Departamento de Teoría de la Señal y Comunicaciones

PROYECTO FIN DE CARRERA

ANÁLISIS DE COMPONENTES PRINCIPALES:

VERSIONES DISPERSAS Y ROBUSTAS AL RUIDO IMPULSIVO

Autor: Andrés Sánchez Mangas

Tutora: Vanessa Gómez Verdejo

Leganes, 25 de Abril de 2012

Título: ANÁLISIS DE COMPONENTES PRINCIPALES: VERSIONES DISPERSAS Y EXTENSIONES CON DIFERENTES COSTES

Autor: Andrés Sánchez Mangas

Director: Vanessa Gómez Verdejo

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 25 de Abril de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar, agradecer a mi familia todo el apoyo y los ánimos que me han dado durante todos estos años. En especial a mis padres porque gracias a ellos he llegado hasta aquí y no lo podría haber logrado sin ellos. Siempre me han apoyado incondicionalmente y me han dado todas las facilidades para encontrar mi camino.

A mi tutora Vanessa Gómez Verdejo que siempre me ha ayudado y me ha mostrado la dirección a seguir cuando surgían complicaciones. Por la paciencia y el trabajo invertidos en mi. Agradecerle la dedicación y los ánimos para que este proyecto saliera adelante.

También me gustaría agradecer a mis compañeros de universidad. Han sido muchos los compañeros con los que he compartido clases, prácticas y demás tiempo en la universidad. Todos y cada uno de ellos me han aportado algo y han hecho más llevadero este viaje. Ha habido buenos y malos ratos pero gracias a vosotros al final siempre había una sonrisa que me permitía volver con ilusión al día siguiente.

A mis amigos por sacarme de la rutina y permitirme desconectar. Por restarle trascendencia a los problemas y aportar siempre una visión optimista. Por esos excepcionales viajes por el mundo en los que hemos conocido otras culturas tan diferentes que nos ha permitido ver la vida de otra manera.

Agradecer particularmente a los que compartieron conmigo mi año Erasmus en Göterborg. Gracias a ellos ese año fue increíble y las experiencias que vivimos allí inolvidables.

En definitiva, dar las gracias a todos aquellos que habéis creído en mi.

Resumen

El análisis de componentes principales ("Principal Component Analysis", PCA) es un método de extracción de características no supervisado ámpliamente usado en la actualidad. Mediante esta técnica se pueden procesar un extenso conjunto de datos y reducir su dimensionalidad con una pérdida mínima de información. Sin embargo, este método presenta algunas desventajas como la dificultad de análisis de los datos resultantes o una función de coste poco robusta frente al ruido.

Partiendo de la formulación estándar del PCA, se desarrollan y evalúan dos extensiones del mismo que buscarán paliar algunas de sus deficiencias.

La primera de las versiones propuestas forzará dispersión sobre los vectores de proyección obtenidos, para así mejorar la interpretabilidad de la solución obtenida. Con este fin, buscará que los vectores de proyección contengan el mayor número de coeficientes nulos. De esta forma cada vector de proyección solo dependerá de unas pocas variables de entrada y se conocerá como influye cada variable de entrada en los datos de salida.

La segunda versión que se enunciará modificará la función de coste del PCA original por una más robusta frente al ruido impulsivo. Se utilizará la misma función de coste que en las máquinas soporte de vectores, la función "ε-insensible".

Para completar el estudio de estas dos extensiones se realizarán varios experimentos, comparando sus resultados con los del PCA original.

Palabras clave: *análisis de componentes principales (PCA), análisis de componentes principales dispersas (SPCA), ε-insensible, extracción de características.*

Abstract

The Principal Component Analysis (PCA) is a non supervised feature extraction method widely used nowadays. Through this technique is possible to process a large dataset, reducing its dimension with a minimum loss of information. However, this method presents some drawbacks such as the difficulty to analyze the provided results or the lack of robustness against impulsive noise..

Building on the standard formulation of the PCA, this final degree project will introduce and evaluate two extensions, which aim to overcome some of its limitations.

The first proposed extension will force sparsity over the projection vectors so that the interpretability of the solution is enhanced. With this purpose, this new approach will make the projection vectors present a large number of zero coefficients. Therefore, each projection vector will only depend on a few input variables, making easier analyze the influence of each input feature over the output data.

The second extension relies on modifying the standard PCA cost function to provide the method robustness against impulsive noise. This new cost function will be the well-known "ε-insensitive" function employed by the Support Vector Machines.

To complete this study, the performance of both proposals will be analyzed in detail over several experiments, comparing their results with those of the standard PCA.

Ke*ywords*: *principal component analysis (PCA), sparse principal component (SPCA), ε-insensitive, feature extraction.*

Índice general

Índice de Figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Aprendizaje máquina y métodos de extracción de características

Aprendizaje máquina ("Machine Learning", ML), es un término difícil de definir porque abarca un espectro bastante grande de conceptos normalmente relacionados con la *inteligencia artificial* [1]. En general, se entiende como una familia de técnicas que permiten mejorar a un sistema en el tiempo a partir de la experiencia. Estas técnicas pueden estar compuestas de tareas como reconocimiento de patrones, planificación, diagnosis, predicción, etc.

Se puede pensar que un sistema tiene que estar diseñado desde el primer instante para funcionar perfectamente en su cometido. Sin embargo, existen múltiples causas por las que esto no ocurre. Hay campos de trabajo donde se producen cambios continuamente y las máquinas se tienen que adaptar para evitar un costoso re-diseño constante. En otros ambientes las tareas no pueden ser definidas sin datos previos, o no se conocen las posibles relaciones entre los datos de entrada. También puede ocurrir que en la etapa de diseño no se conozca perfectamente las características de trabajo del sistema y este se debe adaptar a las condiciones que se encuentre sobre el terreno [1].

Otro caso, el cual va a ocupar este proyecto, acontece cuando hay demasiadas variables disponibles y la información relevante para resolver la tarea se encuentra enmascarada. Esto que dificulta considerablemente el diseño del sistema y suele deteriorar sus prestaciones. Para este caso, es habitual y recomendable emplear una etapa de preprocesado que permita reducir el número de variable empleadas, ya sea mediante una transformación de las mismas, lo que se conoce como métodos de extracción de características, o mediante una selección de las variables más relevantes [2].

En este contexto de escenarios dinámicos de trabajo es donde aparece el concepto ML con el objetivo de que los sistemas sean capaces de aprender y adaptarse. De las diferentes técnicas que forman parte del ML nos centraremos en aquellas que estudian la extracción de características[3].

Existen sistemas en los que la transformación de las variables de entrada antes de trabajar con ellas produce una gran mejora en el funcionamiento. Esta etapa de pre-procesado tiene como principal objetivo convertir las variables a una forma específica a partir de la cual el sistema podrá hacer un mejor uso de ellas, no solo al disponer mejor de la información sino sacando a relucir posibles relaciones ocultas entre las variables. Al pre-procesado que tiene estas particularidades se le llama extracción de características.

La incorporación de esta etapa de preprocesado, antes de la realización del conjunto de tareas originales que conforman el sistema, puede ofrecer una mejora bastante notable en el rendimiento, además de una simplificación del sistema (al tener que trabajar con un menor número de variables) y un mejor entendimiento del problema. Se suele usar en gran medida en sistemas que requieran clasificación o regresión como por ejemplo en el reconocimiento de patrones[4].

Otro motivo por el cual el preprocesado obtiene tanta importancia en el comportamiento de ciertos sistemas se encuentra cuando estos tienen que hacer frente a conjuntos de datos con una alta dimensionalidad. Cuando un conjunto de datos está compuesto por una única variable y después se dispone de una variable adicional el comportamiento del sistema sufre una mejora al tener más información. Esto indicaría que se podría aumentar infinitamente el número de variables con el fin de mejorar el sistema. Pero en la realidad no ocurre así porque llegado a cierta cantidad de variables el funcionamiento se empieza a degradar al aparecen variables redundantes con las anteriores o directamente irrelevantes para la tarea.

Hoy en día es bastante común encontrarse con conjuntos de datos de alta dimensionalidad formados por más 10^4 variables en ámbitos como la clasificación de genes

[5], análisis multiespectral de imágenes [6], el reconocimiento de patrones [4] o los sensores de alta resolución de cualquier tipo[7]. Por esta razón, cada vez son más habituales las etapas de preprocesado que tengan como objetivo disminuir las dimensiones de los datos de entrada y la extracción de características.

Debido a la gran demanda de este tipo de técnicas es posible encontrar diversos métodos que realicen esta etapa de pre-procesado. A continuación se van a presentar brevemente varias técnicas que llevan a cabo este cometido y forman parte de la familia de los *métodos de análisis multivariante* ("MultiVariate Analysis", MVA) [7] y [8]. Estarán diferenciados en métodos no supervisados o supervisados:

- No supervisados. Dentro de esta categoría se encuentra el *Análisis de Componentes Principales* ("Principal Component Analysis", PCA) [9]. Este es un método de reducción de la dimensionalidad que únicamente tiene en cuenta la información contenida en los datos de entrada sin importar las tareas posteriores de clasificación o regresión, por lo que se le considera como una forma de aprendizaje no supervisado [4]. Su cometido es proyectar los datos de entrada en las direcciones de máxima varianza y así poder eliminar aquellas direcciones que aporten menos información. El inconveniente de este método es que al no tener en cuenta el objetivo de los datos no controla si la información eliminada puede ser importante posteriormente.
- Supervisados. El método más destacado dentro de esta categoría es *mínimos cuadrados parciales* ("Partial Least Squares", PLS) [10]. Esta técnica tiene como base la idea que hay información que suele estar representada en varios conjuntos de datos por lo que su dimensión puede ser reducida si se proyectan mediante variables ocultas. Por lo tanto, tiene como cometido analizar las relaciones entre dos conjuntos de datos, a partir de variables latentes. La forma de hallar estas relaciones es diferente dentro de las múltiples variantes que se pueden encontrar de este método, pero son especialmente importantes aquellas que maximizan la covarianza entre las proyecciones de los datos de entrada y las variables de salida o etiquetas asocidadas a los datos de entrada. Existe otro método bastante parecido al PLS pero que en vez de utilizar la convarianza usa la correlación entre los conjuntos de datos proyectados. A este método se le llama *análisis de correlaciones canónicas* ("Canonical Correlation Analysis", CCA) [8], [11]. Se puede considerar el CCA como un caso especial del PLS pero con características propias. El PLS tiene distintas versiones que son bastante utilizadas en la actualidad como el O-PLS [12], MPLS [13] o el PLS-PB [14].

Hasta este momento todos los métodos de extracción de características están basados en proyecciones lineales. Sin embargo, se pueden encontrar conjuntos de datos de entrada y de salida estén relacionados no-linealmente. Para estos casos en los que las aplicaciones lineales no sean muy eficientes se han desarrollado versiones kernel de estos métodos [15]. En estas versiones los datos de entrada son proyectados en espacios con una alta dimensionalidad donde se pueden aplicar los métodos lineales de forma más eficaz. También existen otras formas más complicadas de lidiar con el problema de las relaciones no-lineales de los datos en las que las relaciones entre las variables latentes son no-lineales [7].

En este proyecto nos vamos a centrar en el más sencillo de los métodos MVA presentados, el PCA. Debido a la simplicidad de este método no supervisado podremos estudiar más fácilmente las modificaciones en su estructura para intentar mejorar sus prestaciones. Por este motivo, se limitará la investigación a las estructuras lineales, ya que introducir modelos kernel para tratar relaciones no-lineales complicaría los análisis y se saldría de los objetivos que abarca este proyecto.

1.2 Análisis de Componentes Principales

El análisis de componentes principales es una técnica de extracción de datos ampliamente utilizada en la actualidad. El objetivo principal que persigue dicha herramienta es reducir la dimensionalidad de un conjunto de observaciones con una gran cantidad de variables, ayudándose del estudio de la estructura de varianzas-covarianzas entre las variables que componen los datos de entrada. A partir de la proyección de los datos de entrada sobre las direcciones de máxima varianza se obtendrá un nuevo espacio de representación de los datos en el que se puede eliminar fácilmente aquellas componenetes con menor varianza, garantizando la mínima pérdida de información.

El origen del PCA es difícil de establecer. Preisendorfer y Mobley [16] apuntan que Beltrami [17] y Jordan [18] derivaron independientemente las expresiones de la descomposición en valores singular en la forma utilizada por el PCA. Sin embargo, es ampliamente aceptado que las primeras descripciones de la técnica conocida actualmente como PCA fueron dadas en sendos artículos por Pearson [19] y Hotelling [20].

Los dos artículos adoptan diferentes aproximaciones. En 1901, Karl Pearson [19] se preocupó por encontrar líneas y planos que mejor encajaran el conjunto de datos en un

espacio p-dimensional, y que geométricamente optimizaran los problemas hacia los que se dirigían las componentes principales. Pearson comentó respecto a la computación, más de 50 años antes de la generalización de los ordenadores, que sus métodos podía ser aplicados fácilmente a los problemas numéricos, y aunque decía que los cálculos empezaban a ser bastante voluminosos para cuatro o más variables, se ha demostrado que en computación son factibles.

Por otro lado en 1933, Hotelling [20] realizó una derivación del álgebra común dando un resultado cercano a la actual definición del PCA. La motivación de Hotelling era probar que puede haber un conjunto fundamental menor de variables independientes el cuál determine los valores de las originales *p*-variables. Él explica que tales variables han sido llamadas "factores" en la literatura, pero introduce el término alternativo "componentes" para evitar su confusión con otros usos de la palabra "factor" en matemáticas. Hotelling eligió sus componentes para maximizar las contribuciones del total de varianzas de las variables originales, y llama componentes a aquellas que son derivadas en la forma de componentes principales. El análisis que encuentra tales componentes es, por lo tanto, bautizado como método de componentes principales.

Después de dar la derivación, Hotelling apuntó hacia mostrar cómo encontrar las componentes usando el potencial del método. Él también estudió una interpretación geométrica diferente de la dada por Pearson, en términos de elipsoides de probabilidad constante para distribuciones multivariables normales.

