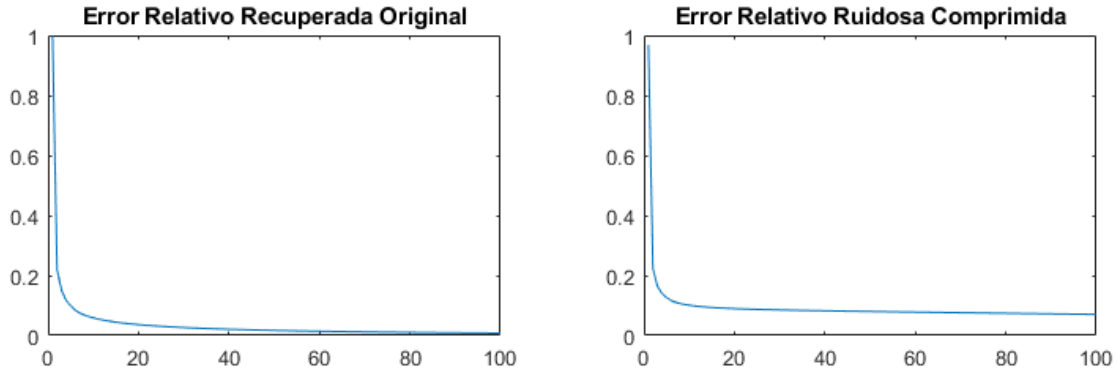


Figura 4.22: Representación del error relativo entre la imagen original y su aproximación con $k = 100$ modos (izquierda) y entre la versión ruidosa y su aproximación con $k = 100$ modos (derecha).

En ambas gráficas de la figura 4.22 podemos observar que ya a partir del cuarto o quinto modo el error tiene una caída muy pronunciada, quedando muy próximo a 0. Conforme van aumentándose los modos, el error continúa disminuyendo, pero de forma más estable. Una vez más observamos que el primer grupo de valores singulares es el que mantiene la mayoría de la información de los datos, por lo que podemos desechar muchos de los valores singulares de poca energía sin perder información de la matriz original.

Contrastando ambas gráficas vemos que el error relativo entre la imagen original y su aproximación al utilizar los 100 modos es prácticamente 0, mientras que el error relativo entre la versión ruidosa y su aproximación es algo mayor, lo que indica que la aproximación se aleja un poco más de la versión original en el caso ruidoso.

A continuación, representamos la proporción máxima de señal a ruido (PSNR) entre la imagen original y su aproximación y entre la versión reducida y su aproximación, también calculados en el código anterior, ambos mostrados en la figura 4.23.

En ambas gráficas de la figura 4.23 podemos observar que a partir de cuarto o quinto modo el valor de la PSNR crece de forma muy pronunciada. A medida que se van aumentando los modos, el valor PSNR va aumentando de forma relativamente constante. Como ya sabemos, el valor PSNR mide el grado de similitud entre la imagen procesada y la imagen original y cuanto más alto es el valor de la PSNR, mejor es la calidad de la imagen.

Lo que cabría esperar es que cuanto más modos utilice en la reconstrucción, más cercana será a la imagen original. Sin embargo, en la gráfica izquierda observamos que hay un pico a partir del cual el valor de la PSNR comienza a decrecer. Esto significa que el número de modos óptimo es aquel en el que se alcanza el valor máximo de la PSNR y a partir de ese valor la representación empieza a tener peor calidad.

El valor máximo que alcanza la PSNR de la gráfica entre la imagen original y

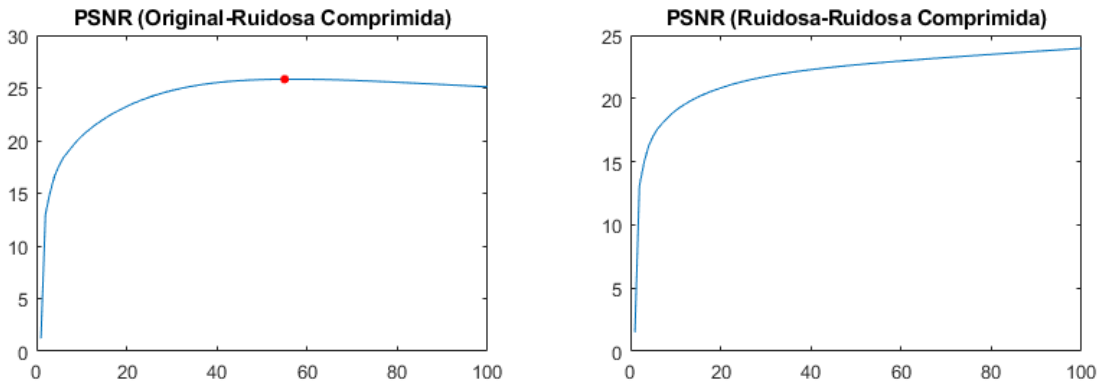


Figura 4.23: Representación del valor PSNR entre la imagen original y la aproximación a partir de la versión ruidosa con $k = 100$ modos (izquierda) y entre la versión ruidosa y su aproximación con $k = 100$ modos (derecha).

la aproximación a partir de la versión ruidosa (gráfica izquierda de la figura 4.23) es 25,8248, alcanzado en el modo $k = 56$. Estos valores se calculan en Matlab mediante el comando **[value, kk]=max(PSNRorirecover)**.

Finalmente, en las figuras 4.24, 4.25, 4.26 y 4.27 se muestran representados los resutados obtenidos tras realizar la reconstrucción sobre las imágenes.

Figura 4.24: Representación de la imagen original (izquierda), de su versión ruidosa (centro) y de la aproximación obtenida a partir de la versión ruidosa utilizando $k = 100$ modos (derecha).

En las figuras 4.24 y 4.25 se pueden comparar los resultados obtenidos al reconstruir las imágenes a partir de la versión ruidosa utilizando 100 modos y 56 modos. Observamos que con 56 modos hemos obtenido una mejor aproximación de la imagen que con 100 modos debido a que 56 es el número óptimo de modos con el que se obtenía una mayor calidad en la representación.

En las figuras 4.26 y 4.27 se han recortado las imágenes original, ruidosa y ruidosa

Figura 4.25: Representación de la imagen original (izquierda), de su versión ruidosa (centro) y de la aproximación obtenida a partir de la versión ruidosa utilizando $k = 56$ modos (derecha).

comprimida, mostrando una región más pequeña de estas, de forma que pueda apreciarse con mayor detalle las diferencias y texturas de las imágenes y, con ello, la calidad de la aproximación. El comando de Matlab para realizar los recortes de una imagen es **imcrop(I)**. El código que se ha utilizado es el siguiente:

Código 4.22: Definir crops de las imágenes

```
%% Crops definicion y representacion
Ic=imcrop(I,[245 108 322 226]);
Jc=imcrop(J,[245 108 322 226]);
I_out=imcrop(Inoiserecover,[245 108 322 226]);
```

Figura 4.26: Representación de una región más pequeña de la imagen original (izquierda), de su versión ruidosa (centro) y de la aproximación obtenida a partir de la versión ruidosa utilizando $k = 100$ modos (derecha).

Figura 4.27: Representación de una región más pequeña de la imagen original (izquierda), de su versión ruidosa (centro) y de la aproximación obtenida a partir de la versión ruidosa utilizando $k = 56$ modos (derecha).

A través de las figuras anteriores podemos observar con mayor detalle que la reconstrucción realizada con $k = 100$ conserva más ruido que el caso de $k = 56$ modos.

5

Conclusiones y líneas futuras

5.1. Conclusiones

Este trabajo de fin de grado pretende visibilizar la importancia de enseñar desde la motivación, proponiéndose, para ello, una metodología docente basada en la puesta en práctica de aplicaciones relevantes en diferentes campos científicos como medio de explicación de conceptos y técnicas matemáticos.

En las siguientes líneas se reúnen las conclusiones extraídas teniendo en cuenta los objetivos planteados como punto de partida.

A lo largo del capítulo 4, concretamente en las secciones 4.1 y 4.2, se ha descrito una metodología docente que pretende, en todo momento, fomentar la motivación en el aprendizaje de los alumnos a través de la inclusión de aplicaciones, ejemplos prácticos y actividades que conecten la teoría con situaciones reales. En este capítulo se incluye, además, un ejemplo completo de seminario y prácticas que sirve como modelo del resto del curso, desarrollado en las secciones 4.3, 4.3 y 4.4. Es por ello que puede concluirse que el objetivo principal planteado ha sido satisfecho, así como los objetivos 5 y 6.

Respecto al objetivo específico 1, la elección de la Descomposición en Valores Singulares (SVD) resulta un tema adecuado para el seminario debido a su relación tan directa con la asignatura de Álgebra Lineal. Es por ello que la metodología propuesta se enmarca dentro de esta asignatura, pues para comprender la teoría, propiedades y aplicaciones de la SVD (que en principio resultan de *alto nivel*) serán necesarios conceptos previos de Álgebra Lineal como son las matrices y sus operaciones, que permitirán trabajar con imágenes de cualquier tipo; los vectores y los espacios vectoriales; la ortogonalidad o la diagonalización de matrices, donde los autovalores son los análogos de los valores singulares.

Durante el desarrollo de todo el seminario sobre la SVD (4.3) se ha utilizado el tratamiento de imágenes digitales como principal herramienta didáctica para motivar a los alumnos, verficando así el objetivo 2. A través de las imágenes se muestran los conceptos meramente teóricos, ejemplos prácticos y aplicaciones, haciéndolos así más accesibles para los estudiantes, ya que, como hemos explicado en la sección 3.2.1, las imágenes pueden tener un impacto muy positivo durante el proceso de enseñanza-aprendizaje.

En este mismo capítulo, a la vez que se explican los conceptos teóricos de la SVD, se han ido introduciendo herramientas de Cálculo simbólico y numérico de tipo Matlab en los ejemplos y aplicaciones desarrollados con la intención de que los alumnos puedan poner en práctica los conceptos y experimentar por ellos mismo las aplicaciones de estos a situaciones concretas, lo que satisface los objetivos 3 y 4. Esto supone otra fuerte herramienta didáctica para motivar a los estudiantes, ya que les ayuda a comprender la utilidad de las matemáticas en la práctica y les permite explorar los modelos matemáticos de una manera más tangible a través de la visualización y realización de simulaciones. Además, la inclusión de prácticas descarga tanto peso en la parte teórica, lo que también sirve de incentivo para la implicación de los alumnos a la hora de aplicar de forma adecuada las herramientas de cálculo simbólico y numérico.

5.2. Limitaciones

La metodología docente que se ha propuesto en este trabajo de fin de grado se enmarca dentro de un curso de Álgebra Lineal que ya cuenta con una metodología de trabajo del curso. Normalmente, los programas académicos de las asignaturas ya cuentan con una limitación en el tiempo, ya que este siempre suele resultar insuficiente. Para el diseño de esta nueva metodología se ha tenido muy en cuenta la distribución temporal para que su puesta en práctica no suponga un gran desafío. Además, el hecho de introducir una mayor experiencialidad no supondría tener que sacrificar parte del temario, sino que la manera de abordar parte de este sería a través de la práctica.

Por otro lado, la propuesta asume que los estudiantes tienen ciertas habilidades matemáticas y conocimientos previos sobre Álgebra Lineal, que habrán ido adquiriendo a lo largo del curso. Sin embargo, es posible que se den diferencias en las habilidades y conocimientos adquiridos por los estudiantes, lo que podría suponer una dificultad a la hora de llevar a cabo esta metodología, ya que se trabajan conceptos algo más avanzados. Por ello, se ha planteado la posibilidad de proporcionar a los alumnos notas con algunos de los resultados la semana antes de empezar el curso, de manera que puedan ir familiarizándose con ellos. Además, con este tipo de metodología se pretende acercar los conceptos a los alumnos de una manera más atractiva y de forma que puedan asimilarse más fácilmente y con un carácter global, lo que podría servirles para terminar de integrar los conceptos previos.

5.3. Líneas futuras

Aunque a lo largo de este trabajo se ha descrito y caracterizado la metodología docente y evaluadora que sigue esta propuesta de intervención, únicamente se ha desarrollado un ejemplo modelo de aplicación de la metodología al Álgebra Lineal y a través de la descomposición en valores singulares. El paso siguiente sería implementar esta metodología en todo un curso, adaptando la forma de enseñar los conceptos y teorías a este formato en el que la parte práctica y aplicada toma un papel protagonista.

Además, el alcance de la propuesta podría ampliarse aplicando la visión artificial para motivar el estudio, no sólo de cualquier otro área de Matemáticas, sino también de Ingenierías e Informáticas. En este sentido, se podría, también, favorecer la colaboración entre departamentos o disciplinas para analizar las aplicaciones interdisciplinarias de la visión artificial y las técnicas matemáticas.

Bibliografía

[1] T. Roughgarden and G. Valiant, "The singular value decomposition (svd) and low-rank matrix approximations," *The Modern Algorithmic Toolbox. Stanford University*, 2022.

[2] J. Shlens, "A tutorial on principal component analysis," *Center for Neural Science, NYU y Systems Neurology Laboratory, Salk Institute for Biological Studies La Jolla*, 2009.

[3] F. Michavila, "La innovación educativa. oportunidades y barreras," *ARBOR Ciencia, Pensamiento y Cultura*, vol. 185, no. Extra, pp. 3–8, 2009. [Online]. Available: https://arbor.revistas.csic.es/index.php/arbor/article/view/373/374

[4] B. Pacheco-Salazar, "Siete claves para la innovación educativa," *El País*, 2020. [Online]. Available: https://elpais.com/elpais/2020/07/31/planeta_futuro/1596204508_015285.html

[5] V. Londoño-Osorio, J. Marín-Pineda, and E. I. Arango-Zuluaga, "Introducción a la visión artificial mediante prácticas de laboratorio diseñadas en matlab," *Tecnológicas*, pp. 591–603, 2013. [Online]. Available: https://www.redalyc.org/articulo.oa?id=344234341045

[6] C. S. González, "Estrategias para trabajar la creatividad en la educación superior: pensamiento de diseño, aprendizaje basado en juegos y proyectos," *Revista de Educación a Distancia (RED)*, no. 40, 2015. [Online]. Available: https://revistas.um.es/red/article/view/234291/180001

[7] M. del Pilar Gutiérrez-Arenas, C. Corpas-Reina, and A. Ramírez-García, "Visual thinking en una metodología activa de enseñanza-aprendizaje universitaria," *HUMAN REVIEW. International Humanities Review / Revista Internacional De Humanidades*, vol. 11, no. Monográfico, pp. 1–16, 2022. [Online]. Available: https://journals.eagora.org/revHUMAN/article/view/4090/2470

[8] I. M. Grande, "Visual thinking: dibujando el aprendizaje," *UNIR*, 2018. [Online]. Available: https://www.unir.net/educacion/revista/visual-thinking-dibujando-el-aprendizaje/

[9] Vive, "El aprendizaje significativo: ¿por qué introducirlo en el aula?" *UNIR*, 2018. [Online]. Available: https://www.unir.net/educacion/revista/aprendizaje-significativo/

[10] J. L. R. Muñoz, "El aprendizaje significativo y la evaluación de los aprendizajes," *Revista de investigación educativa*, vol. 8, no. 14, pp. 47–52, 2004. [Online]. Available: https://revistasinvestigacion.unmsm.edu.pe/index.php/educa/article/view/7098/6272

[11] M. A. Moreira, "¿al final, qué es el aprendizaje significativo?" *Revista Qurriculum*, pp. 29–56, 2012. [Online]. Available: https://riull.ull.es/xmlui/bitstream/handle/915/10652/Q_25_%282012%29_02.pdf?sequence=5&isAllowed=y

[12] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM review*, vol. 35, no. 4, pp. 551–566, 1993.

[13] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[14] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.

[15] H. Abdi, "Singualar value decomposition (svd) and generalized singular value decomposition (gsvd)," *Encyclopedia of measurement and statistics*, 2007.

[16] A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*. Cambridge University Press, 2018. [Online]. Available: https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf

[17] D. G. Chen, "Matriz norm and low-rank approximation," *Mathematical Methods for Data Visualization. San José State University*, s.f.

[18] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. n.d., 2019.

[19] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[20] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[21] A. Eleyan and H. Demirel, *Pca and lda based neural networks for human face recognition*. I-Tech Education and Publishing, 2007.

[22] S. Codesido, M. Hanafi, Y. Gagnebin, V. González-Ruiz, S. Rudaz, and J. Boccard, "Network principal component analysis: a versatile tool for the investigation of multigroup and multiblock datasets," *Bioinformatics*, vol. 37, no. 9, pp. 1297–1303, 2021.

[23] n.d., "Pca whitening," *Standford University*, s.f.

[24] F. P. M. . N. P. Mariné, "Descomposición en valores singulares: introducción y aplicaciones," *Fundació Universitat Oberta de Catalunya (FUOC)*, 2020.

[25] K. Mounika, D. S. N. Lakshmi, and K. Alekya, "Svd based image compression," *International Journal of Engineering Research and General Science*, vol. 3, 2015.

[26] L. Sirovich and M. Kirby, "A low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America A*, vol. 4, no. 3, pp. 519–524, 1987.

[27] ——, "Application of the karhunen-loeve procedure for the characterization of human faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 12, no. 1, pp. 103–108, 1990.

[28] M. Turk and A. Pentlan, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[29] Y. Wang, "Svd and eigenfaces," *Nextjournal*, 2021.

[30] B. P. N. Georghiades, A. S. and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 6, pp. 643–660, 2001.

[31] Y. Z. Qiang Guo, Caiming Zhang and H. Liu, "An efficient svd-based method for image denoising," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 868–880, 2016.

[32] "Singular value decomposition and applications," 2021, https://www.ee.cityu.edu.hk/ hcso/EE4016_svd.pdf.

[33] *The MathWorks, Inc.* [Online]. Available: https://es.mathworks.com/help/vision/ref/psnr.html

[34] *The MathWorks, Inc.* [Online]. Available: https://es.mathworks.com/help/images/ref/imnoise.html

[35] R. Criado and A. Gallinari, *Álgebra.* n.d., 2003.

Apéndices

A

Conceptos previos de Álgebra Lineal

En esta sección se introducen los conceptos de Álgebra Lineal que serán necesarios para entender la construcción y el cálculo de la Descomposición en Valores Singulares (SVD).

A.1. Escalares, Vectores, Matrices y Tensores

Cualquier maginitud física, artificial o lógica puede almacenarse y representarse a través de Escalares, Vectores, Matrices y Tensores. Se entiende como escalar un número real $\lambda \in \mathbb{R}$ y como escalar complejo, un número $z = a+ib = (a, b) \in \mathbb{R}^2$ donde se utiliza un vector $(a, b) \in \mathbb{R}^2$ para almacenar el número complejo, siendo $a \in \mathbb{R}$ la parte real y $b \in \mathbb{R}$ la parte imaginaria del número complejo $z \in \mathbb{R}^2$. Se puede, por tanto, visualizar un número complejo como un punto en el plano real.

Un vector puede definirse mediante una colección ordenada de escalares denotada por $\mathbf{v} = (v_1, ..., v_m) \in \mathbb{R}^n$ siendo $v_i \in \mathbb{R}$ la i-ésima componente. Una colección ordenada de m vectores $\mathbf{v_q}, ..., \mathbf{v_m}$ de $\mathbb{R}^n$ se puede almacenar en una matriz de n filas y m columnas denotada por $A \in \mathbb{R}^{n,m}$. Un elemento genérico de una matriz se denota por $a_{i,j}$ y su valor se corresponde a la intensidad de gris del píxel correspondiente.

Para generar una matriz aleatoria $M \in \mathbb{R}^{m,n}$ y representarla gráficamente se pueden utilizar las instrucciones

Código A.1: Generar matriz aleatoria

```
1 m=8; n=5; A=rand(m,n);
2 figure, colormap gray, imagesc(A)
```

Ejemplo 5. *Sea dada una imagen en escala de grises. Se muestra a continuación la forma*

de leer la imagen en Matlab, almacenarla en una matriz, pasarla a doble presición en el rango [0,1] y visualizarla.

Tras almacenar la imagen en una carpeta se puede leer y almacenar en Matlab mediante el comando

Código A.2: Leer y almacenar imagen en Matlab

```
f=imread('./imagenes/dalibw.jpg');
```

donde $f \in M^{364,507}$. La pasamos a a doble precisión en el rango [0, 1] mediante el comando

Código A.3: Pasar imagen a doble precisión

```
f=im2double(f);
```

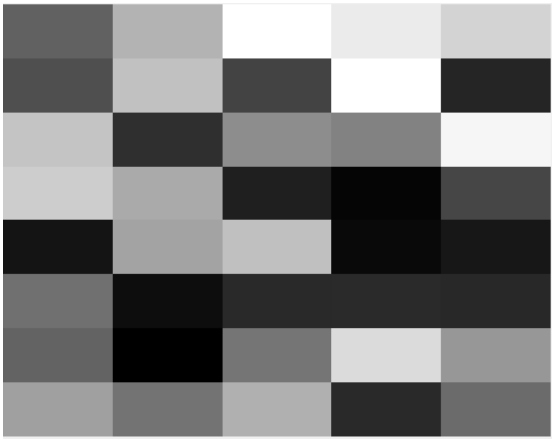


Figura A.1: Matriz aleatoria de dimensiones 8×5. Codificación de la información. Se detecta rápidamente un patrón aleatorio (random).

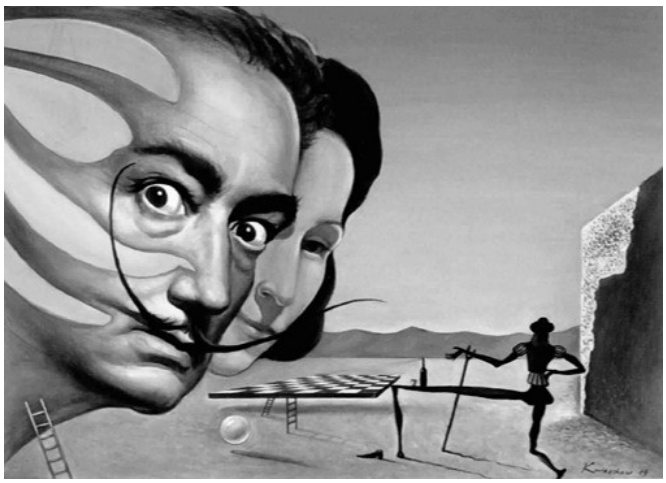


Figura A.2: La imagen se codifica como una matriz de dimensiones 364×507.

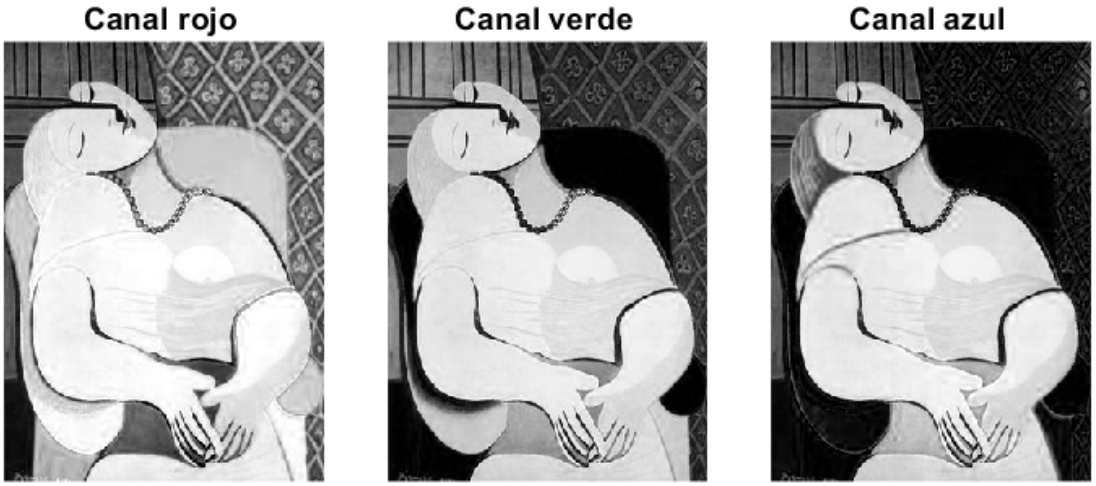
Podemos visualizar la imagen (figura A.2) mediante

Código A.4: Visualizar una imagen

```
figure
imshow(f,[])
```

Una matriz con una única columna representa un vector. Si tiene una única fila es un vector fila. Un vector con una única componente es un escalar. Si se dota a una matriz de una dimensión ulterior se obtiene un **tensor** que se denota con letra mayuscula y en negrita $\mathbf{A}$ siendo $a_{i,j,k} \in \mathbf{A}$ un elemento genérico. Esta estructura se puede generalizar a cualquier dimensión.

Figura A.3: La imagen en color se almacena como un tensor $\mathbf{A}$.