Otro artículo de Hotelling [21] da una versión acelerada de la eficacia del método para encontrar las componentes principales; también Girshick [22] proporciona algunas derivaciones alternativas, e introduce la idea de las muestras sean las estimaciones más probables de la población subyacente de las componentes principales. Después, Girshick [23] investigó el muestreo de distribuciones asintóticas de coeficientes y varianzas de componentes principales. Pero durante los 25 años siguientes al artículo de Hotelling solo aparece una pequeña cantidad del trabajo de diferentes aplicaciones del PCA.

Desde entonces una explosión de nuevas aplicaciones y más profundos desarrollos han sido llevadas acabo. Esta expansión refleja el crecimiento general de la literatura estadística, pero como el PCA requiere un considerable poder de computación, la expansión de su uso coincide con la generalización de los ordenadores electrónicos. A pesar de los comentarios optimistas de Pearson, no es realmente viable realizar PCA a mano, a menos que *p* sea un número igual a cuatro o menor. Sin embargo, es precisamente para valores grandes de *p* que

el PCA es más útil, por eso todo el potencial de esta técnica no pudo ser explotado hasta después del advenimiento de los ordenadores.

En el principio de la expansión del interés por el PCA aparecieron cuatro trabajos que han llegado a ser importantes referencias dentro de este campo:

- El primero de ellos, escrito por Anderson en 1963 [24], es el más teórico de los cuatro. Discute el muestreo de distribuciones asintóticas de coeficientes y varianzas de muestras de componentes principales, construidas al principio del trabajo de Girshick [23], y ha sido frecuentemente citado y posteriormente desarrollado teóricamente.
- El artículo de Rao en 1964 [25] resalta por el gran número de nuevas ideas concernientes al uso, interpretación y extensión del PCA que se introducen.
- Gower (1966) [24] discute las uniones entre el PCA y otras técnicas estadísticas. También proporciona un número importante de puntos de vista geométricos.
- Finalmente, Jeffers (1967) [26] dio un gran ímpetu al lado práctico de este campo mediante la comparación de dos casos de estudio en los cuales el uso del PCA iba más allá de la simple reducción de dimensiones.

Adicionalmente, cabría añadir a esta lista el libro de Preisendorfer y Mobley [16]. Aunque es relativamente desconocido fuera de las disciplinas de meteorología y oceanografía, compite con Rao [25] en nuevas ideas relacionadas con el PCA, algunas de las cuales no han sido todavía completamente exploradas.

Todo este trabajo se ha visto reflejado en el amplio abanico de áreas en las que el PCA ha sido y puede ser aplicado, entre las que destacan: incluye agricultura, biología, química, climatología, demografía, ecología, economía, estudios alimenticios, genética, geología, meteorología, oceanografía [16], psicología y control de calidad. Sería fácil añadir otros elementos a esta lista más específicos del tratamiento de datos como la clasificación de caracteres manuscritos [27], el reconocimiento de rostros humanos [28] o más recientemente ha sido usado en el análisis de genes [29].

El PCA, como ya se ha dicho previamente, es una técnica no supervisada que no tiene en cuenta el objetivo de los datos, lo que le da una gran versatilidad y ha facilitado su uso en tantas aplicaciones.

1.3 Motivación y objetivos de este Proyecto Fin de Carrera

A pesar el gran éxito y de la vasta cantidad de aplicaciones que incluyen el PCA como parte de su funcionamiento, existen inconvenientes en su uso. La principal desventaja está en el hecho que cada componente principal sea una combinación lineal de todas las variables originales, lo que hace muy difícil poder analizar la importancia de cada variable original en el nuevo espacio [30]. Para hacer más fácil la interpretación de las variables resultantes han surgido nuevos métodos como el *PCA disperso* ("Sparse PCA", SPCA) [30] y [31]. Esta técnica reformula la implementación típica del PCA como un problema de optimización basado en la minimización del error de reconstrucción donde se fuerza que los vectores de proyección sean dispersos, es decir, una gran parte de sus coeficientes sea cero.

Siguiendo la idea de [32], [33], [34] ,[35] y [36] se plantea otra versión más robusta del PCA. En concreto, se plantea modificar su coste por una función "ε-insensitive", tal y como [37] ha planteando para el kernel PCA, pero lo formularemos en su versión lineal llegando a una formulación iterativa que nos permite utilizar toolboxes conocidas y resolver fácilmente el problema.

En esta extensión, al igual que para el caso del SPCA, se parte de la formulación del PCA en términos de minimización del error de reconstrucción para modificar la función de coste por un coste insensible para errores de un intervalo [-ε, ε]. Esta función de coste, "ε-insensitive", es típica de las *máquinas de vectores soporte en regresión* ("Support Vector Regression", SVR) y ha adquirido un gran uso por otorgar robustez a estos clasificadores ante la presencia de ruido impulsivo, característica que queremos extender al PCA.

Siguiendo estos objetivos, en este proyecto se van a presentar y analizar estas extensiones del PCA que tienen la finalidad de paliar algunas de las deficiencias de la versión original algoritmo. Se van a proponer varias versiones de cada extensión y se procederá a un estudio pormenorizado de las ventajas y desventajas que ofrecen cada una de ellas en diversos escenarios.

En primer lugar, en el segundo capítulo se procederá a la explicación teórica del PCA original, donde se explicará su funcionamiento analizando las diferentes formulaciones que admite este método, así como dos posibles implementaciones.

En el tercer capítulo se presentan las extensiones que introducen cambios en el algoritmo original del PCA. Antes de ello se mostrará una formulación modificada del PCA para que su ejecución se produzca de forma iterativa. Esta formulación alternativa servirá de paso intermedio entre el PCA original y las otras dos extensiones con cambios más significativos. En la segunda sección de este capítulo se presenta el PCA disperso (SPCA). Esta primera extensión utiliza vectores de proyección dispersos a la hora de hallar las componentes principales para eliminar de los vectores proyectados aquellas componentes que aportan muy poca información. De esta forma se facilita la interpretación de las componentes principales. Para terminar el capítulo se expondrá la versión modificada del PCA que cambia la función de coste por una más robusta frente al ruido impulsivo. A esta extensión se la llamará Ɛ-PCA y su planteamiento tendrá similitud con el de las SVR. Por este motivo nos podremos apoyar en ellas para la resolución del Ɛ-PCA.

A continuación, en el cuarto y en el quinto capítulos se procede con el análisis del funcionamiento del SPCA y del Ɛ-PCA. Para ello en el cuarto capítulo, se estudiaran las distintas versiones del SPCA mediante tres experimentos diferentes. También se compararán los resultados ofrecidos por el SPCA con los del PCA original. En el quinto capítulo se plantean tres experimentos para comprobar el funcionamiento del Ɛ-PCA principalmente frente al ruido impulsivo. También se compararán sus resultados con los del PCA original.

Por último, en el sexto capítulo se expondrán las conclusiones resultantes de la realización de este proyecto y las lineas futuras a seguir en este campo.

Capítulo 2

Análisis de Componentes Principales

A continuación se va a presentar el desarrollo teórico que realiza el PCA con el objetivo final de disminuir las dimensiones de un conjunto de datos de entrada mediante una transformación lineal que los proyecte sobre las direcciones de máxima varianza. Se presentarán dos formulaciones alternativas del PCA, que parten de diferentes puntos de vista, pero que como veremos proporcionan soluciones equivalentes. Por último, se describirán dos posibles formas de implementar el algoritmo: en bloque o recursivamente mediante deflacción.

2.1 Descripción del PCA

La técnica del *análisis de componentes principales* consiste en analizar un conjunto de de datos de entrada, el cuál contiene diferentes observaciones descritas por múltiples variables independientes o dependientes y cuyas relaciones entre sí no tienen porqué conocerse. Como ya se ha dicho anteriormente el objetivo principal es reducir la dimensión del conjunto de datos de entrada intentado mantener la mayor cantidad de información posible para poder analizarlos de forma más fácil y que en etapas posteriores, como clasificadores o regresores, se puedan simplificar los criterios de decisión.

El PCA realiza en primer lugar una transformación lineal de los datos en un nuevo sistema de coordenadas ortogonales. Los vectores de proyección de los datos en el nuevo espacio son las direcciones de máxima varianza de los datos de entrada. Mientras que las nuevas variables resultantes de proyectar los datos de entrada sobre los vectores de proyección se llamarán *componentes principales* ("Principal Component", PC). En este nuevo sistema de coordenadas las componentes principales están ordenadas automáticamente según la varianza de la proyección de datos, es decir, según la cantidad de información que contengan. Finalmente, se puede reducir la dimensión de los datos resultantes en el nuevo espacio eliminando las componentes principales que presenten una menor varianza, es decir, que aporten menos información.

La base matemática que se utiliza para desarrollar el PCA es el álgebra lineal. Veremos como el PCA está íntimamente relacionado con la técnica algebraica de la descomposición en valores singulares ("Singular Value Decomposition", SVD) [38]. Además, la compresión de como están relacionados el PCA y la SVD ayudará al mejor entendimiento de las posibles aplicaciones de este método.

El PCA tiene dos propiedades muy importantes que hacen de este método de reducción de la dimesionalidad tan popular [30]:

1. Las componentes principales obtienen secuencialmente la máxima variabilidad o varianza de X, por lo que se garantiza la mínima pérdida de información (en el sentido de error de reconstrucción).
2. Las componentes principales obtenidas son ortogonales entre si, facilitando su posterior procesado, ya que pueden tratarse independientemente.

Los datos de entrada estarán dispuestos en una matriz $\mathbf{X} \in R^{m \times n}$, donde *m* es el número de variables de entrada y *n* es el número de observaciones. Mientras que los datos proyectados estarán en una matriz $\mathbf{Y} \in R^{p \times n}$, con *p* variables salida y n observaciones, donde el número de variables de salida es menor que el de entrada ($p<m$). Cada vector $\mathbf{x_i} = [x_1, x_2, \ldots, x_m]^T$ [1] contendrá todas las variables de entrada asociadas a una observación *i* e $\mathbf{y_i} = [y_1, y_2, \ldots, y_p]^T$ contendrá las variables de salida.

La tarea del PCA es encontrar una matriz $\mathbf{U} \in R^{m \times p}$ que transforme linealmente el espacio de los datos de entrada en **X** en otra matriz **Y** con un número menor de variables. Este proceso se realiza mediante una combinación lineal de las variables originales de modo que se proyecten sobre las direcciones de máxima varianza de los datos, y se conserve así la

1 El superíndice ^{T} indica la transposición del vector o la matriz.

máxima cantidad de información. Estas direcciones de máxima varianza estarán definidas por los *p* vectores de proyección, $\mathbf{u}_k$, que se encontrarán en las columnas de **U**.

$$X=\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix} \quad \xrightarrow{PCA} \quad Y=U^T X \qquad Y=\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_p \end{bmatrix}, \quad (m>p)$$

Durante este proyecto se llamarán "componentes principales" a las variables de salida $\mathbf{y}_k = \mathbf{u}_k^T \mathbf{X}$, mientras que los vectores de proyección $\mathbf{u}_k$ serán los coeficientes o cargas de la *k*-ésima componente principal.

En principio **U** puede tener hasta p = m vectores de proyección, una por cada variable. Sin embargo, habitualmente se reducirá esta dimensión conservando únicamente las proyecciones con mayor varianza, lo que produce una pérdida de información. Con el objetivo de minimizar esta pérdida el algoritmo del PCA consigue concentrar la mayor parte de la varianza en un número reducido de variables del nuevo espacio. De esta forma se pueden eliminar aquellas variables del espacio de salida que contengan menos varianza y así perder la menor cantidad posible de información. Como premisa se busca que la mayor cantidad de información quede contenida en el menor número posible de variables.

A continuación se van a describir dos formulaciones que pueden desarrollar el PCA para conseguir la matriz proyección del nuevo espacio. Los objetivos de los que parten son diferentes: en el primer planteamiento, el más habitual y conocido, busca maximizar la varianza en el nuevo espacio de datos; por el contrario, el segundo tiene como objetivo minimizar el error de reconstrucción de espacio original a partir del nuevo espacio con menos variables. Aunque, en principio, parecen problemas diferentes, veremos que ambas formulaciones proporcionan soluciones equivalentes por lo que podrán usarse indiferentemente para el cálculo de los vectores de proyección.

2.1.1 Formulación del PCA como método para maximizar la varianza

Como el objetivo del PCA es reducir el número de variables de los datos de entrada conservando la mayor cantidad de información posible, en este apartado se mostrará el desarrollo para alcanzar esta meta centrándonos principalmente en la conservación de la varianza.

A grandes rasgos el funcionamiento del PCA está enfocado a encontrar, a partir de un conjunto de datos de entrada $\boldsymbol{X}$ con m variables, un vector de pesos $\boldsymbol{u}_1$ capaz de proyectar este conjunto de datos sobre la dirección de máxima varianza de $\boldsymbol{X}$, donde $\boldsymbol{u}_1$ es un vector formado por m variables. Una vez que tenemos $\boldsymbol{u}_1$, el siguiente paso se buscar otro vector $\boldsymbol{u}_2$ ortogonal con $\boldsymbol{u}_1$ que retenga la máxima varianza posible. Se continuará así hasta obtener el p-ésimo vector $\boldsymbol{u}_p$ con la máxima varianza siendo ortogonal con $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{p-1}$. La proyección de los datos de entrada sobre estos vectores da lugar a las componentes principales: $\boldsymbol{y}_1 = \boldsymbol{u}_1^T \boldsymbol{X}$, $\boldsymbol{y}_2 = \boldsymbol{u}_2^T \boldsymbol{X}$, …, $\boldsymbol{y}_k = \boldsymbol{u}_k^T \boldsymbol{X}$. Aunque se puede continuar hasta obtener la m-ésima componente principal, normalmente solo será necesario calcular las p primeras donde estará proyectada la mayor parte de la varianza. El resto de variables serán descartadas, lo que ocasiona una pérdida de información, pero será mínima (en términos de varianza).

Este planteamiento se puede explicar de forma más sencilla si empleamos la formulación matricial. Sea $\boldsymbol{U}$ la matriz compuesta por los p vectores de proyección, $\boldsymbol{X}$ la matriz con los datos de entrada, entonces la matriz $\boldsymbol{Y}$ formada por las p componentes principales, se obtendrá:

$$\boldsymbol{Y} = \boldsymbol{U}^T \boldsymbol{X} \tag{2.1}$$

El objetivo del PCA es conservar la máxima cantidad de información en las variables de **Y**, mientras son ortogonales. Una forma de conseguir esta meta es maximizando la covarianza de las componentes principales, es decir,

$$\max_{\boldsymbol{U}} \ \boldsymbol{C}_{\boldsymbol{YY}}$$

$$\text{sujeto a } \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I} \tag{2.2}$$

Para llevar a cabo la concentración de la información en el menor número posible de componentes y que dicha información no esté repetida, la transformación tiene que asegurar que las variables resultantes en el nuevo espacio sean ortogonales, razón por la que se fuerza la restricción $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$. De esta forma cuando eliminemos los atributos menos significativos perderemos la menor cantidad de información. Para encontrar la matriz $\boldsymbol{U}$ que resuelve el problema dado en (2.2) primero hay que desarrollar su expresión de $\boldsymbol{C}_{YY}$ de la siguiente forma:

$$\boldsymbol{C}_{YY} = \frac{1}{n}\boldsymbol{Y}\boldsymbol{Y}^T = \frac{1}{n}(\boldsymbol{U}^T\boldsymbol{X})(\boldsymbol{U}^T\boldsymbol{X})^T = \frac{1}{n}\boldsymbol{U}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{U} = \boldsymbol{U}^T(\frac{1}{n}\boldsymbol{X}\boldsymbol{X}^T)\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{C}_{XX}\boldsymbol{U} \tag{2.3}$$

A partir de esta expresión podemos redefinir el objetivo del PCA como:

$$\max_{\boldsymbol{U}} \; tr\{\boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{C}_{\boldsymbol{XX}} \boldsymbol{U}\}$$

$$\text{sujeto a } \boldsymbol{U}^{T}\boldsymbol{U} = \boldsymbol{I} \tag{2.4}$$

Ahora tenemos un problema de optimización con restricciones, por lo tanto para resolver está esta expresión se aplicarán multiplicadores de Lagrange. En primer lugar se define

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{bmatrix} \tag{2.5}$$

como la matriz de multiplicadores de Lagrange y tenemos que

$$L_p = Tr\{\boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{C}_{\boldsymbol{XX}} \boldsymbol{U}\} - (\boldsymbol{I} - \boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{U})\boldsymbol{\Lambda}$$

$$\frac{d\,L_p}{d\,U} = 2\boldsymbol{C}_{\boldsymbol{XX}} \boldsymbol{U} + (-2\boldsymbol{U})\boldsymbol{\Lambda} = 0$$

$$\boldsymbol{C}_{\boldsymbol{XX}} \boldsymbol{U} = \boldsymbol{U}\,\boldsymbol{\Lambda} \tag{2.6}$$

llegamos a un problema de autovalores y autovectores sobre la matriz de covarianza de los datos.

De esta forma el PCA se puede resolver como un problema de autovectores y autovalores. Los autovectores de $\boldsymbol{C}_{XX}$ proporcionan las direcciones de máxima varianza de $\boldsymbol{X}$. Por lo tanto, si se escoge $\boldsymbol{u}_k$ como el k autovector de $\boldsymbol{C}_{XX}$, entonces $\boldsymbol{u}_k^T\,\boldsymbol{X}$ será la k componente principal. Los autovectores están ordenados de mayor a menor varianza y el autovalor correspondiente a cada autovector indicara la varianza de la componente principal calculada a partir él.

Habitualmente, en el cálculo de los autovectores y autovalores se utiliza la técnica matemática de la *descomposición en valores singulares* ("Singular Value Decomposition", SVD) [38]. La solución de la SVD de **X** es:

$$\boldsymbol{X} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^{\boldsymbol{T}} \tag{2.7}$$

donde se conoce que $\boldsymbol{U}$ y $\boldsymbol{V}$ son matrices ortonormales, y $\boldsymbol{U} = [\boldsymbol{u}_1\ \boldsymbol{u}_2\ \ldots\ \boldsymbol{u}_m]$ tiene en sus columnas los autovectores de $\boldsymbol{X}\boldsymbol{X}^T$. $\boldsymbol{\Sigma}$ es una matriz diagonal con los *valores singulares* de $\boldsymbol{X}\boldsymbol{X}^T$. Cada valor singular está definido como la raíz cuadrada del autovalor correspondiente ($\sigma_i \equiv \sqrt{\lambda_i}$).

De esta forma se obtiene mediante la SVD de $\boldsymbol{X}$ que:

$$\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} = (\boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^{\boldsymbol{T}})\,(\boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^{\boldsymbol{T}})^{T} = \boldsymbol{U}\,\boldsymbol{\Sigma}^{2}\,\boldsymbol{U}^{\boldsymbol{T}}$$

$$\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U} = \boldsymbol{U}\,\boldsymbol{\Sigma}^{2} \tag{2.8}$$

Esta última expresión es comparable a (2.6) salvo porque en esta no se utiliza $\boldsymbol{C}_{XX}$. Con la SVD de $\boldsymbol{X}$ se calculan los autovectores y autovalores de $\boldsymbol{X}\boldsymbol{X}^{T}$ no de $\boldsymbol{C}_{XX}$. Para igualar ambas expresiones y obtener finalmente la matriz $\boldsymbol{U}$ con las proyecciones ortogonales de máxima varianza de $\boldsymbol{X}$ hay que aplicar la SVD a $1/\sqrt{\mathrm{n}}\,\boldsymbol{X}$.

2.1.2 Formulación del PCA como método para minimizar el error de reconstrucción

Otra forma de abordar el problema del PCA es como un problema de mínimos cuadrados. Ahora buscamos minimizar la diferencia entre los datos originales y los datos que se obtendrían proyectando en el espacio original las *p* nuevas variables. A este error se le denomina error de reconstrucción.

En este caso los vectores de proyección se obtienen resolviendo el siguiente problema de optiminzación.

$$\min_{\boldsymbol{U}} \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\|_{F}^{2}$$

$$\text{sujeto a } \boldsymbol{U}^{T}\boldsymbol{U} = \boldsymbol{I} \tag{2.9}$$

Se puede demostrar que la solución obtenida mediante este problema de optimización es equivalente a la de la expresión (2.4). Comenzaremos extendiendo (2.9):

$$\min_{\boldsymbol{U}} Tr\{(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})^{T}(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})\}$$

$$\text{sujeto a } \boldsymbol{U}^{T}\boldsymbol{U} = \boldsymbol{I} \tag{2.10}$$

Ahora si se desarrolla el término a minimizar de esta expresión se obtiene:

$$\begin{aligned} & Tr\{(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})^{T}(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})\} \\ = {} & Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\} - 2\mathrm{Tr}\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} + Tr\{(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})^{T}(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X})\} \\ = {} & Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\} - 2\,Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} + Tr\{\boldsymbol{X}\,\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} \end{aligned} \tag{2.11}$$

Teniendo en cuenta que $\boldsymbol{U}^{T}\boldsymbol{U} = \boldsymbol{I}$, la expresión anterior se puede simplificar de la siguiente forma:

$$\begin{aligned} & Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\} - 2\,Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} + Tr\{\boldsymbol{X}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} \\ & = Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\} - Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\} \\ & = Tr\{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\} - Tr\{\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{U}\} \end{aligned} \tag{2.12}$$

Finalmente, si eliminamos los términos que no influyen en la minimización, el problema del PCA como minimización del error de reconstrucción queda resumido en la siguiente expresión:

$$\max_{\boldsymbol{U}} \; Tr\{\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{C}_{\boldsymbol{XX}}\boldsymbol{U}\}$$
$$\text{sujeto a } \boldsymbol{U}^{T}\boldsymbol{U} = \boldsymbol{I} \tag{2.13}$$

Por lo que se puede comprobar como ambas formulaciones resuelven el mismo problema. Además, esta nueva formulación facilitará la modificación del comportamiento del algoritmo mediante la adición de nuevos términos o la modificación del coste. Esto hace que la formulación del PCA como la minimización del error de reconstrucción sea un excelente paso intermedio entre el PCA simple y algoritmos más complejos que serán presentados en el próximo capítulo.

2.2 Implementaciones del PCA

Existen dos procedimientos diferentes a la hora de implementar el PCA independientemente del planteamiento escogido. La diferencia entre las dos implementaciones radica en el número de veces que se realice el PCA para encontrar las componentes principales. Si se hallan todas las componentes principales a la vez en una ejecución o se calculan de forma iterativa una por una. La elección de una u otra estructura estará determinado por las características de los datos de entrada, el número de componentes a calcular y el entorno de desarrollo de la implementación.

A continuación se detallan brevemente ambas estructuras.

2.2.1 Implementación recursiva mediante técnicas de deflacción

Este tipo de implementación es la que más pasos necesita para realizar el PCA, aunque

dado que en cada paso sólo calcula un único vector de proyección, cada paso es más sencillo. Se calcularan los vectores de proyección uno por uno en dos etapas:

1. En esta primera etapa se ejecuta el PCA sobre ***X***. Se halla el primer vector de proyección y la primera componente principal. Se pueden utilizar las dos formulaciones presentadas en la sección anterior. Si se usa la que maximiza la varianza hay que resolver (2.4). A partir de la matriz de covarianza $\boldsymbol{C}_{XX}$ se calculará el autovalor λ_i y el autovector $\boldsymbol{u}_i$. Si se busca minimizar el error de reconstrucción se resolverá (2.9) por lo que se hallará el vector de proyección $\boldsymbol{u}_i$ mediante optimización.
2. En la segunda etapa, semejante para ambas formulaciones, se deflacta la matriz de datos para eliminar la información recogida en la componente principal hallada. La nueva matriz de datos será la resta entre la matriz de datos anterior y la recuperación de la matriz de datos anterior proyectada mediante $\boldsymbol{u}_i$.

$$\boldsymbol{X} \leftarrow \boldsymbol{X}\left[\boldsymbol{I} - \boldsymbol{u}_i \boldsymbol{u}_i^T\right]$$

2.2.2 Implementación en bloque

En la implementación del algoritmo en bloque se ejecuta el PCA directamente. Se hallan todas las componentes principales a la vez aplicando la formulación matricial dada por la ec. (2.4) o (2.9) según el planteamiento del PCA elegido.

2.3 Ejemplo sintético

Ahora se expone un ejemplo sintético muy simple que nos servirá para ver claramente el funcionamiento del PCA. El ejemplo está formado un conjunto de datos de entrada compuesto por 500 observaciones con 3 variables cada una. Las dos primeras variables son independientes mientras que la tercera es una combinación lineal de las dos anteriores por lo que no aporta información nueva.

Cada observación se genera de la siguiente forma:

$$X_1 \sim N(0,1) \quad , \quad X_2 \sim N(0,1)$$

$$X_3 = X_1 + X_2$$

En la Figura 1 se muestra el conjunto de datos de entrada:

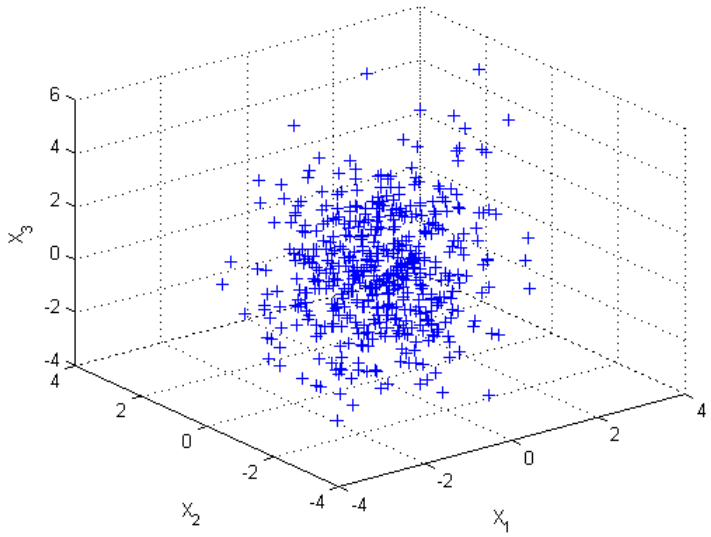

Figura 1: Datos de entrada del ejemplo sintético.

En la Figura 2 se pueden ver los autovalores obtenidos en el ejemplo de sintético. Los autovalores indican la cantidad de información que aporta cada componente principal. En este caso el tercer autovalor es prácticamente 0 por lo que la tercera componente principal no aporta información y puede ser eliminada sin ningún perjuicio para las etapas posteriores. Este resultado era de esperar porque había una variable dependiente de las otras dos en los datos originales.

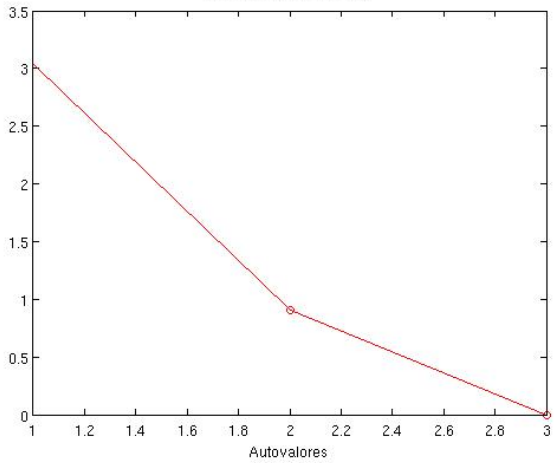

Figura 2: Autovalores obtenidos al aplicar el PCA sobre el conjunto de datos sintéticos del ejemplo.

Como se revela en la Figura 3 (a) tras transformar con el PCA los datos del ejemplo sintético la representación de las tres variables ha cambiado en el nuevo espacio. Ahora hay dos variables que varían perceptiblemente mientras que la tercera variable apenas cambia.

Se observa mejor en la Figura 3 (b, c, d) la relación entre las variables. Queda

demostrado que la tercera variable es claramente prescindible porque contiene muy poca información ya que todas las muestras tienen un valor en torno a 0. Sin embargo, la primera y la segunda variable contienen casi toda la información.