(a) Canal Rojo $i = 1$ (b) Canal Verde $i = 2$ (c) Canal Azul $i = 3$

Figura A.4: Codificación de una imagen en color RGB en un tensor $\mathbf{A}(:,:,i), i = 1, 2, 3$

Como resultado de este formalismo algebraico se observa que todas las imágenes digitales se pueden identificar con una matriz si son imágenes en escala de grises o con un tensor si son

imágenes en color del tipo RGB. Más en general se pueden codificar imágenes multi-canales del tipo de las que aparecen en imágenes médicas cuando se consideran distintas modalidades de adquisición.

Suma de matrices y tensores

Las matrices y los tensores se pueden sumar elemento a elemento, lo que implica una condición de compatibilidad. Deben tener la misma dimensión. Operando elemento a elemento, se tiene

$$A + B = C, a_{i,j} + bi, j = c_{i,j}$$

Rescalamiento

Las matrices se pueden multiplicar por un escalar $\lambda \in \mathbb{R}$ obteniendo una nueva matriz o tensor con la misma dimensión:

$$B = \lambda A, b_{i,j} = \lambda a_{i,j}$$

Combinaciones lineales

En general, dados los escalares $\alpha, \beta \in \mathbb{R}$ se puede realizar una combinación lineal de matrices y tensores en la forma:

$$C = \alpha A + \beta B$$

obteniendo una matriz o tensor de igual dimensión.

Ejemplo 6. *Se consideran dos imágenes en escala de grises de las mismas dimensiones. Fijados $\alpha, \beta \in \mathbb{R}$ calcular y representar su combinación lineal.*

Se muestra a continuación la forma de realizar una combinación lineal entre ambas. Se leen y se almacenan las imágenes como matrices en Matlab y se toman valores para α y β para realizar la combinación lineal de la siguiente forma

Código A.5: Combinación lineal

```
1 A=imread('./imagenes/cameraman.png');
2 B=imread('./imagenes/barbara.png');
3 alpha=0.6
4 beta=0.4
5 C=alpha*A+beta*B;
```

El resultado obtenido tras realizar la combinación lineal entre las dos imágenes se muestra en la figura A.5.

Figura A.5: Combinación lineal entre matrices.

Espacio vectorial

Dotado de las operaciones suma y producto escalar, el conjunto de las matrices de las mismas dimensiones $V = M^{m,n}(\mathbb{R})$ tiene estructura de espacio vectorial. Ocurre lo mismo con el conjunto de tensores $V = M^{m,n,k}(\mathbb{R})$.

Trasposición

La trasposición es una de las operaciones más importantes en álgebra lineal. Sea $A = (a_{i,j})$. Se define la matriz traspuesta como $A^T = (a_{j,i})$. Es decir, se intercambian las filas por las columnas.

Una de las razones fundamentales de la importancia y aplicabilidad del álgebra a la Visión Artificial es que las matrices codifican las imágenes digitales permitiendo aplicar el álgebra lineal al procesado de imágenes.

Modelo de ruido gaussiano

Las operaciones suma y multiplicación permitirán, por ejemplo, definir un proceso de contaminación aditivo (y gaussiano) para imágenes digitales. Para ello, se puede generar una imagen conteniendo sólo ruido gaussiano a través de los comandos

Código A.6: Ruido gaussiano

```
1 f = imread('.\imagenes\...png');
2 f=im2double(f);
3 noise=randn(size(f));
```

```
4 lambda=max(f(:))*0.1;
```

A continuación, se puede definir un modelo auditivo de ruido uniforme de tipo gaussiano mediante

Código A.7: Modelo auditivo de ruido uniforme Gaussiano

```
1 fn=f+lambda*noise;
```

donde $\lambda \in (R)^+$ es un parámetro real positivo que pondera la intensidad del ruido. La cantidad de ruido se rescala en función del máximo de la imagen y la imagen del ruido gaussiano (normal) matiene las mismas dimensiones de la imagen de entrada.

A.1.1. Multiplicación de Matrices y Vectores

La operación de multiplicación entre matrices, conocida como producto de filas por columnas, se define como:

$$A_{m,n}B_{n,p} = C_{m,p}, \qquad c_{i,j} = \sum_k a_{i,k}b_{k,j}$$

La condición de compatibilidad consiste en que el número de fias de A sea igual al número de columnas de B. La operación producto tiene las siguientes propiedades [35]:

1. No es *conmutativa*, es decir, $AB \neq BA$.
2. Es *asociativa*: $A(BC) = (AB)C$
3. Es *distributiva* respecto de la suma: $A(B + C) = AB + AC$

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

$$(AB)^T = B^T A^T$$

El Producto de Hadamard

Para matrices de la misma dimensión se define el Producto de Hadamard, que multiplica las matrices elemento a elemento dando como resultado una matriz de la misma dimensión que los operandos. Se denota por

$$A \odot B = C, \qquad c_{i,j} = a_{i,j}b_{i,j}$$

El Producto de Hadamard sí cumple la propiedad conmutativa.

Ejercicio 5. *Sean las matrices*

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix}$$

Calcular el producto AB con Matlab y verificar que no es conmutativo. Calcular el producto de Hadamard $A \odot B$ y comprobar que es conmutativo.

Producto escalar

Los vectores pueden multiplicarse mediante la operación de producto escalar, que da como resultado un escalar. Existen diferentes formas de denotar el producto escalar

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} u_i v_i = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u} = |\mathbf{u}||\mathbf{v}| cos(\theta) \in \mathbb{R}$$

donde se enfatiza que el producto escalar es conmutativo, ya que el resultado de la operación es un escalar. En particular se tiene

$$\langle \mathbf{u}, \mathbf{u} \rangle = \mathbf{u} \cdot \mathbf{u} = \mathbf{u}^T \mathbf{u} = |\mathbf{u}|^2 \in \mathbb{R}$$

es decir, el producto escalar de un vector consigo mismo es igual a su módulo al cuadrado. El producto escalar permite definir el concepto de ortogonalidad entre vectores y entre subespacios vectoriales. Dos vectores no nulos son ortogonales si y sólo si

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u} = |\mathbf{u}||\mathbf{v}| cos(\theta) = 0$$

ya que, para cumplirse la ecuación necesariamente $\theta = \pi/2$.

Ejercicio 6. *Sean los vectores* $\mathbf{u} = [2, 1]^T$, $\mathbf{v} = [-1, 3]^T$. *Calcular el producto escalar* $\langle \mathbf{u}, \mathbf{v} \rangle$ *y el ángulo en grados entre los dos vectores.*

Complemento ortogonal

Sea $V \in \mathbb{R}^m$ un subespacio vectorial de $\mathbb{R}^m$. Se llama Complemento Ortogonal de V al espacio $V^\perp$ tal que

$$V^\perp = \{\mathbf{x} | \langle \mathbf{z}, \mathbf{x} \rangle = 0, \forall \mathbf{z} \in V\}$$

Se tiene que V y su complemento ortogonal están en suma directa (ortogonal)

$$V \oplus V^\perp = \mathbb{R}^m$$

es decir, todo vector $\mathbf{w} \in \mathbb{R}^m$ puede descomponerse, de manera única, como suma de dos vectores

$$\mathbf{w} = \mathbf{z} + \mathbf{x}, \qquad \mathbf{z} \in V, \qquad \mathbf{x} \in V^\perp$$

Ejercicio 7. *Dado el subespacio vectorial* $V \in \mathbb{R}^3$ *generado por los vectores* $\mathbf{u}_1 = (1, 2, 1)^T, \mathbf{u}_2 = (-1, 0, 0)^T$, *es decir,* $V = L[\mathbf{u}_1, \mathbf{u}_2]$. *Demostrar que su complemento ortogonal es* $V^\perp = L[\mathbf{u}_3]$, *siendo* $\mathbf{u}_3 = (0, 1, -2)^T$. *Verificar que un genérico vector* $\mathbf{w} \in \mathbb{R}^3$ *puede escribirse de manera única como suma de un elemento de* V *y uno de* V^T. *Por último, verificar que el vector* $\mathbf{w} = (0, 1, 2)^T$ *sólo puede escribirse en la forma*

$$\frac{4}{5}\begin{pmatrix}1\\2\\1\end{pmatrix} + \frac{4}{5}\begin{pmatrix}-1\\0\\0\end{pmatrix} - \frac{3}{5}\begin{pmatrix}0\\1\\-2\end{pmatrix} = \begin{pmatrix}0\\8/5\\4/5\end{pmatrix} + \begin{pmatrix}0\\-3/5\\6/5\end{pmatrix} = \begin{pmatrix}0\\1\\2\end{pmatrix}$$

Finalmente, se observa que es posible definir también un producto entre vectores de la forma columnas por filas, cuyo resultado es una matriz. Sean $\mathbf{u}_{n,1}, \mathbf{v}n, 1$ vectores de $\mathbb{R}^n$. Entonces

$$\mathbf{u}_{n,1}(\mathbf{u}_{n,1})^T = \mathbf{u}_{n,1}\mathbf{u}_{1,n}^T = A_{n,n}$$

Esta operación será utilizada para descomponer una matriz mediante la técnica de valores

singulares.

A.1.2. Sistemas Lineales

Con estas nociones puede definirse el problema fundamental del Álgebra Lineal, que consiste en la resolución del Sistema de Ecuaciones Lineales definido por

$$A\mathbf{x} = \mathbf{b}$$

donde $A \in \mathbb{R}^{m,n}$ es la matriz de sistema, $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ es el vector de incógnitas y $\mathbf{b} \in \mathbb{R}^{\mathbf{m}}$ es el vector de datos. Si $\mathbf{b} = \mathbf{0}$ el sistema $A\mathbf{x} = \mathbf{b}$ es homogéneo. El conjunto de soluciones de un sistema homogéneo es un espacio vectorial. Dependiendo de los valores m y n el análisis y resolución del sistema presenta distintos escenarios. Existe una multitud de aplicaciones basadas en resolver sistemas lineales. Uno de los retos más importantes del cálculo científico es conseguir una resolución computacionalmente eficiente a través de algoritmos.

Dependiendo del número de ecuaciones m y del número de variables incógnitas n, los sistemas lineales se clasifican en **sobredeterminados** si $m > n$ e **indeterminados** si $m < n$. Si $m = n$ el sistema es cuadrado.

Un sistema lineal se dice **Bien Planteado en el sentido de Hadamard** si se cumplen 3 condiciones:

1. Existe solución.
2. Es única.
3. Depende con continuidad de los datos del problema.

Si alguna de las tres propiedades no se cumple el sistema se dice de **mal planteado**.

Se recuerda que un sistema lineal se denomina **compatible** si existe al menos una solución e **incompatible** si no existe ninguna. En caso de tener una única solución el sistema se denomina **compatible determinado**.

La última de las tres propiedades que verifica un Sistema Bien Planteado (Dependencia Continua) se resume en que si los datos del problema tienen pequeñas perturbaciones la solución del problema también tendrá pequeñas perturbaciones en sus componentes. Se puede analizar considerando el número de condicionamiento del sistema, lo que se verá más adelante al apoyarse en el concepto de autovalor.

A.1.3. Matriz Identidad e Inversa

Para la resolución de sistemas lineales de tipo cuadrado ($m = n$) la opración de inversión matricial es una herramienta fundamental. Primeramente, es necesario introducir la definición de Matriz Identidad. Se trata de una matriz cuadrada, denotada por $I_n \in \mathbb{R}^{n,n}$ tal que

$$I_n\mathbf{x} = \mathbf{x}, \qquad \forall \mathbf{x} \in \mathbb{R}^{\mathbf{n}}$$

es decir, deja invariante al vector sobre el que actúa. Por ejemplo, en $\mathbb{R}^3$ se tiene

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

y, en general, denotando $a_{i,j} \in I_n$ a la genérica componente, se tiene $a_{i,j} = 0$ si $i \neq j$ y $a_{i,j} = 1$ $\forall i$.

Si $A \in \mathbb{R}^{n,n}$ es una matriz invertible, se define su matriz inversa como la matriz $A^{-1} \in \mathbb{R}^{n,n}$ tal que

$$A^{-1}A = AA^{-1} = I_n$$

Por tanto, se resuelve (cuando sea posible) el sistema cuadrado

$$A\mathbf{x} = \mathbf{b}$$

multiplicando por la inversa A^{-1} y obteniendo su solución $\mathbf{x}$ en la forma

$$A^{-1}A\mathbf{x} = I_n\mathbf{x} = \mathbf{x} = \mathbf{A^{-1}b}$$

El algoritmo para el cálculo de la inversa de una función se aplica en Matlab mediante el comando **inv.m**. Este algoritmo se basa en la factorización LU de una matriz, que se verá más adelante al definir el proceso de **Eliminación de Gauss** para la resolución de sistemas lineales.