Esto también se puede ver en la representación de los autovalores de los datos en la Figura 2 donde el primer autovalor contiene la mayor parte de la información, el segundo autovalor también guarda información aunque menos y el tercer autovalor que está muy cerca de 0 porque casi no tiene información porque toda casi toda la varianza está contenida en los dos primeros autovalores.

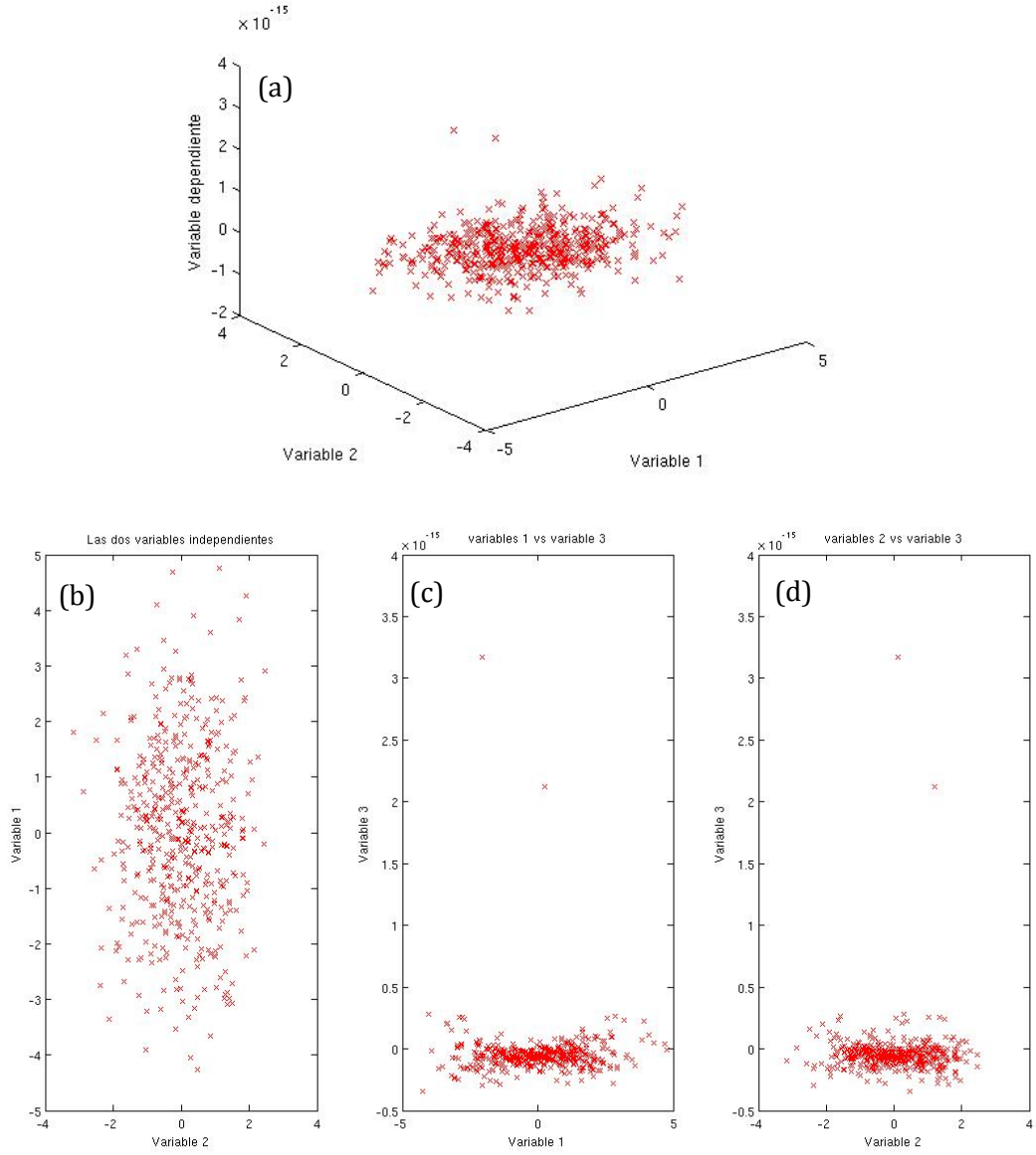

Figura 3: Nuevas Variables obtenidas al proyectar los datos en el nuevo espacio. En (a) están representadas las 3 variables y en (b, c, d) se muestran las variables de 2 en 2.

Capítulo 3

Extensiones del PCA

En este capítulo se van a presentar las versiones modificadas del PCA que van a ser objeto de estudio en este PFC. En primer lugar se plantea una formulación iterativa para resolver el PCA. Esta formulación nos permitirá plantear el PCA de la manera adecuada para que se le pueda introducir modificaciones. Dependiendo de las modificaciones que se realicen se alcanzarán las extensiones propuestas del PCA, SPCA o Ɛ-PCA, que serán presentadas en las secciones 3.2 y 3.3.

3.1 PCA Iterativo

Hasta ahora se han mostrado varias formulaciones equivalentes de resolver el PCA, pero que no aportan ninguna mejora entre ellas. A continuación se va a presentar una forma de resolver el PCA de manera iterativa que facilitará modificar su formulación y así obtener versiones mejoradas del PCA.

Para desarrollar esta formulación iterativa se va a realizar un planteamiento similar al PCA como una minimización del error de reconstrucción, planteado en 2.12. En este planteamiento el PCA es un problema de mínimos cuadrados, que busca minimizar el error de reconstrucción de una matriz de datos tras un cambio de espacio.

Con esta nueva configuración se puede alterar de forma sencilla el término de

minimización para cambiar el comportamiento del algoritmo. Por lo que el PCA iterativo es un excelente paso intermedio entre el PCA simple y versiones más complejas.

3.1.1 Una formulación iterativa del PCA

En primer lugar se va formular el PCA como un problema de regresión teniendo en cuenta que cada PC es una combinación lineal de *p* variables, y por lo tanto las componentes principales pueden ser recuperadas haciendo regresión [30].

La matriz $\boldsymbol{X}$ contiene los datos, $\boldsymbol{y}_i$ es la *i*-ésima componente principal y suponemos que $\hat{\boldsymbol{u}}'$ es estimada por:

$$\hat{\boldsymbol{u}}' = arg\,\min_{\boldsymbol{u}} ' \left| \boldsymbol{y}_i - \boldsymbol{X}\boldsymbol{u}' \right|^2 \tag{3.1}$$

donde $\hat{\boldsymbol{u}} = \frac{\hat{\boldsymbol{u}}'}{|\hat{u}'|}$, y $\hat{\boldsymbol{u}} = \boldsymbol{U}_i$. Esta es una manera sencilla de aproximar el PCA a la regresión.

Sin embargo, a diferencia de la formulación como una minimización del error de reconstrucción ahora vamos a tener dos vectores diferentes para este proceso: un vector, $\boldsymbol{u}$, para la proyección en el nuevo espacio y otro para la recuperación de los datos, $\boldsymbol{v}$. A continuación se presenta una expresión autocontenida de tipo regresión con la que se obtendrán las componentes principales:

$$(\hat{\boldsymbol{v}}, \hat{\boldsymbol{u}}) = arg\,\min_{\boldsymbol{v},\boldsymbol{u}} \sum_{i=1}^{n} \left| \boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{u}^T \boldsymbol{x}_i \right|^2$$

$$\text{sujeto a} \quad |\hat{\boldsymbol{v}}|^2 = 1 \tag{3.2}$$

donde $\hat{\boldsymbol{u}} \propto \boldsymbol{U}$ y $\boldsymbol{x}_i$ es la i-esima fila de la matriz $\boldsymbol{X}$ de datos de entrada.

Si consideramos $\boldsymbol{u}$ *y* $\boldsymbol{v}$ de forma matricial ($\boldsymbol{U}$ y $\boldsymbol{V}$) con dimensiones *m* x *p* podemos obtener las primeras *k* componentes principales utilizando la expresión (3.2), si $\boldsymbol{V}$ es restringida a $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_m$. Entonces se alcanza que $\hat{\boldsymbol{u}}_i \propto \boldsymbol{U}_i$ para *i* = 1, 2, …, p.

La parte crítica es la minimización. La meta es que el algoritmo converja tras varias iteraciones a que **v** sea igual a $\boldsymbol{u}$, como ocurría en el apartado 2.1.2. De esta forma se alcanza $\sum_{i=1}^{n} \left| \boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{u}^T \boldsymbol{x}_i \right|^2 = \sum_{i=1}^{n} \left| \boldsymbol{x}_i - \boldsymbol{u}\boldsymbol{u}^T \boldsymbol{x}_i \right|^2$. Por lo que la minimización bajo la restricción de $\boldsymbol{v}$ ortonormal hace que se obtenga exactamente las *k* primeras componentes principales del PCA ordinario.

Ahora el problema se va a resolver en dos pasos: en el primero se calculará $\boldsymbol{u}$ como un problema de mínimos cuadrados; y en el segundo paso se hallará $\boldsymbol{v}$ como un problema con restricciones mediante la SVD.

3.1.1.1 Cálculo de u

Antes de calcular **u**, se propone un desarrollo matemático que reduce la parte a minimizar con el correspondiente ahorro computacional y la simplificación del funcionamiento del algoritmo [30].

Se parte de la expresión (3.2):

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} \left| \boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{u}^T \boldsymbol{x}_i \right|^2 \\ &= (\boldsymbol{v}-\boldsymbol{u}) \boldsymbol{X}^T \boldsymbol{X} (\boldsymbol{v}-\boldsymbol{u}) \\ &= \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{v} - \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} - \boldsymbol{u}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{v} + \boldsymbol{u} \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} \end{aligned} \tag{3.3}$$

Si se eliminan los términos que no van a ser optimizados, es decir, aquellos que no contengan $\boldsymbol{u}$ en la expresión que tiene que ser optimizada, queda de la siguiente forma:

$$\hat{\boldsymbol{u}} = \min_{\boldsymbol{u}} \left(\boldsymbol{u}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} - 2 \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} \right) \tag{3.4}$$

De esta forma, el cálculo de $\boldsymbol{u}$ queda reducido a un problema de optimización, con el vector $\boldsymbol{v}$ fijo durante este paso.

3.1.1.2 Cálculo de v

Una vez calculado $\boldsymbol{u}$, el término $\boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u}$ tiene que ser maximizado teniendo en cuenta que $\boldsymbol{v}^T \boldsymbol{v} = \boldsymbol{I}_1$. Este problema de optimización se resuelve considerando el siguiente problema de maximización con restricciones. Teniendo los vectores $\boldsymbol{v}$ y $\boldsymbol{u}$ de rango k.

$$\hat{\boldsymbol{v}} = \arg\max_{\boldsymbol{v}} \; \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u}$$

$$\text{sujeto a que} \quad \boldsymbol{v}^T \boldsymbol{v} = I \tag{3.5}$$

Por lo tanto, si se calcula la SVD de $\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u}$ se obtiene $\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{P}^T$ entonces:

$$\hat{\boldsymbol{v}} = \boldsymbol{Q}\boldsymbol{P}^T \tag{3.6}$$

3.1.2 Implementaciones del PCA Iterativo

A continuación se presentan dos formas de implementar este algoritmo para hallar las primeras k componentes principales. En la primera se buscan los vectores de proyección

uno por uno, mientras que en la segunda se buscan las matrices ***U*** y ***V*** en bloque.

3.1.2.1 PCA Deflactado

Esta primera forma de implementar el algoritmo PCA iterativo resulta de adaptar la implementación del PCA recursivo a la versión iterativa. Por lo tanto, se hallarán los vectores de proyección individualmente. Habrá un bucle por cada uno de ellos y a medida que se hallan se guardan en una matriz. El objetivo es minimizar la diferencia entre $\boldsymbol{u}_i$ y $\boldsymbol{v}_i$ en cada componente. Hay que tener en cuenta que a medida que se obtienen los vectores de proyección hay que deflactar la matriz **X** con los datos de entrada con el fin de eliminar la información que contienen de los datos en el nuevo espacio y que el siguiente vector sea independiente de las anteriores.

El pseudocódigo de este algoritmo está descrito en la Tabla 1.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. *for i = k* componentes principales
    5. Inicializar **v** aleatoriamente
    6. $dif_{uv} = 1$
    7. *while* dif_{uv} > cota
        8. Se halla **u** mediante:
        $$\hat{\boldsymbol{u}} = \min_{\boldsymbol{u}} (\boldsymbol{u}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} - 2 \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u})$$
        9. Se halla **v** mediante: SVD de $\boldsymbol{X}^T\boldsymbol{X}$ **u.**
        $\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{P}^T \rightarrow \boldsymbol{v} = \boldsymbol{Q}\boldsymbol{P}^T$
        10. Se actualiza la diferencia entre ***u*** y ***v***
        $$dif_{uv} = \sum_{i=1}^{m} (u_i - v_i) / (|\boldsymbol{u}|^2 |\boldsymbol{v}|^2)$$
    11. *end*
    12. Se normaliza **u** y se guarda en **U**
    13. Se deflacta **X:**
    $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{u}^T\boldsymbol{v}\boldsymbol{X}$
14. *end*
15. Se calculan las componentes principales mediante $\boldsymbol{Y} = \boldsymbol{U}^T\boldsymbol{X}$

Tabla 1: Pseudocódigo de la implementación algoritmo PCA Iterativo deflactado.

3.1.2.2 PCA en Bloque

En esta segunda forma de implementar el algoritmo PCA iterativo se hallan los vectores de proyección todos a la vez, por lo que la diferencia entre ***U*** y ***V***, que se usa como criterio de parada, debe evaluarse sobre todas los vectores a la vez.

El pseudocódigo para esta segunda implementación del PCA Iterativo se muestra en la Tabla 2.

Aunque el comportamiento de ambos algoritmos debería ser igual se encuentran pequeñas diferencias. En primer lugar en el algoritmo del PCA Iterativo en bloque no se controla que las componentes principales estén ordenadas según su varianza mientras que en el algoritmo deflactado si que se conoce al calcularse una por una.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. Se elige el número *k* de componentes principales a calcular
5. Inicializar **v** aleatoriamente
6. $dif_{UV} = 1$
7. *while* $dif_{UV} >$ cota
   8. Se halla **U** mediante:
      $$\hat{U} = \min_{U} \left(U^T X^T X U - 2 V^T X^T X U \right)$$
   9. Se halla V mediante: SVD de $X^T X$ **U**
      $X^T X U = QDP^T \rightarrow V = QP^T$
   10. Se actualiza la diferencia entre ***U*** y ***V***
      $$dif_{UV} = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{|U_i|^2 |V_i|^2} \sum_{j=1}^{m} u_{ij} - v_{ij}$$
11. *end*
12. Se normalizan las columnas de **U**
13. Se calculan las componentes principales mediante $Y = U^T X$

Tabla 2: Pseudocódigo de la implementación algoritmo PCA Iterativo en bloque.

Las diferencias vienen dadas no solo por la manera de implementar el algoritmo sino también por la variable diferencia. Esta variable, la cual se encuentra en ambas implementaciones, es la medida de similitud que es buscada entre los vectores de proyección ***u*** en ***U,*** y los vectores de reconstrucción ***v*** en ***V.*** Recordar que el modelo del PCA

iterativo para que sea exactamente igual al PCA original se basa en que $\boldsymbol{u}$ y $\boldsymbol{v}$ sea iguales. Sin embargo, alcanzar esta condición puede ser muy costoso en términos computacionales por lo que simplemente se busca un similitud de compromiso que sea suficiente para resolver el problema adecuadamente.

Otro factor a tener en cuenta es el coste computacional. En el PCA Iterativo deflactado se realizan p ejecuciones del PCA, una por cada componente principal que se quiera hallar. Cada una de estas ejecuciones tendrá un coste de $O(D^3)$ para el cálculo de cada $\boldsymbol{u}$ mediante mínimos cuadrados y un coste de $O(1^3)$ para el cálculo de $\boldsymbol{v}$ mediante la SVD.

Para el PCA Iterativo en bloque solo hará falta una ejecución del PCA pero al trabajar con matrices el coste será mayor en los pasos intermedios. En este caso, el cálculo de $\boldsymbol{U}$ mediante mínimos cuadrados tendrá un coste de $O(D^3)$ y el del cálculo de $\boldsymbol{V}$ mediante la SVD será de $O(p^3)$.

En cualquier caso, y aunque el coste computacional esté siempre supeditado a la elección de la diferencia mínima entre los vectores de proyección y reconstrucción, el uso de uno u otro vendrá dado por el número de dimensiones del problema, el número de vectores de proyección que se deseen obtener y el saber a priori cuantas componentes principales se van a calcular.

3.1.3 Ejemplo de aplicación del PCA Iterativo

A continuación se muestra las diferencias de las dos implementaciones del PCA iterativo a través de un ejemplo sobre una base de datos real, *SpamBase*. Esta base de datos recoge 58 variables de correos electrónicos con el fin de determinar si son correos *spam.*

En la Figura 4 están representados todos los autovalores de la matiz covarianza de los datos. Los dos primeros autovalores contienen una mayor cantidad de información relevante. La cantidad de información desciende exponencialmente desde el primer autovalor hasta el tercero mientras que a partir del tercero la cantidad de información comienza a descender linealmente.

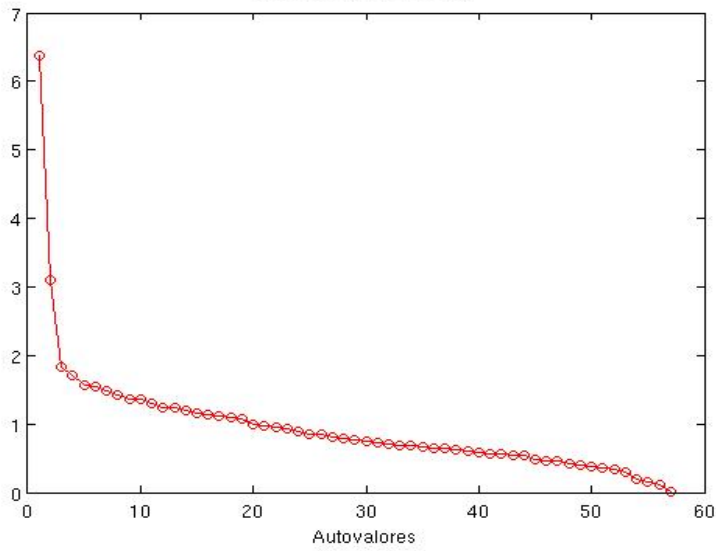

Figura 4: *Autovalores de la matriz de covarianza en la base de datos SpamBase*

En la Figura 5 se pueden observar las diferencias entre las componentes principales de ambos métodos del PCA Iterativo y las componentes del PCA original. Para esta comparación se ha utilizado la diferencia coseno.

La diferencia coseno es el coseno del ángulo que forman los dos vectores por lo que si tiene un valor próximo a la unidad significará que los vectores tienen direcciones semejantes. Mientras que si el valor es nulo los vectores serán perpendiculares. De esta forma se conoce la desviación de la dirección de cada uno de los vectores de proyección que es lo que nos interesa al fin y al cabo. La expresión para calcular la diferencia coseno de dos vectores cualesquiera **α** y **β** es la siguiente:

$$diferencia\ coseno = \frac{\alpha^T \cdot \beta}{|\alpha||\beta|} \tag{3.7}$$

Continuando con ejemplo del funcionamiento del PCA Iterativo, es necesario recordar para este análisis que el número de autovalores significativos es tres, como se ve en la Figura 4, por lo tanto, las tres primeras componentes principales tendrán la mayor cantidad de información.

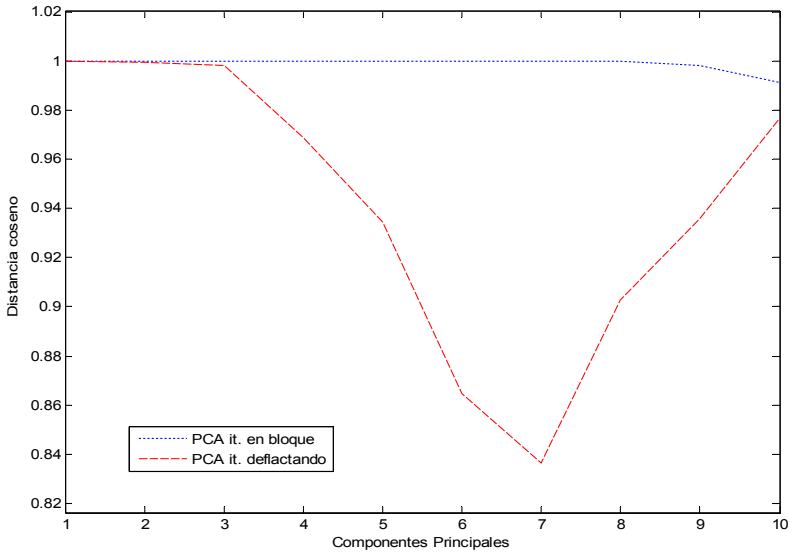

Figura 5: Diferencia coseno entre los vectores de proyección obtenidos con el PCA original y las versiones deflactando y en bloque del PCA Iterativo

Se comprueba que el PCA Iterativo en bloque es el método que produce componentes principales más cercanas al PCA original. Sin embargo, las dos últimas componentes no son iguales a las componentes del PCA original. Esto es debido a que como según se hallan más componentes, sus variables poseen un valor absoluto más pequeño. Por lo tanto, es más costoso alcanzar el valor correcto y los errores permitidos se acumulan en estas últimas componentes que pesan menos sobre el error de reconstrucción. En el caso que se hubiesen requerido más componentes los errores hubiesen pasado a las ultimas componentes como se puede ver en la Figura 6.

También se ve en la Figura 5 como las componentes principales del PCA Iterativo deflactado a partir de la tercera componente cada vez se desvían más de las componentes del PCA original. Esto de debe a que para esta implementación los errores no se acumulan en las últimas componentes principales como ocurría en la versión en bloque, sino que se permiten más errores a la hora de calcular cada componente. Además estos errores se van propagando de una componente a la siguiente en el proceso de deflacción. De todas formas se puede comprobar que este efecto aparece a partir del calculo de las componentes menos significativas y variables con valores más cercanos a cero por lo que las componentes con más información son iguales que las halladas mediante el PCA original.

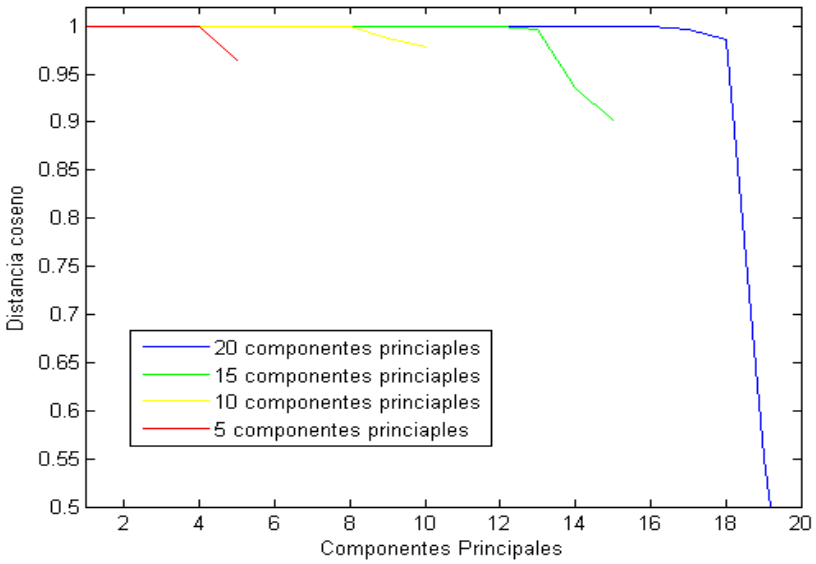

Figura 6: Variación de la diferencia entre PCA original y PCA iterativo en bloque según el número de componentes principales utilizadas.

En cuanto al tiempo de ejecución, como se observa en la Figura 7, el PCA Iterativo deflactado utiliza un tiempo menor al calcular pocas componentes principales. Para este caso el tiempo es mejor hasta que se hallan siete componentes. Cuando se requiere calcular más componentes el PCA iterativo en bloque necesita menos tiempo por lo general, aunque su evolución no es tan estable como en el PCA iterativo deflactado. Esto da ventaja al PCA en bloque porque cuando se hallan pocas componentes la diferencia con el PCA deflactado es únicamente de décimas de segundo, mientras que para hallar muchas componentes las diferencias aumentan a segundos, siendo el PCA deflactado bastante peor.

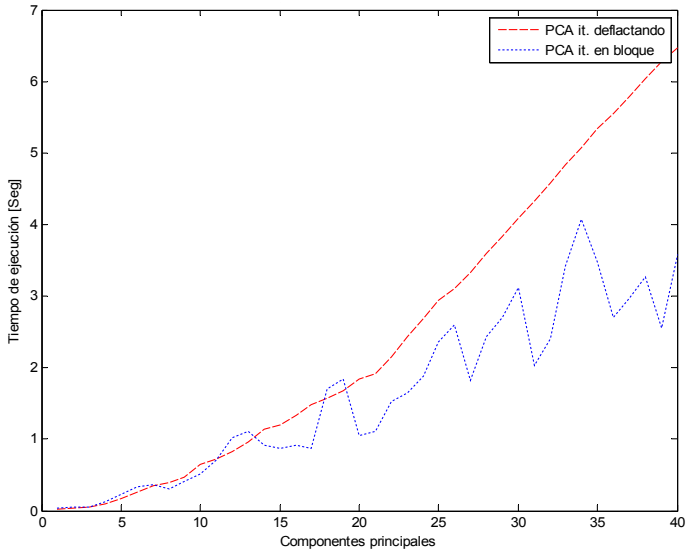

Figura 7: Tiempo de ejecución respecto al número de componentes principales calculadas.

El análisis PCA encaja perfectamente en problemas de clasificación y regresión, donde los datos de entrada tienen muchas variables. Para comprobar que este método funciona correctamente y que podemos reducir atributos sin perder información significativa aplicaremos un clasificador tras la extracción de las componentes principales. Se ha elegido regresión lineal porque es un método muy sencillo de implantar y nos puede dar una idea bastante clara de la calidad del PCA.

Para llevar a cabo la regresión lineal aplicaremos la siguiente expresión:

$$\hat{\boldsymbol{W}} = \left(\boldsymbol{Y}_e^{\boldsymbol{T}}\boldsymbol{Y}_e\right)^{-1}\boldsymbol{Y}_e^{-1}\boldsymbol{s} \tag{3.8}$$

Donde $\hat{\boldsymbol{W}}$ es la matriz de regresión, $\boldsymbol{Y}_e$ es la matriz de datos en el nuevos espacio a la que se le ha añadido una primera columna de 1 y $\boldsymbol{s}$ es el vector de etiquetas.

A continuación se presenta la evolución del error producido en la base de datos de SpamBase a medida que se utilizan un mayor número de componentes principales para formar el nuevo espacio de datos. Se han empleado tres algoritmos diferentes con el fin de poder compararlos. Por un lado tenemos el PCA original en su versión en bloque y por otro lado se han utilizado las dos implementaciones del PCA iterativo vistas en esta sección.