A continuación se definirá el concepto de Dependencia e Indepenedencia Lineal entre vectores, lo que permite definir los Sistemas de Generadores y extraer de ellos unas bases. Gracias a la operaión de producto escalar se podrán caracterizar las bases ortogonales. Fijadas las bases se puede asociar a una matriz su única representación en términos de una aplicación lineal y definir los espacios imagen y núcleo que caracterizan la transformación.

Luego, se pasará a medir vectores y matrices mediante el concepto de norma y, finalmente, energía. Se presentarán las normas más comunmente utilizadas y la relación especial entre la norma euclídea y el producto escalar. Con la definición de algunos tipos y estructuras de matrices particularmente simples y la importante operación de trasposición se introducirán las matrices simétricas, las ortogonales y las definidas positivas. Esto permitirá definir y clasificar las formas cuadráticas atendiendo a su signo.

Para su clasificación se considera el problema de autovalores y la información fundamental que se obtiene al resolverlo. El conocimiento de los autovalores del problema será fundamental para determinar el condicionamiento de un sistema y permitirá saber si el sistema está bien planteado.

Tras introducir las Factorizaciones de Cholesky y QR se observará cómo resolver, mediante descomposición espectral, el problema de la diagonalización y se generalizará esta teoría al caso no cuadrado mediante la técnica de Descomposición en Valores Singulares.

A.1.4. Espacios Vectoriales

Un Espacio Vectorial es un conjunto no vacío de infinitos elementos, llamados vectores, *cerrado* con respecto a las operaciones de suma de vectores y multiplicación por un escalar.

Definición 4. *Sean* $\mathbf{u}$ *y* $\mathbf{v}$ *dos vectores genéricos del conjunto* V*, se dice que* V *es un Espacio Vectorial si*

$$\alpha\mathbf{u} + \beta\mathbf{v} \in V, \qquad \forall \alpha, \beta \in \mathbb{R}$$

La operación anterior recibe el nombre de Combinación Lineal y permite generar, mediante la variación de αyβ, todos los vectores del espacio.

Ejemplo 7. *Los espacios vectoriales en los que más suele trabajarse son los* $V = \mathbb{R}^n$*. Tienen dimensión finita* n*. Un ejemplo de espacio vectorial de dimensión finita es el espacio de las*

funciones continuas en un intervalo, denotado por $V = C(I)$. El conjunto de matrices $\mathcal{M}^{m,n}(K)$ es un espacio vectorial sobre el cuerpo K de los números reales o complejos.

Definición 5. *Si U y V son espacios vectoriales y $U \subset V$ entonces U es un subespacio vectorial.*

Ejemplo 8. *Sean $\mathbf{u}$, $\mathbf{v}$ dos vectores de $\mathbb{R}^3$. Si los vectores no son paralelos, es decir, no es múltiplo uno del otro, entonces generan un espacio U de dimensión 2 (un plano que pasa por el origen), que es un subespacio de $V = \mathbb{R}^3$. Si los vectores son paralelos, es decir, uno es múltiplo de otro, entonces generan un espacio U de dimensión 1 (una recta que pasa por el origen), que es un subespacio de $V = \mathbb{R}^3$.*

Ejemplo 9. *Dado un espacio vectorial definido por la ecuación*

$$V = (x_1, x_2, x_3) \in \mathbb{R}^3 / x_1 - 4x_2 - 5x_3 = 0$$

Para definirlo en términos de combinaciones lineales, explicitamos x_1 en la ecuación

$$x_1 = 4x_2 + 5x_3$$

y escribimos un vector genérico de V en la forma

$$(x_1, x_2, x_3) = (4x_2 + 5x_3, x_2, x_3) = \alpha(4, 1, 0) + \beta(5, 0, 1)$$

donde $\alpha = x_2, \beta = x_3$ son parámetros reales.

A.1.5. Dependencia e Independencia Lineal

La operación de Combinación Lineal permite definir el concepto de Dependencia e Independencia Lineal de un conjunto de vectores.

Definición 6. *Se dice que un conjunto de vectores $\{\mathbf{v_1}, ..., \mathbf{v_m}\}$ de $\mathbb{R}^n$ es Linealmente Dependiente si el sistema de n ecuaciones*

$$\sum_{i=1}^{m} \alpha_i \mathbf{v}_i = \alpha_1 \mathbf{v}_1 + ... + \alpha_m \mathbf{v}_m = \mathbf{0}$$

tiene únicamente la solución trivial $\mathbf{0}$, es decir, todos los coeficientes α_i son nulos:

$$\alpha_1 = ... = \alpha_m = 0$$

La operación de suma de vectores multiplicados por un escalar $\sum_{i=1}^{m} \alpha_i \mathbf{v}_i$ utilizada en la definición anterior es una Combinación Lineal de vectores y genera un vector. Por tanto, establecer si un conjunto de vectores es linealmente dependiente se reduce a demostrar que el sistema homogéneo anterior $\sum_{i=1}^{m} \alpha_i \mathbf{v}_i = \mathbf{0}$ sólo admite la solución trivial $\mathbf{0}$.

Observar que un sistema lineal homogéneo es siempre Compatible puesto que siempre tiene la solución trivial $\mathbf{0}$ entre sus soluciones, por definición de sistema homogéneo.

Ejemplo 10. *Sean los vectores $\mathbf{v}_1 = (1, -1, 0)^T, \mathbf{v}_2 = (0, 1, 2)^T$ y $\mathbf{v}_3 = (1, 0, 2)^T$ se pide estudiar si son Linealmente Independientes. Constrimos la combinación lineal*

$$\sum_{i=1}^{3} \alpha_i \mathbf{v}_i = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 = \mathbf{0}$$

y resolvemos el sistema homogéneo

$$\alpha_1 \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} + \alpha_3 \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

definido por las ecuaciones

$$\begin{cases} \alpha_1 + \alpha_3 & = 0 \\ -\alpha_1 + \alpha_2 & = 0 \\ 2\alpha_2 + 2\alpha_3 & = 0 \end{cases}$$

que tiene las infinitas soluciones

$$\bar{\alpha} = (\alpha_1, \alpha_2, \alpha_3)^T = (-\alpha_3, -\alpha_3, \alpha_3)^T \in L[(-1, -1, 1)]$$

luego, los tres vectores son Linealmente Dependientes.

Ejercicio 8. *Sean los vectores* $\mathbf{v}_1 = (1, 1, 0)^T, \mathbf{v}_2 = (0, 1, 1)^T$ *y* $\mathbf{v}_3 = (-1, 2, 1)^T$. *Estudiar si son Linealmente Independientes.*

Rango

Utilizando los conceptos de Dependencia e Independencia Lineal se puede definir el **rango de un sistema de vectores y de una matriz** de la siguiente forma:

Definición 7. *Sea* V *un espacio vectorial y* $\{u_1, ..., u_m\}$ *un sistema de vectores de* V. *Se denomina rango del sistema* $\{u_1, ..., u_m\}$ *a la dimensión del subespacio vectorial* $L[(u_1, ..., u_m)]$. *Para indicar que el rango del sistema es* r, *se denota*

$$rg(u_1, ..., u_m) = r.$$

[35]

Definición 8. *Sea* $A \in \mathbb{R}^{m,n}$. *Si* $\{A^1, ..., A^n\}$ *es el sistema de vectores de* $\mathbb{R}^{m,1}$ *formado por las columnas de* A, *se denomina rango de* A *al rango de dicho sistema de vectores. Luego*

$$rg(A) = rg(A^1, ..., A^n).$$

[35]

Es decir, el rango de una matriz es el número máximo de vectores columna de la matriz que son Linealmente Independientes.

Este número dará la dimensión del espacio imagen $\mathcal{R}(A)$ de la aplicación lineal asociada a la matriz. En general se tiene

Definición 9. *La Dimensión de un conjunto de vectores* S *viene dada por el rango de la matriz definida por los vectores de* S.

Definición 10. *Una matriz* $A \in \mathbb{R}^{m,n}$ *se dice de* ***Rango Máximo*** *si*

$$rg(A) = \text{mín}(m, n)$$

Una forma rápida de establcer la dependencia o independencia entre vectores consiste en generar la matriz con los vectores columna y aplicar el algoritmo de reducción a forma escalonada

por filas implementado en Matlab con el comando **rref(A)**. Este comando se basa en el proceso de Eliminación de Gauss que se verá en el contexto de la Factorización 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 ecuaciones).

Definición 11. *Una matriz se define en forma reducida por filas si las cabeceras unitarias están ordenadas (la cabecera de la fila superior aparece siempre a la izquierda de la cabecera de las filas inferiores).*

En cada columa aparece una única cabecera y los vectores columna con cabecera son linealmente independientes.

Ejercicio 9. *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 número máximo de vectores linealmente independientes.

A.1.6. Conjunto de Generadores, Bases y Bases Ortogonales

Conjunto de Generadores

Definición 12. *Sea G un conjunto de m vectores $G = \{\mathbf{v_1}, ..., \mathbf{v}_,\}$ de $\mathbb{R}^n$ siendo $m \leq n$. El subespacio $V \subset \mathbb{R}^n$ generado por $\{\mathbf{v_1}, ..., \mathbf{v}_,\}$ se denota por*

$$V = L[\mathbf{v_1}, ..., \mathbf{v_m}] \subset \mathbb{R}^n$$

y los infinitos vectores de V se generan mediante la operación de combinación lineal:

$$V = L[\mathbf{v_1}, ..., \mathbf{v_m}] = \{\mathbf{w} \in \mathbb{R}^n / \mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{v_i}, \alpha_i \in \mathbb{R}\} \subset \mathbb{R}^n$$

Si los vectores son Linealmente Independientes el espacio $V \subset \mathbb{R}^n$ generado es de dimensión m. Si los vectores son Linealmente Dependientes la simensión de V será $p < m$, siendo p el número máximo de vectores linealmente independientes del conjunto.

Ejemplo 11. *Dados dos vectores $\mathbf{u}_1 = (-1, 1, 0)^T, \mathbf{u}_2 = (0, 1, 1)^T$ de $\mathbb{R}^3$. Calcular las ecuaciones de subespacio $U = L[\mathbf{u}_1, \mathbf{u}_2]$ generado y determinar su dimensión.*

Se define un elemento genérico de subespacio U

$$\mathbf{v} = \alpha \mathbf{u}_1 + \beta \mathbf{u}_2, \qquad \alpha, \beta \in \mathbb{R}$$

En componentes se tiene el sistema de 3 ecuaciones lineales

$$v_1 = -\alpha, \qquad v_2 = \alpha + \beta, \qquad v_3 = \beta$$

Eliminando parámetros

$$\alpha = -v_1, \qquad \beta = v_3$$

se deduce la ecuación que define el subespacio (lineal):

$$U = \{\mathbf{v} = (v_1, v_2, v_3)^T / v_2 = v_3 - v1\} \subset \mathbb{R}^3$$

Al ser $U \subset \mathbb{R}^3$ un subespacio y tener que satisfacer una ecuación se deduce que la dimensión del subespacio U es dim(U)=3-1=2. Observando desde el comienzo que los dos vectores no son múltiplos se podía deducir que son independientes, luego generan un subespacio de dimensión 2.

Bases

Definición 13. *Sea V un espacio vectorial y sea $U \subset V$, se dice que un sistema $\{u_1, ..., u_m\}$ de vectores de U es una base de U si $u_1, ..., u_m$ son linealmente independientes y $\forall v \in U$ se verifica que v es combinación lineal de $u_1, ..., u_m$. [35]*

En otras palabras, una base B de un espacio vectorial V es un sistema de vectores de V linealmente independientes y generadores del espacio V.

Un Conjunto de Generadores consituido por m vectores no puede ser Linealmente Independiente (es decir, una base) si $m > n$. Por ejemplo, $m = 3$ vectores en un espacio de dimensión $n = 2$ no pueden ser linealmente independientes.

Suponiendo independencia lineal si $m = n$, entonces el conjunto de vectores $\mathbf{v_1}, ..., \mathbf{v_m}$ constituye una base del espacio $\mathbb{R}^m$. La Dependencia Lineal es una forma de redundancia en la información. En una base no hay redundancia y el número m de vectores independientes define la dimensión del espacio generado.

Bases Ortogonales

Definición 14. *Si el producto escalar entre dos vectores cualesquiera de una base es nulo, entonces la base es ortogonal:*

$$\langle \mathbf{v_i}, \mathbf{v_j} \rangle = \mathbf{v_i} \cdot \mathbf{v_j} = 0, i \neq j$$

Si además,

$$\langle \mathbf{v_i}, \mathbf{v_j} \rangle = 1, \forall i$$

la base es ortonormal.

A continuación, veremos cómo construir fácilmente una base de un espacio vectorial a partir de un conjunto de generadores. Para ello, se utiliza el algoritmo de eliminación de Gauss.

Formas escalonadas reducidas por filas

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

Ejemplo 12. *Sean los vectores*

$$\mathbf{u}_1 = (4, -5, 7)^T, \qquad \mathbf{u}_2 = (2, -4, 4)^T, \qquad \mathbf{u}_3 = (1, 1, -2)^T, \qquad \mathbf{u}_4 = (2, -1, 1)^T$$

Determinar el espacio generado, es decir $L[(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4)]$. *Extraer un base de* $\mathbb{R}^3$ *usando la forma de matriz escalonada por filas.*

Utilizando la siguiente sintaxis se define la matriz (en formato racional)

Código A.8: Definir 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 Rerducida por Filas se obtiene mediante

Código A.9: Obtener matriz escalonada lineal

```
A_red=rref(A)
```

y se deduce que los primeros tres vectores de $\mathbb{R}^3$ son independientes, luego

$$L[(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4)] = \mathbb{R}^3.$$

Para determinar la combinación lineal que permite escribir $\mathbf{u}_4$ mediante $\mathbf{u}_1$ y $\mathbf{u}_2, \mathbf{u}_3$ (coeficientes o pesos de la combinación lineal) se extrae la última columna de la matriz escalonada

Código A.10: Determinar combinación lineal

```
c=A_red(:,4)
```

Sistemas de coordenadas

Una vez introducidos los conceptos de base y base ortogonal se puede definir el concepto de **coordenadas** en una base. Sea $\mathbf{x} = (x_1, ..., x_n)^T$ un vector de $\mathbb{R}^n$ donde $(x_1, ..., x_n)$ son las coordenadas en la base canónica $\mathbf{e}_i = (0, ., 1., , 0)$, con $\mathbf{e}_{i,i} = 1$ y $\mathbf{e}_{i,j} = 0$. Sea $\mathcal{B} = \{_1, ..., \mathbf{v}_n\}$ una base de $\mathbb{R}^n$. Entonces si se definde $B = (\mathbf{v}_1|...|\mathbf{v}_n)$ como la matriz asociada a la base $\mathcal{B}$, las nuevas coordenadas de $\mathbf{x}$ en la base $\mathcal{B}$ vienen dadas por la (única) solución del sistema lineal

$$B\mathbf{y} = \mathbf{x}$$

que es

$$\mathbf{y} = B^{-1}\mathbf{x}$$

Esto quiere decir que $P = B^{-1}$ es una matriz de peso o cambio de coordenadas del sistema canónico a un sistema de coordenada definido por $\mathcal{B}$. Este razonamiento se puede extender para pasar de un sistema de coordenadas asociado a la base $\mathcal{B}$ a uno asociado a la base $\mathcal{B}'$.

Ejemplo 13. *Dadas las bases*

$$\mathcal{B}_C = \{(1, 0)^T, (0, 1)^T\}, \qquad \mathcal{B}_1 = \{(1, 0)^T, (1, 1)^T\}, \qquad \mathcal{B}_2 = \{(-1, 1)^T, (0, 2)^T\}$$

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

$$P(\mathcal{B}_C, \mathcal{B}_1), \qquad P(\mathcal{B}_1, \mathcal{B}_2), \qquad P(\mathcal{B}_2, \mathcal{B}_C)$$

Suponer tener un dato $\mathbf{x}_1 = (2, 2,)^T$ *en la base canónica. Calcular su coordenadas en las bases* $\mathcal{B}_1 y \mathcal{B}_2$.

Escribiendo las matrices en forma matricial se tiene

$$B_C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad B_2 = \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix}$$

Para calcular la primera matriz de paso

$$P_1 = P(\mathcal{B}_C, \mathcal{B}_1) = (\mathbf{p}_1 | \mathbf{p}_2)$$

hay que escribir los vectores de $\mathcal{B}_C$ en términos de los vectores de $\mathcal{B}_1$. Esto equivale a resolver los dos sistemas

$$B_1 \mathbf{p}_1 = \mathbf{e_1}, \qquad B_1 \mathbf{p}_2 = \mathbf{e_2},$$

que se pueden escribir en forma compacta $B_1 P_1 = I_d$, luego $P_1 = B_1^{-1}$.

Para calcular la matriz de paso

$$P_2 = P(\mathcal{B}_1, \mathcal{B}_2) = (\mathbf{p}_1 | \mathbf{p}_2)$$

hay que escribir los vectores de $\mathcal{B}_1$ en términos de los vectores de $\mathcal{B}_2$. Esto equivale a resolver los dos sistemas

$$B_2 \mathbf{p}_1 = \mathbf{b}_{11}, \qquad B_2 \mathbf{p}_2 = \mathbf{b}_{12},$$

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

$$P_3 = P(\mathcal{B}_2, \mathcal{B}_C) = (\mathbf{p}_1 | \mathbf{p}_2)$$

hay que escribir los vectores de $\mathcal{B}_2$ en términos de los vectores de $\mathcal{B}_C$. Esto equivale a resolver los dos sistemas

$$B_C \mathbf{p}_1 = \mathbf{b}_{21}, \qquad B_C \mathbf{p}_2 = \mathbf{b}_{22},$$

que se pueden escribir en forma compacta $B_C P_3 = B_2$, luego $P_3 = B_2$.

Se tiene

$$P_1 = B_1^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \qquad P_2 = B_2^{-1} B_1 = \begin{pmatrix} -1 & -1 \\ 1/2 & 1 \end{pmatrix},$$

siendo

$$P_3 = B_2 = \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix}$$

Estas matrices permiten obtener las coordenadas en las otras bases mediante proyección. Dado, por ejemplo, $\mathbf{x}_1 = (2, 2)^T$ en la base canónica si se quieren conocer sus coordenadas $\mathbf{y}_{B1}$

en B_1 se hace la proyección

$$\mathbf{y}_{B1} = P_1\mathbf{x}_1 = (0,2)^T$$

Para conocer sus coordenadas $\mathbf{y}_{B2}$ en B_2 se hace la proyección

$$\mathbf{y}_{B2} = P_2P_1\mathbf{x}_1 = P_2\mathbf{y}_{B1} = (-2,2)^T$$

Ejercicio 10. *Dadas las bases*

$$\mathcal{B}_V = \{(1,0)^T, (1,1)^T\}, \qquad \mathcal{B}_N = \{(-1,1)^T, (0,2)^T\},$$

Sea $\mathbf{u} = (3,7)^T$ *un vector expresado en términos de la base canónica* $\mathcal{B}_C = \{(1,0)^T, (0,1)^T\}$. *Calcular las coordenadas* $\mathbf{x}$ *de* $\mathbf{u}$ *en la base* $\mathcal{B}_V$. *Definir la matriz de paso* $P = P(\mathcal{B}_V, \mathcal{B}_N)$ *y calcular las coordenadas* $\mathbf{y}$ *de* $\mathbf{u}$ *en la base* $\mathcal{B}_N$.

Bases Ortogonales

Una base ortogonal implica que un vector genérico $\mathbf{w} \in \mathbb{R}^n$ se expresa de manera única en esta base mediante sus componentes:

$$\mathbf{w} = \sum_{i=1}^{n} w_i\mathbf{v}_i$$

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

$$\langle \mathbf{w}, \mathbf{v}_i \rangle = \mathbf{x} \cdot \mathbf{v}_i = \sum_{i=1}^{n} w_i\mathbf{v}_i \cdot \mathbf{v}_i = w_i||\mathbf{v}_i||^2 = w_i$$

Si la base es únicamente ortogonal la fórmula es

$$w_i = \frac{\langle \mathbf{w}, \mathbf{v}_i \rangle}{||\mathbf{v}_i||^2} \in \mathbb{R}$$

Bases Ortonormales

Las bases ortonormales son bases ortogonales cuyos vectores tienen norma unitaria. La base canónica es el ejemplo más utilizado de base ortogonal. Mediante el algoritmo de ortonormalización de Gram-Schmidt de cualquier base se puede extraer una base ortogonal. Este algoritmo se basa en la factorización QR de una matriz que se verá más adelante.

A.1.7. Matrices asociadas a Transformaciones Lineales

Las matrices $A \in \mathbb{R}^{m,n}$ pueden identificarse con Transformaciones Lineales definidas por la actuación que tiene la operación multiplicación matriz por vector sobre todos os vectores de un espacio vectorial. En concreto se tiene:

Definición 15. *Dadas las bases canónicas de los espacios vectoriales* $\mathbb{R}^m y \mathbb{R}^n$ *y dada una matriz* $A \in \mathbb{R}^{m,n}$. *Entonces existe una única Transformación Lineal definida, mediante las báses*

canónicas, en la forma

$$T(\mathbf{x}) = \mathbf{Ax} \in \mathbb{R}^{\mathbf{m}}, \forall \mathbf{x} \in \mathbb{R}^{\mathbf{n}}$$

La aplicación construida, $T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $T(\mathbf{x}) = \mathbf{Ax}$ es lineal ya que verifica la propiedad

$$T(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha\mathbf{T}(\mathbf{x}) + \beta\mathbf{T}(\mathbf{y}), \forall \alpha, \beta \in \mathbb{R}$$

Ejemplo 14. *Dada la aplicación $T : \mathbb{R}^2 \to \mathbb{R}^2$ definida por*

$$T(x, y) = (x - y, x^2)$$

Demostrar que no es lineal. Para ello, sean $\mathbf{u} = (u_1, u_2)^T, \mathbf{v} = (v_1, v_2)^T$ y $\alpha, \beta \in \mathbb{R}$. Se tiene

$$T(\alpha\mathbf{u} + \beta\mathbf{v}) = (\alpha u_1 + \beta v_1 - \alpha u_2 - \beta v_2, (\alpha u_1 + \beta v_1)^2)$$

Por otra parte

$$\alpha T(\mathbf{u}) + \beta T(\mathbf{v}) = (\alpha u_1 + \beta v_1 - \alpha u_2 - \beta v_2, (\alpha u_1^2 + \beta v_1^2)$$

luego la aplicación no es lineal, ya que las segundas componentes son, en general, distintas:

$$(\alpha u_1 + \beta v_1)^2) \neq \alpha u_1^2 + \beta v_1^2$$

Por construcción, una aplicación lineal siempre manda el cero al cero, es decir,

$$T(\mathbf{0}) = \mathbf{0}$$

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

Ejercicio 11. *Estudiar si las siguentes aplicaciones son lineales:*

$$f(x, y) = (x - y, x + 1), g(x, y) = (x - y, x)$$

Respuesta: Se demuestra, aplicando la definición, que $g(x, y)$ es lineal. Basta observar que es la aplicación lineal asociada a la matriz

$$\begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix}$$

Por otra parte, $f(0, 0) = (0, 1) \neq (0, 0)$, luego f no es lineal. Se verifica que f es una aplicación afín:

$$f(x, y) = \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

A.1.8. Espacio Imagen y Núcleo

Dada una matriz $A \in \mathbb{R}^{m,n}$ se puedn definir dos subespacios vectoriales que caracterizan el comportamiento de la matriz y de la aplicación lineal asociada. Observar que dada una matriz siempre se puede considerar la aplicación lineal asociada $f : \mathbb{R}^n \to \mathbb{R}^m$ definida por

$$f(\mathbf{x}) = \mathbf{Ax}$$

Definición 16. *Sea $A \in \mathbb{R}^{m,n}$ una matriz y $f : \mathbb{R}^n \to \mathbb{R}^m$ su aplicación lineal asociada. Se*

define el espacio imagen de $f, im(f)$, al subconjunto de $\mathbb{R}^m$ dado por

$$Im(f) = \mathcal{R}(A) = \{\mathbf{y} = \mathbf{Ax} \in \mathbb{R}^m | \mathbf{x} \in \mathbb{R}^n\} \subset \mathbb{R}^m$$

El conjunto de vectores que pertenecen a la imagen de f es un subespacio vectorial de $\mathbb{R}^m$ cuya dimensión viene dada por el rango de A.

Definición 17. *Sea $A \in \mathbb{R}^{m,n}$ una matriz y $f : \mathbb{R}^n \to \mathbb{R}^m$ su aplicación lineal asociada. Se define el espacio del núcleo de $f, ker(f)$, al subconjunto de $\mathbb{R}^n$ dado por*

$$Ker(f) = \mathcal{N}(A) = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{Ax} = \mathbf{0} \in \mathbb{R}^m\} \subset \mathbb{R}^n$$

El conjunto de vectores que pertenecen al Núcleo (Kernel) de f es un subespacio vectorial de $\mathbb{R}^n$.