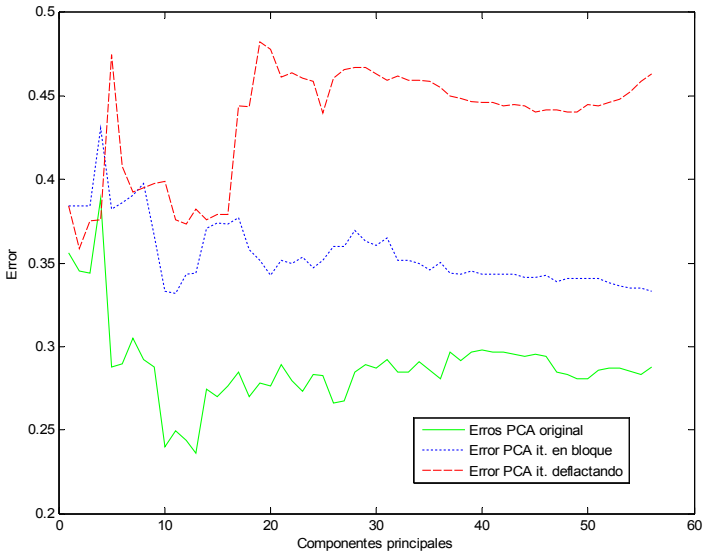

Figura 8: Evolución del error con el número de componentes principales usadas en la base de datos SpamBase para los algoritmos PCA original, PCA iterativo en bloque y PCA iterativo deflactando.

En la Figura 8 se muestra la evolución del error con el número de componentes principales utilizadas en el PCA. Se observa como a medida que se disminuyen las componentes se reduce el error hasta cierto límite que coincide con el número de los valores propios más significativos. En este caso el límite está en tres componentes principales. La utilización de más componentes principales aporta información no determinante que aumenta el error por lo que su uso es contraproducente.

Respecto a las diferencias entre las tres curvas, vemos que aunque siguen evoluciones similares la del PCA original ofrece un menor error. Esto se debe a que en el PCA iterativo hay que encontrar el compromiso entre la convergencia de $\boldsymbol{u}$ y $\boldsymbol{v}$, y el coste computacional, como se explicó anteriormente. Por lo tanto, que $\boldsymbol{u}$ y $\boldsymbol{v}$ nunca sean iguales empeora el error de las implementaciones del PCA iterativo.

Las diferencias entre el PCA en bloque y el PCA deflactado se deben a la diferencia en calcular la convergencia entre $\boldsymbol{u}$ y $\boldsymbol{v}$. En ambos algoritmos la cota mínima de convergencia es la misma, pero en el deflactado es para cada par $\boldsymbol{u}$ y $\boldsymbol{v}$, y en el en bloque es para todas las $\boldsymbol{u}$ y $\boldsymbol{v}$. Por lo que en el algoritmo en bloque la cota es más restrictiva produciendo un menor error que para el deflactado.

3.2 Sparse PCA

En esta sección se presentará la primera de las extensiones del PCA que vamos a analizar, llamada *análisis de componentes principales dispersas* ("Sparse PCA", SPCA). En este nuevo modelo se continua persiguiendo contener en las componentes principales la máxima varianza de los datos originales, pero forzando que los vectores de proyección tengan una gran parte de sus componentes nulas. De esta forma se facilita el análisis y la comprensión del modelo obtenido al depender cada componente principal de un número reducido de las variables de entrada.

3.2.1 Introducción

Hasta ahora se han mostrado varias formas de implementar el PCA que consiguen el objetivo de reducir la dimensionalidad de los datos. Sin embargo, el PCA tiene una clara desventaja y es que cada componente principal es una combinación lineal de todas las variables originales, por lo que los coeficientes de los vectores de proyección son habitualmente distintos de cero. Esto hace que la interpretación de las componentes principales sea difícil. Sin embargo, en algunas aplicaciones (v. gr. biología, genética) estos pesos tienen un significado físico y nos interesa que algunos sean nulos. En otras aplicaciones también nos interesará que los algunos pesos sean cero para reducir el coste computacional.

Con el objetivo de resolver este problema y facilitar la interpretación de las componentes principales han surgido varias técnicas. Algunos expertos han utilizado técnicas de rotación para mejorar el análisis y la comprensión de las componentes principales [9]. Otra técnica propuesta con esta finalidad fue la presentada por Vines [39], la cuál restringe el valor disponible que se le puede asignar a las cargas de las componentes principales a un pequeño conjunto de enteros, por ejemplo, -1, 0 y 1.

Sin embargo, una reducción de la dimensionalidad no es siempre suficiente por lo que también se ha buscado reducir del valor de las variables. Una forma ad hoc es artificialmente asignar un valor absoluto a las cargas menor que un umbral cercano a cero. Este método es utilizado frecuentemente en la práctica pero puede llevar potencialmente a ideas erróneas [40]. Otro método que evoluciona hacia lo que estamos buscando es el presentado por Jolliffe, Trendafilov y Uddin [41], el cual introduce el algoritmo SCoTLASS que modifica las componentes principales y permite coeficientes con valor nulo. Este algoritmo introduce una restricción que sacrifica varianza para mejorar la interpretación de las componentes principales. La restricción que proponen es un límite a la suma del valor absoluto de los coeficientes de los vectores de proyección de cada componente.

Continuando esta linea de interpretación se llega a modelos de regresión lineal múltiple, donde la respuesta es predicha por una combinación lineal de variables de entrada. La interpretabilidad de estos modelos se obtiene introduciendo un término de regularización L1 en el problema de optimización, lo que permite una selección automática de variables. En este campo hay dos técnicas principales de selección de variables con las que se está trabajando. Tibshirani [42] propuso la técnica *Lasso* ("least absolute shrinkage and selection operator") que simultáneamente produce modelos precisos y dispersos. Más adelante, Zou & Hastie [43] propusieron la técnica de la *red elástica* ("elastic net"), una generalización de la anterior, en la que se combina un término de regularización L_1 con otro de regularización L_2.

Finalmente a partir de estas dos técnicas surge una nueva aproximación al PCA que buscar obtener componentes principales con cargas dispersas. Este método propuesto por Zou et ál. [30] es en el que se va a centrar el modelo propuesto en este capítulo y se llama *Análisis de Componentes Principales Dispersas ("Sparse Principal Component Analysis ", SPCA).*

El SPCA parte de la premisa que el PCA puede ser construido como un problema de minimización del error de reconstrucción y pude aplicarse el PCA iterativo que se ha presentado en la sección anterior. Al construirse de esta forma la integración de lasso (o la

red elástica) es directa dentro de los criterios de regresión produciendo vectores de proyección con coeficientes dispersos, a cambio de sacrificar ortogonalidad en los vectores de proyección y varianza en las componentes principales.

En el siguiente apartado se explica el funcionamiento de lasso y la red elástica. Más adelante se muestra como entran a formar parte de la formulación del PCA iterativo dando como resultado el algoritmo SPCA.

3.2.2 Minimización del error cuadrático promedio con regularización L1

Para comenzar se considera un problema de regresión lineal con un conjunto de datos que tiene *n* observaciones y *p* variables. Donde $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]^T$ son las observaciones e $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n]^T$ los valores deseados a la salida del regresor para cada observación. Se asume que $\boldsymbol{X}$ y $\boldsymbol{Y}$ están centrados.

El método lasso fuerza una dispersión en los coeficientes de regresión al método de mínimos cuadrados mediante la inclusión de la regularización L_1 sobre los coeficientes de regresión. Entonces lasso obtiene los coeficientes de regresión, $\boldsymbol{u}$, resolviendo el siguiente problema de optimización:

$$\hat{\boldsymbol{u}}_{lasso} = arg\ \min_{\boldsymbol{u}} \left| Y - \sum_{i=1}^{p} \boldsymbol{X}_i \boldsymbol{u}_i \right|^2 + \lambda \sum_{i=1}^{p} |u_i| \qquad (3.9)$$

Donde λ es un valor no negativo que fuerza a los coeficientes a anularse. Según aumenta su valor, más coeficientes se anulan. Primero se anularán los que tengan un valor más reducido ya que aportan menos información.

Existe varias formas de resolver lasso. Originalmente fue resuelta mediante programación cuadrática [42]. Más tarde se comprobó que la estimación lasso como función de λ es lineal a tramos por lo que propusieron el algoritmo llamado LARS para resolver el camino entero de la solución lasso con el mismo coste computacional que un único ajuste de mínimos cuadrados [44].

El funcionamiento de lasso es sencillo ya que reduce continuamente los coeficientes hacia cero teniéndose que alcanzar un compromiso entre número de coeficientes nulos y varianza en las componentes principales. Además debido a la naturaleza de la penalización L_1, algunos coeficientes llegarán a ser exactamente cero si λ es suficientemente grande. Con

estas características lasso consigue simultáneamente precisión y un modelo disperso, creando un modelo de selección de variables bastante bueno.

Sin embargo, la formulación lasso tiene varias limitaciones [43]. La más relevante es que el número de variables seleccionadas por lasso está limitado al número de observaciones. Debido a esta limitación lasso no es adecuada para aquellos conjuntos de datos que tengan más variables que observaciones.

Con el fin de solucionar esta limitación se propuso la formulación de *la red elástica* [43], que es una generaliza la formulación lasso introduciendo un término de regularización L_2 adicional al L_1, por lo que tiene similares características en el proceso de optimización. Ahora tendremos λ_1 y λ_2 como parámetros no negativos para la estimación de la red elástica, y los coeficientes de regresión vendrán dados por la resolución de :

$$\hat{\boldsymbol{u}}_{en} = arg\min_{\boldsymbol{u}} \left| Y - \sum_{i=1}^{p} \boldsymbol{X}_i \boldsymbol{u}_i \right|^2 + \lambda_2 \sum_{i=1}^{p} |u_i|^2 + \lambda_1 \sum_{i=1}^{p} |u_i| \qquad (3.10)$$

Como se puede observar la red elástica es una combinación convexa de la regularización L_2 de Tikhonov [45] (aplicada usualmente a problemas mal condicionados) y la penalización lasso. Obviamente, la lasso es un caso especial del red elástica para λ_2 nula. Con este modelo pueden ser incluidas potencialmente todas las variables incluso en problemas con más dimensiones que datos por lo que la red elástica resuelve la limitación de la lasso.

3.2.3 Modelo del SPCA

El modelo algorítmico que construye el SPCA comienza con el mismo desarrollo que el modelo del PCA iterativo. Se parte de un problema de optimización de una regresión en la que se hace exactamente el PCA. El cual permite la modificación directa mediante lasso (red elástica) con el objetivo de obtener las variables dispersas [30].

Se partirá de la expresión (3.2), a la que se le añade la penalización lasso:

$$(\hat{\boldsymbol{v}}, \hat{\boldsymbol{u}}) = arg\min_{\boldsymbol{v},\boldsymbol{u}} \sum_{i=1}^{n} \left| \boldsymbol{X}_i - \boldsymbol{v}\boldsymbol{u}^T \boldsymbol{X}_i \right|^2 + \sum_{j=1}^{p} \lambda_{1,j} |u_j|_1$$

$$\text{sujeto a que } \boldsymbol{v}^T \boldsymbol{v} = \boldsymbol{I}_p. \qquad (3.11)$$

Donde p es el número de componentes principales que son buscadas. Hay que tener en cuenta que, al igual que en la expresión (3.2), la resolución de la expresión (3.11) está

determinada por la condición de convergencia entre los valores de **u** y ***v***.

Nótese que el PCA Iterativo es una caso particular de (3.9), al hacer $\lambda_1=0$ se obtiene (3.2).

Otra consideración a la que hay que hacer referencia es que en la expresión aquí planteada se está suponiendo que los problemas que se van a estudiar el número de observaciones siempre será mayor que el número de variables por lo que solo se ha añadido la penalización lasso de tipo L_1. En caso contrario habría que añadir la penalización de la red elástica completa, quedando la expresión siguiente:

$$(\hat{\boldsymbol{v}},\hat{\boldsymbol{u}}) = arg\min_{\boldsymbol{v},\boldsymbol{u}} \sum_{i=1}^{n} \left|\boldsymbol{X}_i - \boldsymbol{v}\boldsymbol{u}^T\boldsymbol{X}_i\right|^2 + \sum_{j=1}^{k} \lambda_{1,j}\left|u_j\right|_1 + \lambda_2 \sum_{i=1}^{p} \left|u_i\right|^2$$

sujeto a que $\boldsymbol{v}^T\boldsymbol{v} = \boldsymbol{I}_k$. (3.12)

Como en el PCA Iterativo, para resolver (3.12) se emplearán dos pasos. En el primero se calculará **u** y en el segundo **v**. A continuación se propone las soluciones numéricas a estos dos pasos y se proseguirá con las dos soluciones alternativas de implementar el algoritmo SPCA: de forma iterativa o en bloque.

3.2.3.1 Cálculo de **u**

En este apartado se propone un desarrollo matemático para que el SPCA calcule **u** simplificando el problema de la optimización. La solución numérica que se muestra a continuación es una evolución de la solución numérica mostrada en la sección anterior. La principal diferencia es que se ha añadido el término de penalización lasso que también forma parte de la optimización en este caso.

Ahora partimos de la expresión (3.12):

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} \left|\boldsymbol{X}_i - \boldsymbol{v}\boldsymbol{u}^T\boldsymbol{X}_i\right|^2 + \sum_{j=1}^{k} \lambda_{1,j}\left|u_j\right|_1 \\ &= (\boldsymbol{v}-\boldsymbol{u})^T\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{v}-\boldsymbol{u}) + \textstyle\sum_{j=1}^{k} \lambda_{1,j}\left|u_j\right|_1 \\ &= \boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} - \boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} - \boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} + \boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} + \textstyle\sum_{j=1}^{k} \lambda_{1,j}\left|u_j\right|_1 \end{aligned} \quad (3.13)$$

Si se eliminan los términos que no van a ser optimizados, es decir, aquellos que no contengan **u** y se fija **v**. La expresión que tiene que optimizada queda de la siguiente forma:

$$\min_{\boldsymbol{u}} \ \boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} - 2\boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} + \sum_{j=1}^{k} \lambda_{1,j}\left|u_j\right|_1 \quad (3.14)$$

3.2.3.2 Cálculo de **v**

Una vez obtenido **u** se fija su valor para poder maximizar el término $\boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{u}$ teniendo en cuenta que $\boldsymbol{v}^T\boldsymbol{v} = \boldsymbol{I}_p$.

Este último problema de optimización se resuelve exactamente igual que en el apartado 3.1.1.2. Se calculará la SVD de $\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{u}$ y se hallará **v** mediante (3.6).