La imagen de la transformación lineal es el espacio imagen

$$Im(f) = \mathcal{R}(A)$$

y viene dado por todos los vectores $\mathbf{y} \in \mathbb{R}^m$ tales que el sistema $\mathbf{y} = \mathbf{Ax}$ es compatible. El espacio imagen viene generado por los vectores imagen de la base canónica.

El núcleo de la transformación lineal es el espacio

$$Ker(f) = \mathcal{N}(A)$$

y viene definido por las soluciones del sistema de ecuaciones lineales

$$f(\mathbf{x}) = \mathbf{Ax} = \mathbf{0}_m$$

Para estudiar los sistemas lineales puede utilizarse la **Fórmula de la dimensión** que explica la relación entre los subespacios imagen y núcleo:

Teorema 6. *Sea $A \in \mathbb{R}^{m,n}$ una matriz y $f : \mathbb{R}^n \to \mathbb{R}^m$ su aplicación lineal asociada. Se tiene entonces*

$$n = dim\mathcal{R}(A) + dim\mathcal{N}(A)$$

Observar que $dim\mathcal{R}(A) \leq m$ porque es un subespacio. Lo mismo ocurre con el espacio núcleo de la aplicación: $dim\mathcal{N}(A) \leq n$.

Definición 18. *Sea $A \in \mathbb{R}^{m,n}$ una matriz y $f : \mathbb{R}^n \to \mathbb{R}^m$ la aplicación lineal asociada. Si $\mathcal{N}(A) = \{\mathbf{0}\}$ entonces se dice que la aplicación lineal es **inyectiva**. Si $\mathcal{R}(A) = \mathbb{R}^m$ se dice que la aplicación lineal es **suprayectiva**. Una aplicación lineal inyectiva y suprayectiva se dice **biyectiva** o isomorfismo.*

En Matlab existen los comandos **null(A)** y **orth(A)** que permiten calcular una base ortonormal de los subespacios vectoriales $\mathcal{N}(A)$ y $\mathcal{R}(A)$ respectivamente.

Ejercicio 12. *Dada la matriz*

$$\begin{pmatrix} 1 & -2 & 0 \\ 0 & 0 & 0 \\ 2 & -4 & 0 \end{pmatrix}_{3,3}$$

Definir la aplicación lineal asociada. Calcular la dimensión imagen $\mathcal{R}(A)$ y del núcleo $\mathcal{N}(A)$. Calcular una base de los mismos. Determinar si la aplicación es inyectiva, suprayectiva o biyectiva.

Ejercicio 13. *Dada la matriz*

$$\begin{pmatrix} 2 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix}_{3,2}$$

Definir la aplicación lineal asociada. Calcular la dimensión imagen $\mathcal{R}(A)$ y del núcleo $\mathcal{N}(A)$. Calcular una base de los mismos. Determinar si la aplicación es inyectiva, suprayectiva o biyectiva.

A.1.9. Normas

La norma generaliza el concepo de módulo de un vector y permite medir matrices y distancias entre ellas. Se puede, por tanto, medir la distancia entre dos imágenes o calcular el residuo o error cometido al realizar numéricamente una descomposición o una factorización. Las normas aparecen en la definición de los funcionales de energía en cálculo variacional y en las funciones de pérdida en aprendizaje automático. A continuación se verán las *normas p*, que son las normas asociadas a los espacios de Lebesgue. En el contexto del cálculo numérico discreto, se tiene:

Definición 19. *La norma p de un vector $\mathbf{x} \in \mathbb{R}^n$ se calcula mediante la forma*

$$||\mathbf{x}||_p = (\sum_i |x_i|^p)^{1/p}, \qquad \forall p \in \mathbb{R}, p \geq 1$$

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

Definición 20. *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función (campo escalar) que verifica las siguientes propiedades:*

1. $f(x) = 0 \iff x = 0 \in \mathbb{R}^n$
2. $f(x + y) \leq f(x) + f(y), \qquad \forall x, y \in \mathbb{R}^n$
3. $f(\alpha x) = |\alpha| f(x), \qquad \forall \alpha \in \mathbb{R}, \qquad \forall x \in \mathbb{R}^n$

entonces f es una norma.

Para $p = 2$ se tiene la clásica norma Euclídea o norma 2, denotada normalmente sin el subíndice.

$$||\mathbf{x}||_2 = ||\mathbf{x}|| = (\sum_i |x_i|^2)^{1/2}$$

Para $p = 1$ se tiene la norma 1

$$||\mathbf{x}||_1 = (\sum_i |x_i|)$$

Para $p = \infty$ se tiene la norma ∞ o norma del máximo

$$||\mathbf{x}||_\infty = \max_i |x_i|$$

En Matlab el comando **norm** calcula la norma de una matriz o un vector. Concretamente, se tiene:

$$||\mathbf{x}||_2 = \mathbf{norm}(x, 2) = \mathbf{norm}(x)$$
$$||\mathbf{x}||_1 = \mathbf{norm}(x, 1)$$
$$||\mathbf{x}||_\infty = \mathbf{norm}(x, Inf)$$

Un vector es unitario si

$$||\mathbf{x}|| = 1$$

El conjunto de todos los vectores unitarios define la Bola o 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 métrica o distancia.

Definición 21. *Dados dos vectores* $\mathbf{x}, \mathbf{y}$ *en el espacio vectorial* $V = \mathbb{R}^n$*. La distancia entre ellos se define por*

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$$

de donde la longitud de un vector, que es la distancia del punto (vector) al origen, se define como

$$d(\mathbf{x}, \mathbf{0}) = ||\mathbf{x}||$$

A menudo se utiliza la norma al cuadrado de un vector, lo que representa la **energía** del vector.

Definición 22. *La energía de un vector se calcula a través del producto escalar del vector consigo mismo:*

$$\langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v} \cdot \mathbf{v} = ||\mathbf{v}||^2$$

Generalizando, se puede escribir la operación de producto escalar mediante la definición de norma:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u} = ||\mathbf{u}||_2||\mathbf{v}||_2 cos(\theta) \in \mathbb{R}$$

Para medir una matriz $A \in \mathbb{R}^{m,n}$ y definir una distancia en el espacio de las matrices, en aprendizaje automático se utiliza a menudo la norma de Frobenius definida por

$$||A||_F = (\sum_{i,j} |a_{i,j}|^2)^{1/2}$$

Observar que la norma de Frobenius es la norma euclídea del vector generado concatenando las componentes de la matriz como un vector mn dimensional. Se tienen las siguientes afirmaciones de las normas matriciales más utilizadas

$$||A||_1 = \max_j (\sum_{i=1}^{m} |a_{i,j}|),$$

$$||A||_\infty = \max_i(\sum_{j=1} |a_{i,j}|),$$

$$||A||_2 = \max_{x\leq 0, y\leq 0} \frac{Ax \cdot y}{||x||_2||y||_2}$$

En Matlab se utilizan los siguientes comandos:

$$||A||_2 = \mathbf{norm}(A, 2) = \mathbf{norm}(A)$$

$$||A||_1 = \mathbf{norm}(A, 1)$$

$$||A||_\infty = \mathbf{norm}(A, Inf)$$

$$||A||_{fro} = \mathbf{norm}(A, fro)$$

Ejercicio 14. *Dadas dos imágenes. Contaminar utilizando modelo ruido gaussiano con parámetros* $\lambda = 1, \lambda = 0{,}1 y \lambda = 0{,}01$. *Calcular la distancia de Frobenius entre las imágenes contaminadas y la original. Calcular la distancia entre las imágenes originales.*

Ejercicio 15. *Dada una imagen en color. Pasarla a doble precisión en el intervalo* $[0, 1]$ *y a una imagen en escala de grises mediante el comando* ***rgb2gray.m****. Calcular sus normas* $1, 2, \infty$ *y la de Frobenius. Calcular la energía*

$$E = \frac{1}{2}||\mathbf{x}||_2^2 = \frac{1}{2}\sum_i |x_i|^2$$

A.1.10. Tipos especiales de matrices y formas cuadráticas

Algunos tipos especiales de matrices son muy útiles computacionalmente debido a su estructura. Las **matrices diagonales** $D \in \mathbb{R}^(n, n)$ se caracterizan por las ecuaciones $a_{i,j}$ si $i \neq j$. La **matriz nula** es una matriz diagonal con ceros en su diagonal. En Matlab se genera con el comando **zeros(n)**. La **matriz identidad** es una matriz digonal, generada mediante el comando **eye(n)**. Dado un vector $\mathbf{v} \in \mathbb{R}^n$ se define por **diag(v)** a la matriz cuadrada cuya 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 formado por los elementos de la diagonal. Multiplicar un vector por una matriz diagonal es computacionalmente eficiente

$$diag(\mathbf{v})\mathbf{x} = \mathbf{v} \odot \mathbf{x}$$

y corresponde a rescalar cada componente x_i por el escalar v_i. Es eficiente también invertir una matriz diagonal

$$diag(\mathbf{v})^{-1} = diag([1/v_1, ..., 1/v_n])$$

lo que sólo es posible si $v_i \neq 0, \forall i$.

Las matrices triangulares inferiores y superiores que tienen ceros por arriba y por abajo, respectivamente, de la diagonal principal son fundamentales. Los comandos que extraen de una matriz A su parte triangular inferior y superior son **tril**(A) y **triu**(A), respectivamente. Si la matriz no es cuadrada, los mismos comandos dan como resultado matrices trapezoidales.

La **matriz simétrica** definida mediante la operación de trasposición es otra estrucutra muy importante.

Definición 23. *Sea* $A = (a_{i,j})$ *una matriz cuadrada. Se dice que* A *es* ***simétrica*** *si* $a_{i,j} = a_{j,i}$,

es decir, si se verifica la ecuación matricial

$$A^T = A$$

Otras de las matrices cuya estrucutra resulta muy relevante son las **matrices ortogonales**, cuyos vectores filas (respect. columna) son mutuamente ortogonales. Si además tienen norma unitaria, la matriz es ortonormal.

Definición 24. *Sea $A = (a_{i,j})$ una matriz cuadrada. Se dice que A es **ortogonal** si $a_{i,j} = a_{j,i}$, es decir, si se verifica la ecuación matricial*

$$A^T A = AA^T = I_n$$

Una propiedad importante que se deduce a partir de la unicidad de la matriz inversa es que una matriz ortogonal puede invertirse simplemente trasponiendo sus componentes:

$$A^{-1} = A^T$$

Formas Cuadráticas

En primer lugar se define la forma cuadrática asociada a una matriz cuadrada.

Definición 25. *Sea $A \in \mathbb{R}^{n,n}$ una matriz cuadrada. Se dice que la forma cuadrática $q : \mathbb{R}^n \to \mathbb{R}$ definida por*

$$q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$$

está asociada a la matriz A.

Equivalentemente, puede escribirse:

$$q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} = \sum i = 1^n \sum j = 1^n A_{i,j} x_i x_j$$

Se pueden ahora definir las matrices Definidas Positivas mediante el signo de la forma cuadrática asociada.

Definición 26. *Sea $A \in \mathbb{R}^{n,n}$. Se dice que A es **Definida Positiva** si la forma cuadrática $q : \mathbb{R}^n \to \mathbb{R}$ definida por $q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$ verifica:*

$$q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} > 0, \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

Se obtienen definiciones análogas para matrices **Semi-Definidas Positivas**, que son siempre no negativas

$$\mathbf{x}^T A\mathbf{x} \geq 0, \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

Por otro lado, las matrices **Definidas Negativas** se caracterizan por satisfacer la desigualdad

$$\mathbf{x}^T A\mathbf{x} < 0, \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

y las **Semi-Definidas Negativas** se obtienen al relajar la desigualdad anterior

$$\mathbf{x}^T A\mathbf{x} \leq 0, \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

Finalmente, las matrices **Indefinidas** se obtienen cuando ninguna de las desigualdades anteriores es cierta.

La matriz que define una forma cuadrática no tiene porqué ser simétrica. Sin embargo, no hay pérdida de generalidad si se considera que es simétrica. Esto es así ya que

$$\langle \mathbf{x}, A\mathbf{x} \rangle = \mathbf{x}^T A\mathbf{x} = \mathbf{x}^T B\mathbf{x} = \langle \mathbf{x} = B\mathbf{x} \rangle$$

si

$$B = \frac{1}{2}(A + A^T),$$

es decir, si B es la parte simétrica de A.

La notación de producto escalar introducida en la igualdad $\mathbf{x}^T A\mathbf{x} = \mathbf{x}^T B\mathbf{x} = \langle \mathbf{x}$ permite obtener este resultado utilizando la propiedad de simetría del producto escalar (una función bilineal simétrica).

En general, no es fácil determinar el carácter de una forma cuadrática excepto en algunos casos más evidentes. Sin embargo, el cálculo de los autovalores de la matriz proporciona un criterio muy simple. De momento, se va a considerar la interpretación gráfica de estos conceptos mediante Matlab u Octave.

Ejemplo 15. *Sean dadas las formas cuadráticas*

$$q_1(x, y) = 2x^2 + 4xy + 2y^2, q_2(x, y) = -2x^2 + 4xy - 2y^2,$$

$$q_3(x, y) = -2x^2 + 4xy + 2y^2, q_4(x, y) = 2x^2 + 4y^2$$

Clasificarlas utilizando métodos gráficos.

Estas consideraciones serán esenciaes a la hora de optimizar formas cuadráticas o funciones aproximadas localmente por ellas mediante los Desarrollos de Taylor de orden II. Las gráficas de las formas cuadráticas se pueden obtener con los comandos:

Código A.11: Obtener gráficas de las formas cuadráticas

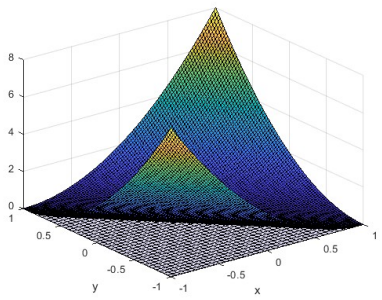
```
A = [-2 4; 0 2]; B = (1/2)*(A+A');
syms x y real
X = [x y]';
q = expand(X'*B*X);
Z = 0*x+0*y %plano de ecuacion z = 0
figure
fsurf(q,[-1,1,-1,1],'MeshDensity',100)
hold on
fsurf(Z,[-1,1,-1,1],'FaceAlpha',0.2,'MeshDensity',40)
xlabel('x'),ylabel('y')
```

donde se ha simetrizado la matriz de las formas cuadráticas. A través del siguiente código se pueden definir las dos variables reales (x, y), almacenarlas en un vector $X = (x, y) \in \mathbb{R}^2$ y calcular explícitamente la forma cuadrática.

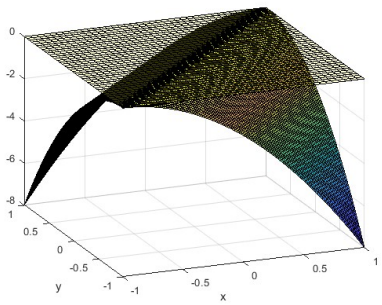
Código A.12: Calcular forma cuadrática

```
syms x y real
q = expand(X'*B*X)
```

Como se quiere representar un campo escalar $q = (x, y)$ de varias variables (2 en este caso), se utiliza para su graficación el comando **fsurf(q,[-1,1,-1,1], 'MeshDensity',100)** en un dominio definido $D_q = [-1, 1, -1, 1]$ y con una denisidad de puntos prefijada. El comando **FaceAlpha** confiere transparencia a la gráfica del plano para estudiar el signo de la forma cuadrática.



(a) $q_1(x, y) = 2x^2 + 4xy + 2y^2$



(b) $q_2(x, y) = -2x^2 + 4xy - 2y^2$

Figura A.6: Se verifica gráficamente que $q_1(x, y)$ y $q_2(x, y)$ son definida positiva y definida negativa, respectivamente, puesto que toman valores por encima del plano de ecuación z = 0 y por debajo, respectivamente.

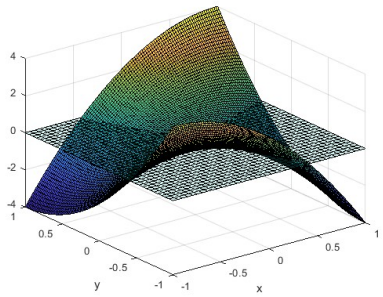


Figura A.7: Se verifica gráficamente que $q_3(x, y)$ es una forma cuadrática indefinida puesto que toma valores por encima y por debajo del plano de ecuación z = 0. Observa que esta función no tiene mínimos ni máximos.

A.1.11. Autovalores y autovectores

Definición 27. *Dada una matriz cuadrada $A \in \mathbb{R}^{m,n}$ el Problema de autovalores consiste en encontrar un número escalar $\lambda \in \mathbb{R}$ (que puede ser complejo) y un vector no nulo $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$ tales que*

$$A\mathbf{x} = \lambda\mathbf{x}$$

*donde λ es **autovalor** y **x autovector**. El conjunto de autovalores de una matriz se llama **espectro** y se denota por*

$$\sigma(A) = \lambda_1, ..., \lambda_n$$

*La cantidad $\rho(A)$ es el **Radio Espectral** definido por el autovalor módulo máximo:*

$$\rho(A) = \max_i\{|\lambda i|\}$$

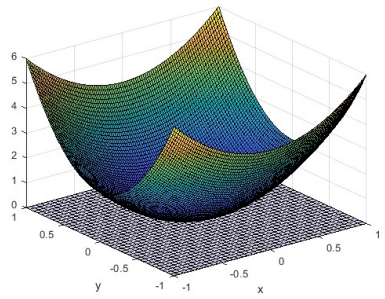


Figura A.8: Se verifica gráficamente que $q_4(x, y)$ es una forma cuadrática definida positiva ya que toma valores por encima o iguales a los del plano de ecuación z = 0. Observa que esta función tiene un único punto de mínimo absoluto.

Los autovectores definen **direcciones invariantes** de la trasformación lineal asociada a la matriz. Si **x** es autovector entonces $\alpha\mathbf{x}, \alpha \neq 0$ es autovector. Puede existir el autovalor nulo $\lambda = 0$, pero, por definición, no existe el autovector nulo. Los autovalores se pueden calcular resolviendo la **ecuación característica** (polinómica) de grado n:

$$p_A(\lambda) = det(A - \lambda I) = 0 \tag{A.1}$$

La teoría de los sistemas lineales permitirá deducir la ecuación A.1 para la caracterización de los autovalores. De momento se observa que la ecuación A.1 es **no lineal** y no existen fórmulas analíticas exactas para su resolución cuando el grado del polinomio es elevado. En general, las ecuaciones no lineales se resolverán mediante algoritmos numéricos que darán una solución aproximada del problema. Sin embargo, al ser polinómica existen resultados teóricos sobre el número y signo de sus raíces reales, que serán utilizados a continuación. Por el **Teorema Fundamental del Álgebra** se tiene:

Teorema 7. *Toda matriz cuadrada de orden n con coeficientes complejos tiene n autovalores, reales o complejos, iguales o distintos.*

Definición 28. *El número de veces que un autovalor λ aparece como raíz del polinomio característico es la **multiplicidad algebraica** y se denota por $m_A(\lambda)$*

Asociado a cada autovalor hay un **autoespacio** definido por:

Definición 29. *Sea A una matriz cuadrada y sea λ un autovalor de A. El autoespacio asociado al autovalor se define mediante el núcleo de la aplicación lineal asociada a la matriz $A - \lambda I$, siendo I la matriz identidad. En fórmulas*

$$E(\lambda) = \mathcal{N}(A - \lambda I) \subset \mathbb{R}^n$$

Para calcular los autovectores que están en cada autoespacio hay que calcular una base del subespacio $E(\lambda) = \mathcal{N}(A - \lambda I)$.

Definición 30. *La dimensión de cada autoespacio,*

$$dim(E(\lambda)) = dim(\mathcal{N}(A - \lambda I))$$

*da la **multiplicidad geométrica** $m_G(\lambda)$ del autovalor.*

Siempre se tiene que

$$m_G(\lambda) \leq m_A(\lambda)$$

Es decir, la multiplicidad geométrica nunca puede ser mayor que la multiplicidad algebraica.

Ejemplo 16. *Sea la matriz*

$$A = \begin{pmatrix} 1 & 2 & 2 \\ -1 & 4 & 1 \\ -2 & 2 & 5 \end{pmatrix}_{3,3}$$

Calcular su espectro y la multiplicidad algebraica y geométrica de cada autovalor. Encontrar explícitamente las ecuaciones que definen a los autoespacios. Determinar su factorización espectral

$$AV = VD$$

Hay dos autoespacios asociados a los autovalores $\lambda_1 = 3$ (de multiplicidad algebraica 2) y $\lambda_2 = 4$. Se tiene

$$E(\lambda_1) = E(3) = (x, y, z)/x = y + z,$$

de dimensión 2, luego la multiplicidad geométrica es 2. Por otro lado,

$$E(\lambda_2) = E(4) = (x, y, z)/x = 2y, z = 2y$$

tiene dimensión 1, luego la multiplicidad geométrica es 1. Hay diagonalizabilidad.

El cálculo de autovalores se realiza mediante un algoritmo iterativo que está basado en la factorización QR (que se presenta más adelante) obtenida utilizando el algoritmo de ortonormalización de Gran-Schmidt. En Matlab el método QR está implementado con la función **eig**. La sintaxis es: D = **eig**(A) para obtener un vector D que contiene todos los autovalores de A. En la forma

$$[V, D] = \mathbf{eig}(A)$$

se obtiene también la matriz V cuyas columnas son los autovectores de A y que satisface la relación

$$AV = VD$$

denominada **Factorización espectral** para la matriz cuadrada A.

A.1.12. Diagonalizabilidad

El conocimiento del espectro de una matriz permite resolver el problema de la diagonalizabilidad. No todas las matrices cuadradas son diagonalizables.

Definición 31. *Sea A una matriz cuadrada de prden n, $A \in \mathbb{R}^{n,n}$. Se dice que A es **diagonalizable** si **existe** una matriz cuadrada no singular V tal que*

$$V^{-1}AV = D = diag(\lambda_1, ..., \lambda_n)$$

La matriz V se construye almacenando los autovectores que se corresponden con los autovalores que están en la diagonal de la matriz D siguiendo exactamente el mismo orden. En concreto, sea λ_i un autovalor de multiplicidad algebraida m_A. Sus autovectores asociados se determinarán resolviendo el sistema cuadrado homogéneo

$$B(\lambda_i \mathbf{x} = (A - \lambda_i I)\mathbf{x} = \mathbf{0} \tag{A.2}$$

e imponiendo la existencia de infinitas soluciones del sistema A.2. La dimensión del espacio de soluciones es la multiplicidad geométrica m_G del autovalor.

Teorema 8. *Sea A una matriz cuadrada de orden n, $A \in \mathbb{R}^{n,n}$. Una* ***condición suficiente*** *para la diagonalizabilidad es que la matriz tenga n autovalores reales y distintos.*

Esto es porque autovalores distintos tienen autovectores linealmente independientes.

Ejercicio 16. *Sea la matriz*

$$A = \begin{pmatrix} -2 & -2 & 3 \\ -5 & 1 & 3 \\ -6 & 0 & 5 \end{pmatrix}_{3,3}$$

Demostrar que es diagonalizable calculando las matrices V y D tales que

$$A = VDV^{-}1$$

Teorema 9. *Una* ***condición necesaria y suficiente*** *para la diagonalizabilidad es que*

$$m_A = m_G$$

El teorema afirma que la multiplicidad algebraica m_A de cada autovalor (número de veces que aparece como raíz del polinomio característico) coincida con su multiplicidad geométrica m_G (dimensión del autoespacio). Se dirá 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 10. *Toda matriz A de dimensión n y coeficientes reales es diagonalizable si, y sólo si, existe una base de $\mathbb{R}^n$ formada por autovectores.*

Ejemplo 17. *Demostrar que la matriz A es diagonalizable siendo*

$$A = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{pmatrix}_{3,3}$$

La matriz es simétrica y tiene sólo 2 autovalores distintos:

$$\lambda_1 = -2, \lambda_2 = 2$$

siendo el segundo de ellos de multiplicidad algebraica $m_2 = 2$. Los autovectores asociados al segundo autovalor son del tipo

$$\mathbf{v}_2 = (a, b, a)^T = a(1, 0, 1) + b(0, 1, 0)$$

luego, hay diagonalizabilidad puesto que el auto-espacio asociado tiene dimensión 2, ela multiplicidad geométrica de λ_2.