3.2.4 Implementaciones del SPCA

A continuación se presentan dos formas de implementar el algoritmo SPCA para hallar las primeras *k* componentes principales con cargas dispersas. En la primera se buscan las componentes una a una mientras que en la segunda se buscan todas a la vez en bloque, al igual que ocurre para el PCA iterativo.

Como ya se explicó cuando se estudio el algoritmo PCA normal, la solución que produce este método cumple el principio de ortogonalidad entre todas las componentes principales. Esto es necesario para que cada componentes principales contenga información exclusiva y no esté compartida entre varias.

Sin embargo, el algoritmo SPCA explicado en el apartado anterior rompe con esta restricción de ortogonalidad para permitir que la información de las variables en las componentes dispersas pueda ser nula. Al añadir la formulación lasso en (3.11) el SPCA fuerza que los coeficientes en **u** se anulen, mientras que se mantiene la ortogonalidad de **v**. Por este motivo y pese a que se busque que **u** y **v** sean semejantes nunca llegarán a ser exactamente iguales y las componentes de **U** no será ortogonales.

No obstante existe otra forma de forzar que **U** sea ortogonal gracias al SPCA deflactando. Este algoritmo ejecuta el SPCA componente a componente. Cada vez que se obtiene una componente principal dispersa se elimina la información que se consigue a partir de ella de los datos originales del problema y se procede con la siguiente componente dispersa. De esta forma una componente no puede contener información ya obtenida mediante una componente calculada anteriormente por lo que los vectores de proyección mediante SPCA deflactando son ortogonales.

La otra forma de implementar el SPCA es en bloque, es decir, aplicando el algoritmo SPCA tal cual. Ambas implementaciones llegan al mismo espacio vectorial, pero lo hacen a través de vectores de proyección diferentes.

Las ventajas de utilizar el SPCA deflactado son varias:

- Por un lado se obtiene lo mejor de cada uno de los algoritmos PCA y SPCA porque, como ya se ha explicado anteriormente, se obtienen componentes ortogonales que además tienen variables dispersas. Originando que la interpretación de las componentes principales sea mucho más sencilla.
- Por otra parte, el SPCA deflactado asegura un convergencia mínima para cada componente al tratar cada componente por separado en el algoritmo. Esto no ocurre para el SPCA en bloque ya que la convergencia se estudia conjuntamente para todas las componentes, lo que puede producir pequeños desajustes y resultados menos exhaustivos.
- Sin embargo, el tratar cada componente de manera secuencial y no en paralelo como ocurre en el SPCA en bloque, se exige al algoritmo SPCA deflactado realizar una optimización de la convergencia por cada componente en vez de una sola optimización para todas las componentes. Esto produce un mayor tiempo de ejecución del algoritmo, al ser una condición más restrictiva, supeditado siempre a la complejidad de cada problema.

3.2.4.1 SPCA Deflactado

En esta primera forma de implementar el algoritmo SPCA iterativo se hallan las componentes principales una a una. Habrá un bucle por cada componente principal y a medida que se hallan se guardan en una matriz. El objetivo es minimizar la diferencia de cada $\mathbf{u}$ con cada $\boldsymbol{v}$ para cada componente principal.

El pseudocódigo para este algoritmo se muestra en la Tabla 3.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. *for i = k* componentes principales
    5. Inicializar **v** aleatoriamente
    6. $dif_{uv} = 1$
    7. *while* dif_{uv} > cota
        8. Se halla **u** mediante:

$$\hat{\boldsymbol{u}} = \min_{\boldsymbol{u}} \left(\boldsymbol{u}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} - 2 \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{u} \right) + \sum_{j=1}^{k} \lambda_{1,j} \left| u_j \right|_1$$

        9. Se halla **v** mediante: SVD de $\boldsymbol{X}^T\boldsymbol{X}$ **u**
        $\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{P}^T \rightarrow \boldsymbol{v} = \boldsymbol{Q}\boldsymbol{P}^T$
        10. Se actualiza la diferencia entre $\boldsymbol{u}$ y $\boldsymbol{v}$

$$dif_{uv} = \sum_{i=1}^{m} (u_i - v_i) / (|\boldsymbol{u}|^2 |\boldsymbol{v}|^2)$$

    11. *end*
    12. Se normaliza **u** y se guarda en **U**
    13. Se deflacta **X:** $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{u}^T\boldsymbol{v}\boldsymbol{X}$
14. *end*
15. Se calculan las componentes principales mediante $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$

Tabla 3: Pseudocódigo de la implementación algoritmo SPCA deflactado.

3.2.4.2 SPCA en Bloque

En esta segunda forma de implementar el algoritmo SPCA iterativo se hallan las componentes principales todas a la vez. Para ello se requiere la preparación de una matriz que contenga los datos en su diagonal tantas veces como componentes se quieran hallar. Ahora al calcularse todas las componentes principales simultáneamente la diferencia entre **u** y **v** se evalúa para todas las componentes juntas.

El pseudocódigo para este algoritmo queda reflejado en la Tabla 4.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. Se elige el número *k* de componentes principales a calcular
5. Inicializar **v** aleatoriamente
6. $dif_{UV} = 1$
7. *while* $dif_{UV} >$ cota
    8. Se halla **U** mediante:

$$\hat{U} = \min_{U} \left(U^T X^T X U - 2 V^T X^T X U \right) + \sum_{j=1}^{k} \lambda_{1,j} \left| u_j \right|_1$$

    9. Se halla V mediante: SVD de $X^T X$ U
    $X^T X U = QDP^T \rightarrow V = QP^T$
    10. Se actualiza la diferencia entre U y V

$$dif_{UV} = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{|U_i|^2 |V_i|^2} \sum_{j=1}^{m} u_{ij} - v_{ij}$$

11. *end*
12. Se normalizan las columnas de **U**
13. Se calculan las componentes principales mediante $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$

Tabla 4: Pseudocódigo de la implementación algoritmo SPCA en bloque.

3.3 Ɛ-PCA

A lo largo de esta sección se desarrollará una nueva versión del PCA. El propósito del nuevo algoritmo es la mejora del PCA mediante la introducción de una nueva función de coste más robusta que permitirá insensibilizar el funcionamiento de esta versión del ruido impulsivo.

3.3.1 Introducción al Ɛ-PCA

En este capítulo se va a proceder con otra variante del PCA en la cual se posee un mejor insensibilidad frente al ruido de los datos gracias a una nueva función de coste. En este caso se va tratar el *análisis de componentes principales* como si fuera un problema de minimización del error de reconstrucción con una función de coste más robusta frente al ruido impulsivo.

Es posible encontrar planteamientos parecidos que buscan añadir robustez al PCA [32], [33], [34], [35] y [36]. En [37] se presentan dos métodos que resuelven problemas no-lineales con el kernel PCA usando una función de coste "*ε-insensible".* En el primer método resuelven las ecuaciones mediante programación cuadrática secuencial ("sequential quadratic programming", SQP) como un problema de autovalores y autovectores; y en el segundo propone un esquema simplificado de pesos para resolver un problema de autovalores generalizados iterativos.

A partir de esta idea, pero empleando un procedimiento diferente con una formulación lineal se desarrollará una nueva versión del PCA. A este nuevo algoritmo se le llama Ɛ-PCA porque la función de coste que se utilizará será del tipo "*ε-insensible"* y el parámetro más importante es la tolerancia al error *ε*. Gracias a este parámetro se conseguirá reducir el impacto del error o ruido impulsivo asociado a los datos, como se explicará en el siguiente apartado.

Para la creación del modelo Ɛ-PCA se ha partido del algoritmo PCA Iterativo propuesto en la sección 3.1. Se ha planteado un desarrollo similar a este algoritmo salvo con la diferencia de la implementación del coste en el problema de minimización del error de reconstrucción. La función de coste "*ε-insesitive"* que será añadida es utilizada con gran asiduidad de las *máquinas de vectores soporte* y en especial en su variante en regresión. Esta parte es donde radica la principal diferencia con las extensiones anteriores.

Las *máquinas de vectores soporte* ("Support Vector Machine", SVM) tiene como objetivo realizar una clasificación o regresión dependiendo del problema en cada caso. Poseen un funcionamiento muy eficiente y un comportamiento robusto frente a errores [46]. Estos motivos hacen que esta técnica a priori, a diferencia de otras, pueda llevar a cabo la resolución de un problema de optimización de forma más adecuada presentando una mejor generalización. Por este motivo, nos proponemos averiguar si la función de coste utilizada por este algoritmo puede mejorar los resultados de las distintas variantes del PCA.

En el siguiente apartado se explica en profundidad como funcionan las técnicas SVM en regresión y donde encaja el análisis de componentes principales. A continuación de este apartado, se expondrá el modelo implementado para el Ɛ-PCA.

3.3.2 SVM en Regresión

Las Máquinas de Vectores Soporte (SVM) son una clase de algoritmos diseñados para

resolver problemas de aprendizaje máquina de tipo supervisado. Una de sus principales características es su buena capacidad de generalización, lo que ha hecho que se hayan empleado en multitud de aplicaciones [47].

La historia de los algoritmos de vectores soporte comienza en los años sesenta del siglo XX en Rusia con el trabajo de Vapnik, Lerner y Chervoneskis sobre teoría del aprendizaje estadístico [48] y [49]. En la forma actual, las SVM han sido desarrolladas en los laboratorios AT&T Bell Laboratories por Vapnik y sus ayudantes. Debido a este entorno de desarrollo las SVM están fuertemente orientadas a aplicaciones reales. El principal campo donde fueron aplicados y donde son sistemas altamente competitivos es en el ámbito del reconocimiento óptico de caracteres [47], después su uso se ha extendido a multitud de áreas de trabajo. Su aplicación está ligada principalmente a tareas de clasificación, regresión, detección de novedad, versiones online, etc; ya sea en sistemas lineales o no lineales con la introducción de funciones kernel. Entre estás versiones nos centraremos en la máquina de vectores soporte para regresión ("Support Vector Regression", SVR).

El SVR posee una estructura similar a la del coste "ε-insensible" que se va a añadir al PCA por lo que es interesante conocer el funcionamiento de esta técnica. Para presentar el modelo vamos a considerar que tenemos un conjunto de datos de entrenamiento de la siguiente forma $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\} \in \mathbb{R}^D \times \mathbb{R}$, donde $\boldsymbol{X}$ son las observaciones o datos del espacio de entrada e $\boldsymbol{Y}$ las variables a estimar en el espacio de salida.

En la SVR [50], el objetivo es encontrar una función $\hat{y} = f(x)$ la cual tenga para cada dato de entrenamiento a ε como desviación máxima a la hora de hallar $\hat{y}$, y al mismo tiempo sea lo más plana posible. En otras palabras, no se permiten desviaciones o errores más grandes que ε, mientras que si son menores que este valor son aceptadas. De esta forma se asegura que para cualquier observación la máxima desviación entre el valor de salida de la máquina y la etiqueta sea ε [47].

$$|\zeta|_\varepsilon := \begin{cases} 0 & si \quad |\zeta| \leq \varepsilon \\ |\zeta| - \varepsilon & en\ el\ resto \end{cases} \tag{3.15}$$

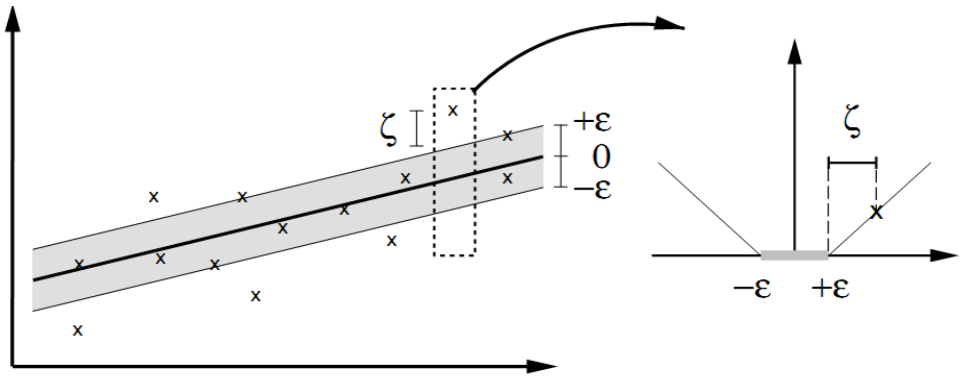

Figura 9: Función ε-insensible para SVM [47].

A continuación se van a describir la formulación lineal de la SVR. Este caso es el más sencillo y, además, es el que va a ser usado para implementar el Ɛ-PCA. Las funciones lineales toman la siguiente forma:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b \quad donde \quad \boldsymbol{w} \in \mathbb{R}^D,\ b \in \mathbb{R} \tag{3.16}$$

Con el objetivo de que la función se lo más suave posible[2], $|\boldsymbol{w}|^2$ tiene que ser lo más pequeña posible. Una forma de conseguir este propósito es incluir la minimización de este término en el problema de optimización. Teniendo en cuenta que la salida de la máquina viene dada por (3.16) y que según el coste "ε insensitive" los errores de estimación deben ser menores que ε, la SVR resuelve el siguiente problema de optimización:

$$\hat{\boldsymbol{w}} = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{sujeto a} \quad \begin{cases} y_i - \boldsymbol{w}^T \boldsymbol{x}_i - b \leqslant \varepsilon \\ \boldsymbol{w}^T \boldsymbol{x}_i + b - y_i \leqslant \varepsilon \end{cases} \quad i = 1, 2, \ldots, N \tag{3.17}$$

Se asume tácitamente que en (3.17) la función $f(\cdot)$ existe y que aproxima el valor de $f(x_i)$ *a* y_i para $\forall i$ con una precisión de ε, es decir, que el problema de optimización convexa es resoluble. Sin embargo, a veces no es de esta manera o se permiten errores. Por este motivo se introducen las restricciones laxas ζ_i , ζ_i^* y así poder manejar las posibles restricciones inviables de los problemas de optimización de (3.17).