Ejemplo 18. *Establecer si la matriz A es diagonalizable siendo*

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 2 & 2 \\ -1 & 1 & 3 \end{pmatrix}_{3,3}$$

La matriz no es diagonalizable. Calculando los autovalores (a mano o con el algortimo **eig** de Matlab) se obtiene

$$\lambda_1 = 1, \lambda_2 = 4$$

y se observa en la matriz D

$$D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{3,3}$$

que el primer autovalor, $\lambda_1 = 1$, es *doble*, es decir, con multiplicidad algebraica $m_a(\lambda_1) = 2$ y el segundo, $\lambda_2 = 4$, es *simple*, ya que su multiplicidad algebraica $m_a(\lambda_2) = 1$. Los autovectores asociados al segundo autovalor, que es simple, son del tipo

$$\mathbf{v}_2 = (0, a, a)^T, a \in \mathbb{R}$$

y las multiplicidades geométrica y algebraica coinciden y $\dim(E(\lambda_2))$=1 (dimensión del autoespacio). Esto ocurre siempre que el autovalor es simple. De hecho, no es necesario siquiera comprobarlo. Si el autovalor es simple, entonces el autoespacio es de dimensión 1.

Los autovectores asociados al primer autovalor, que es doble, son del tipo

$$\mathbf{v}_1 = (0, -2a, a)^T, a \in \mathbb{R}$$

luego, el autoespacio $E(\lambda_1)$ tiene dimensión 1, $\dim(E(\lambda_1))$=1, no puede obtenerse una base de autovectores y no hay diagonalizabilidad.

La diagonalizabilidad se puede definir siguiendo la siguiente interpretación. Una matriz A es **diagonalizable** si es **semejante** a una matriz **diagonal**. En efecto, se tiene la siguiente definición de **semejanza**.

Definición 32. *Dos matrices cuadradas A, B, de las mismas dimensiones se dicen **semejantes** y se denota por*

$$A \sim B$$

si existe una matriz no singular P tal que

$$P^{-}1AP = B$$

Todas las matrices semejantes tienen el mismo polinomio característico y, por tanto, el mismo espectro (mismos autovalores).

Ejercicio 17. *Sean dadas las matrices*

$$\begin{pmatrix} -3 & -1 & 1 \\ 1 & -6 & 1 \\ 1 & -2 & -3 \end{pmatrix}_{3,3}, \begin{pmatrix} -4 & -1 & -1 \\ -1 & -5 & 1 \\ -1 & 0 & -3 \end{pmatrix}_{3,3}$$

Estudiar su diagonalizabilidad.

Para las matrices simétricas se tiene un resultado más fuerte.

Teorema 11. *Toda matriz simétrica de coeficientes reales es ortogonalmente diagonalizable y sus valores propios son reales.*

Formas Cuadráticas

Una aplicación importante del espectro de una matriz es la posibilidad de clasificar la forma cuadrática asociada

$$q(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$$

En las secciones anteriores se estudió su clasificación a través de métodos gráficos. En esta sección se verá la manera de hacerlo mediante métodos analíticos y numéricos utilizando la teoría del problema de autovalores. En términos del espectro de una matriz, se tienen las siguientes definiciones.

Definición 33. *Se dice que una matriz cuadrada A es* ***Definida Positiva*** *si su espectro es positivo y* ***Semi-Definida Positiva*** *si su espectro es no negativo (al menos un autovalor es nulo). Definiciones análogas se obtienen para matrices* ***Semi-Definidas Positivas****, donde*

$$\mathbf{x}^T A\mathbf{x} \geq 0$$

y los autovalores son todos no negativos, las ***Definidas Negativas****, donde*

$$\mathbf{x}^T A\mathbf{x} < 0$$

y los autovalores son todos negativos, las ***Semi-Definidas Negativas****, donde*

$$\mathbf{x}^T A\mathbf{x} \leq 0$$

y los autovalores son todos no positivos y las ***Indefinidas****, que no tienen signo definidos (es decir, hay autovalores positivos y negativos).*

Ejercicio 18. *Sea dada la matriz*

$$\begin{pmatrix} 10 & 2 & 1 \\ 2 & 1 & 0 \\ 0 & 0 & 10 \end{pmatrix}_{3,3}$$

Demostrar que es simétrica y definida positiva. Calcular una base de autovectores ortonormal. Dado el vector $\mathbf{u} = (1, 0, 1)^T$ *calcular sus coordenadas en la base ortonormal.*

A.1.13. Operadores Traza y Determinante

El operador Traza es un campo escalar que calcula la suma de los elementos de la diagonal de una matriz cuadrada.

Definición 34. *Sea* $A \in \mathbb{R}^n$*. Se define a Traza de A por*

$$\mathbf{Tr}(A) = \sum ia_{i,i}$$

La Traza de una matriz cuadrada puede calcularse en Matlab utilizando el comando **trace(A)**.El operador Traza es invariante frente a la operación de trasposición:

$$\mathbf{Tr}(A) = \mathbf{Tr}(A^T)$$

Permite dar una definición alternativa a la Norma de Frobenius de una Matriz:

$$||A||_F = (\mathbf{Tr}(AA^T))^{1/2}$$

El operador Determinante de una matriz cuadrada $A \in \mathbb{R}^{n,n}$ es otro campo escalar con numerosas aplicaciones que se define de forma recursiva mediante la *Regla de Laplace.*

Definición 35. *Sea* $A \in \mathbb{R}^{n,n}$*. El determinante de la matriz A se define mediante la expresión*

$$det(A) = \begin{cases} a_{1,1} & n = 1 \\ \sum_{j=1}^{n} \triangle_{i,j} a_{i,j}, & n > 1, \forall i = 1..n \end{cases}$$

siendo $\triangle_{i,j}$ los co-factores de la matriz A, que se calculan con la fórmula

$$\triangle_{i,j} = (-1)^{i+j} det(A_{i,j})$$

y donde $A_{i,j}$ denota la matriz que se obtiene al eliminar la i-ésima fila y j-ésima columna de A.

El algoritmo que corresponde a la fórmula recursiva tiene complejidad $2(n+1)!$ y sólo sería utilizable para matrices de pequeña dimensión. Un algoritmo mucho más eficiente y que se usa en la práctica (pues es la base del algoritmo que se implementa en Matlab, **det(A)**) se obtiene a través de la técnica **Factorización LU** o proceso de **Eliminación de Gauss**. Su complejidad es de orden polinómico n^3. El cálculo del determinante de una matriz cuadrada permite establecer si esta es invertible.

Definición 36. *Sea $A \in \mathbb{R}^{n,n}$ una matriz cuadrada. Si $det(A) = 0$, la matriz se dice **singular** y, por tanto, no es invertible. Si $det(A) \neq 0$ la matriz es **invertible**.*

La condición de anulación del determinante se utiliza para forzar que el sistema A.2 tenga infinitas soluciones. Otra propiedad importante del determinante es que si $A = BC$ entonces

$$det(A) = det(B)det(C)$$

A través del conocimiento del espectro de una matriz, es decir, sus autovalores, se puede calcular el determinante y la traza mediante las fórmulas:

$$det(A) = \prod_{i=1}^{n} \lambda_i, \mathbf{Tr}(A) = \sum_{i=1}^{n} \lambda_i$$

además de las normas espectral (norma euclídea para matrices cuadradas)

$$||A||_2 = \sqrt{\lambda_{\text{máx}}(A^T A)}$$

y de Frobenius

$$||A||_F = (\sum_{i=1}^{n} \lambda_i^2)$$

Condicionamiento de un Sistema

Conociendo los autovalores máximo y mínimo de una matriz que define un sistema se puede calcular una constante llamada **número de condición espectral** de la matriz, definida por

$$K(A) = \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}}$$

Para obtener este número con Matlab se utiliza el comando **cond**(A).

Ejercicio 19. *Sea dada la mtriz de Hilbert de orden n definida por*

$$a_{i,j} = \frac{1}{i+j-1}, \qquad i,j = 1..n$$

*que en Matlab puede construirse usando el comando **hilb(n)**. Calcular el condicionamiento de la matriz de Hilbert de orden n para n=1..100 y representar gráficamente su comportamiento.*

Factorización de Cholesky y Factorización QR

La factorización de Cholesky se utiliza para un tipo especial de matrices y es muy útil computacionalmente.

Definición 37. *Sea A una matriz cuadrada. Si A es* ***simétrica*** *y* ***definida positiva*** *entonces existe una* ***Factorización de Cholesky*** *en la forma*

$$A = HH^T$$

donde H es una matriz triangular inferior cuya diagonal está compuesta por elementos positivos.

El coste computacional de esta factorización es de orden $n^3/3$, que es la mitad de la factorización LU. En Matlab este algoritmo está implementado mediante el comando **chol(A)** y da como output la matriz triangular superior H^T. Informalmente se puede decir que se está calculando la raíz cuadrada de una matriz siempre que sea simétrica y definida positiva.

Ejercicio 20. *Sean dadas las matrices*

$$A_1 = \begin{pmatrix} 2 & 1 \\ 1 & 5 \end{pmatrix}, A_2 = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$$

Calcular, si existe, su factorización de Cholesky.

La **Factorización QR** se utiliza a menudo para trabajar con sistemas Sobredeterminados ($m > n$).

Definición 38. *Sea $A \in \mathbb{R}^{m,n}$ de rango máximo, existe una única Factorización QR, es decir, existe una matriz $Q \in \mathbb{R}^{m,m}$ ortogonal y una matriz $R \in \mathbb{R}^{m,m}$ trapezoidal superior tales que*

$$A = QR$$

En Matlab, el comando para obtener la Factorización es

$$[Q, R] = \mathbf{qr}(A).$$

Ejercicio 21. *Sea dada la matriz*

$$A = \begin{pmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 3 \\ 1 & 3 & 7 \end{pmatrix}_{4,3}$$

Calcular su factorización QR. Comprobar que la norma de Frobenius de la matriz residuo E = A - QR es de orden 10^{-15}.

A.1.14. Descomposición Espectral

Anteriormente fue visto que dada una matriz cuadrada diagonalizable entonces existe una matriz cuadrada no singular V tal que

$$V^{-1}AV = D = diag(\lambda_1, ..., \lambda_n)$$

Es decir, A es semejante a una matriz diagonal y, por tanto, es diagonalizable. La matriz V tiene como columnas los autovectores de A que forman una base en el espacio, puesto que es no singular al ser diagonalizable. Por tanto, se tiene que A puede factorizarse mediante su espectro

$$A = VDV^{-1}$$

permitiendo estudiar la estructura de la matriz, su carácter y comportamiento. Si la matriz dada no es diagonalizable la descomposición espectral sigue siendo cierta en la forma

$$AV = VD$$

Teorema 12. *Sea dada una matriz cuadrada y simétrica. Entonces el espectro es real y los autovectores son ortonormales.*

Ejemplo 19. *Sea dada la matriz simétrica*

$$A = \begin{pmatrix} 10 & -6 \\ -6 & 5 \end{pmatrix}$$

La ecuación característica es:

$$p_A(\lambda) = (10-\lambda)(5-\lambda) - 36 = \lambda^2 - 15\lambda + 14 = (\lambda - 1)(\lambda - 14)$$

luego el espectro

$$\sigma(A) = \{1, 14\}$$

Resolviendo el sistema lineal homogéneo

$$(A - 14I)\mathbf{v}_1 = \mathbf{0}$$

se tiene el autovector normalizado

$$\mathbf{v}_1 = (3/\sqrt{13}, -2/\sqrt{13})$$

Resolviendo el sistema lineal homogéneo

$$(A - I)\mathbf{v}_2 = \mathbf{0}$$

se tiene el autovector normalizado

$$\mathbf{v}_2 = (2/\sqrt{13}, 3/\sqrt{13})$$

Autovectores que provienen de autovalores distintos son ortogonales:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$$

La matriz V que diagonaliza A es ortogonal,

$$V^T = V^{-1}$$

y verifica

$$V = \frac{1}{\sqrt{13}} \begin{pmatrix} 3 & 2 \\ -2 & 3 \end{pmatrix}, \qquad V^T A V = D = \begin{pmatrix} 14 & 0 \\ 0 & 1 \end{pmatrix}$$

Teorema 13. *Dada una matriz cualquiera $A \in \mathbb{R}^{m,n}$, las matrices*

$$A^T A \in \mathbb{R}^{n,n}, \qquad AA^T \in \mathbb{R}^{m,m}$$

son cuadradas y simétricas. Sus autovalores son reales y no negativos.

Este resultado tiene implicaciones importantes al permitir realizar la Descomposición por Valores Singulares (SVD) de cualquier matriz. Además, afirma que el espectro es no negativo, luego A^TA y AA^T son semi-definidas positivas o definidas positivas. En cualquier caso, las formas cuadráticas asociadas son convexas y existe al menos un óptimo del problema de minimización.