Este desarrollo es análogo a la función de perdidas de "margen blando" [51] las cuales fueron adaptadas más tarde a las SVM [52]. Y nos permite llegar a la formulación propuesta por Vapnik [50]:

2 La suavidad de la función va asociada a su generalización, habitualmente, funciones suaves (con el menor número de curvas) suelen generalizar mejor.

$$\hat{\boldsymbol{w}} = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*)$$

$$\text{sujeto a} \quad \begin{cases} y_i - \boldsymbol{w}^T\boldsymbol{x} - b \leqslant \varepsilon + \zeta_i \\ \boldsymbol{w}^T\boldsymbol{x} + b - y_i \leqslant \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geqslant 0 \end{cases} \tag{3.18}$$

La constante C > 0 determina el compromiso entre la suavidad de *f(·)* y la cantidad de errores o desviaciones mayores de ε permitidos. Esto corresponde con el comportamiento de la función de perdidas ε-insensible descrita anteriormente por (3.15).

En la Figura 9, presentada al inicio de esta sección, se puede observar el funcionamiento de (3.16). Los puntos dentro de la zona marcada tienen un error tolerable al estar a una distancia menor que ε. Mientras que los puntos fuera de la zona marcada contribuyen al coste que penaliza al algoritmo. Esta función de coste permite resolver un problema de regresión mediante un problema de optimización con restricciones sobre los datos. A diferencia del típico error cuadrático medio, este coste penaliza linealmente (no de forma cuadrática) las desviaciones más allá de ε, lo que hace que sea más robusto en presencia de ruido impulsivo.

También con esta formulación se pueden adaptar de manera más sencilla las SVM a funciones no-lineales mediante el uso de kernels [53]. Sin embargo, en este proyecto solo se va a trabajar con funciones lineales por lo que la aproximación SVM a las funciones no-lineales queda fuera del ámbito (objetivo) de este proyecto.

3.3.3 Modelo del Ɛ-PCA

Al igual que ocurre con el SPCA, el Ɛ-PCA parte de la solución del PCA Iterativo presentada en el apartado 3.1 y las modificaciones radican en la forma de resolver el problema de optimización en su paso de regresión. En esta parte es donde se modifica la función de coste para resolver el problema de regresión. Como en el SPCA, se sustituye la expresión (3.2) por otra con las características del método que vamos a utilizar en este caso. Por lo tanto, ahora se tiene que adaptar (3.18) para resolver el PCA, es decir, tendrá que minimizar el vector **u** de proyección de los datos.

Cuando se adapta (3.18) al caso del PCA hay que tener en consideración que se busca que ***u*** *y* ***v*** tengan la mayor similitud para que los datos de entrada ***X*** al transformarse en otra dimensión ***Y*** puedan ser recuperados con la menor pérdida de información posible. En primer lugar hay que tener en cuenta que se va a minimizar ***u***, por lo que ***u*** sustituye a ***w***. En

segundo lugar, el resultado final $\boldsymbol{y}_i$ es equivalente $\boldsymbol{v}\boldsymbol{x}_i$. Por último, no hay que olvidarse del término *b* para equilibrar los datos (eliminar el sesgo del estimador). Para este proyecto se está trabajando con datos normalizados antes de entrar en el algoritmo por lo que *b* no es necesaria.

Si se añaden todas las modificaciones anteriores a (3.16) resulta la siguiente expresión que sustituye a (3.2) para el Ɛ-PCA:

$$\min_{\boldsymbol{u}} \ \frac{1}{2}\|\boldsymbol{u}\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*)$$

$$\text{sujeto a} \quad \begin{cases} \boldsymbol{v}\boldsymbol{x}_i - \boldsymbol{u}^T\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i \\ \boldsymbol{u}^T\boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geqslant 0 \end{cases} \tag{3.19}$$

Durante todo este proyecto se están analizando diferentes modelos del PCA, que hasta ahora buscaban únicamente una buena precisión o una mejor interpretación de los resultados sin preocuparse ni del coste ni del ruido asociado a los datos. Por lo tanto, los parámetros que podían ser variados en las realizaciones de cada uno de los algoritmos anteriores eran, dependiendo del tipo de modelo: el número de componentes principales, la diferencia mínima entre $\boldsymbol{u}$ y $\boldsymbol{v}$ para la convergencia del algoritmo, λ de dispersión (λ_1) y λ de doble regularización (λ_2). Con la utilización de esta nueva función de coste en el PCA se manejaran dos parámetros más, ε y C, que permitirán un gobierno más exhaustivo del comportamiento del algoritmo.

En este modelo también existen dos métodos de implementarlo: deflactado o en bloque. Sin embargo, las diferencias entre ambos métodos es menor que en los casos precedentes porque debido a las características de la optimización SVM siempre se minimizarán las $\boldsymbol{u}$ una por una. La diferencia entre ambos métodos se encontrará en la manera de estudiar las diferencias entre $\boldsymbol{u}$ y $\boldsymbol{v}$, si se calculan una por una, deflactado, o todas en conjunto, en bloque.

También se puede resolver en bloque pero:

- Si al aplicar en bloque las componentes no son ortogonales.
- La formulación tiene mayor dificultad. Al hacerlo iterativo, en cada paso se resuelve una única SVR, con la posibilidad de usar toolboxes adaptadas a ello.

A continuación, en las Tablas 5 y 6 se presentan los algoritmos del Ɛ-PCA deflactado y en bloque.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$, Coste = 10, $\varepsilon = 1 \cdot 10^{-3}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. *for* i = k componentes principales
    5. Inicializar **v** aleatoriamente
    6. $dif_{uv} = 1$
    7. *while* dif_{uv} > cota
        8. Se halla **u** mediante:

$$\hat{\boldsymbol{u}} = \min_{\boldsymbol{u}} \frac{1}{2}\|\boldsymbol{u}\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \quad \text{s.t.} \quad \begin{cases} \boldsymbol{v}\boldsymbol{x}_i - \boldsymbol{u}^T\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i \\ \boldsymbol{u}^T\boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geqslant 0 \end{cases}$$

        9. Se halla **v** mediante: SVD de $\boldsymbol{X}^T\boldsymbol{X}$ **u**
        $\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{P}^T \rightarrow \boldsymbol{v} = \boldsymbol{Q}\boldsymbol{P}^T$
        10. Se actualiza la diferencia entre ***u*** y ***v***

$$dif_{uv} = \sum_{i=1}^{m}(u_i - v_i)/(|\boldsymbol{u}|^2|\boldsymbol{v}|^2)$$

    11. *end*
    12. Se normaliza ***u*** y se guarda en **U**
    13. Se deflacta **X:** $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{u}^T\boldsymbol{v}\boldsymbol{X}$
14. *end*
15. Se calculan las componentes principales mediante $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$

Tabla 5: Pseudocódigo de la implementación algoritmo $\mathcal{E}$-PCA deflactado.

1. Constantes iniciales: cota = $1 \cdot 10^{-6}$
2. Partimos de la matriz de datos **X**
3. Se elimina la media de **X**
4. Se elige el número *k* de componentes principales a calcular
5. Inicializar **v** aleatoriamente
6. $dif_{UV} = 1$
7. *while* dif_{UV} > cota
    8. *for* i = k componentes principales
        9. Se halla **U** mediante:

$$\hat{\boldsymbol{u}} = \min_{\boldsymbol{u}} \frac{1}{2}\|\boldsymbol{u}\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \quad \text{s.t.} \quad \begin{cases} \boldsymbol{v}\boldsymbol{x}_i - \boldsymbol{u}^T\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i \\ \boldsymbol{u}^T\boldsymbol{x}_i - \boldsymbol{v}\boldsymbol{x}_i \leqslant \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geqslant 0 \end{cases}$$

        10. Se normaliza $\boldsymbol{u}$ y se guarda en **U**
    11. *end*
    12. Se halla V mediante: SVD de $\boldsymbol{X}^T\boldsymbol{X}$ **U** ($\boldsymbol{X}^T\boldsymbol{X}\,\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{P}^T$) → $\boldsymbol{V} = \boldsymbol{Q}\boldsymbol{P}^T$
    13. Se actualiza la diferencia entre **U** y **V**

$$dif_{UV} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|\boldsymbol{U}_i|^2|\boldsymbol{V}_i|^2}\sum_{j=1}^{m} u_{ij} - v_{ij}$$

14. *end*
15. Se calculan las componentes principales mediante $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$

Tabla 6: Pseudocódigo de la implementación algoritmo Ɛ-PCA en bloque.

Capítulo 4

Experimentos SPCA

A continuación se presentan los experimentos realizados utilizando el SPCA. En general tienen como objetivo analizar el funcionamiento del SPCA y sus características principales. Además se confrontarán los resultados obtenidos mediante el SPCA con los del PCA normal en cada problema.

Se van a realizar tres tipos de experimentos:

- En el primero de ellos se comparará la información obtenida por cada uno de los algoritmos PCA y SPCA mediante un ejemplo sintético. También se estudiará la influencia de la inicialización en las prestaciones del SPCA en este primer experimento.
- En el segundo experimento se verificará el funcionamiento del SPCA para los casos especiales en los que se disponga de menos observaciones que número de variables. En este experimento también se emplearán datos generados *ad hoc*, para tal propósito.
- Finalmente se presentará un problema de clasificación con datos reales en los que se verificarán los resultados obtenidos utilizando tanto los métodos PCA y SPCA como sin usarlos.

Las simulaciones de dichos experimentos se llevaran a cabo en MATLAB. Además para resolver el problema de optimización que compone el SPCA se utilizará un software

específico para optimización, llamado Mosek [54]. Mosek es una herramienta que permite la resolución eficiente de problemas de optimización matemática. Para la resolución de la del primer paso del SPCA se empleará una función particular para programación cuadrática.

Una consideración dentro del tipo de algoritmo que se va utilizar en estos experimento es el tipo de normalización empleada en los datos originales del problema. Por una parte hay que realizar siempre la normalización en media sobre los datos de entrada debido a que la solución del SPCA presentada en el apartado 3.2.3 está pensada para datos con media nula.

Por otra parte es muy conveniente normalizar en varianza los datos de entrada en el algoritmo con el fin de minimizar la amplitud de estos. A la hora de normalizar en varianza se han empleado dos métodos durante los experimentos presentados en este proyecto.

1. Normalizar la varianza por variables. Se normaliza separadamente cada variable sobre su propia varianza.
2. Normalizar la varianza de conjuntamente de todos los datos. En este caso se utiliza la varianza global de todos los datos de entrada.

En principio es mejor normalizar la varianza conjuntamente porque se trata a todas las variables de igual manera y entre los datos normalizados se mantendrán las proporciones de varianza entre las variables. Si se normaliza por variables todas las variables tendrán la misma varianza al contrario que pasaba en los datos de entrada.

Sin embargo, en la práctica usar un método u otro de normalizar no varia los resultados finales del SPCA si se seleccionan correctamente los parámetros. El parámetro λ_1 tendrá un rango mayor de posibles valores cuando se normaliza conjuntamente en vez por variables. Debido a esto para el experimento 4.1 se ha empleado el método uno, pero en los apartados 4.2 y 4.3 se empleará el método dos.

4.1 Simulación con datos artificiales

A continuación se va a generar un modelo artificial que nos permita comparar el funcionamiento del PCA con del SPCA. Aunque esta simulación fue propuesta por primera vez en [30], se desarrollará a partir de la presentación realizada por d'Aspremont et al. en [55].

En primer lugar se crean tres factores ocultos, V_1, V_2 y V_3, según:

$$V_1 \sim N(0, 290), \qquad V_2 \sim N(0, 300)$$

$$V_3 = -0.3V_1 + 0.925V_2 + \varepsilon, \qquad \varepsilon \sim N(0, 300)$$

V_1, V_2 y ε son independientes.

A continuación se generan 10 variables de la siguiente forma:

$$X_i = V_1 + \varepsilon^1_{i,} \qquad \varepsilon_i^1 \sim N(0, 1), \qquad i = 1,2,3,4,$$

$$X_i = V_2 + \varepsilon^2_{i,} \qquad \varepsilon_i^2 \sim N(0, 1), \qquad i = 5,6,7,8,$$

$$X_i = V_3 + \varepsilon^3_{i,} \qquad \varepsilon_i^3 \sim N(0, 1), \qquad i = 9,10,$$

$\{\varepsilon^j_i\}$ son independientes, $j = 1,2,3 \;\; i = 1,...,10$.

Para realizar el experimento se usa la matriz exacta de covarianzas de $(X_1, ..., X_{10})$ con el objetivo de evitar la aleatoriedad entre las diferentes realizaciones de PCA y SPCA.

Los tres factores ocultos, V_1, V_2 y V_3, están asociados con 4, 4 y 2 variables respectivamente, mientras que su varianza es: 290, 300 y 283.8. Por lo tanto, V_2 es el factor con mayor importancia seguido de cerca por V_1 y quedando V_3 como el factor con menor importancia, a bastante distancia de los otros dos. Al aplicar el PCA original a los datos de este modelo las dos primeras componentes principales halladas explican conjuntamente el 99.6% de la varianza total. Esto es debido a que las diez variables están generadas a partir de dos factores ocultos independientes, V_1 y V_2. Por lo tanto, solamente son necesarias dos combinaciones lineales dispersas de los datos originales que recojan la información de estos dos factores para explicar la mayoría de la información de los datos de este modelo [30]. Es decir, con dos componentes principales dispersas se obtiene la mayor parte de la varianza de los datos originales. La solución ideal recogería únicamente en el primer vector de proyección las variables asociadas al factor V_2 (X_5, X_6, X_7, X_8) y en el segundo vector de proyección se recogería el factor V_1 usando las variables (X_1, X_2, X_3, X_4). Usando la matriz de covarianza exacta y la predicción de que la representación dispersa ideal utiliza solo cuatro variables, se ejecuta el algoritmo SPCA sobre los datos del modelo. En primer lugar hay que seleccionar los parámetros del algoritmo (λ_2, λ_1). Como los datos del modelo son n>>m entonces $\lambda_2 = 0$. La elección de λ_1 estará basada en las gráficas de la Figura 11, 12 y 13. Se intenta que cada PC dispersa tenga la mayor cantidad de varianza, mientras se mantienen los criterios de dispersión. Es decir, que la primera componente dispersa utilice las cuatro

variables del factor V_2 y la segunda componente utilice las cuatro variables del factor V_1.

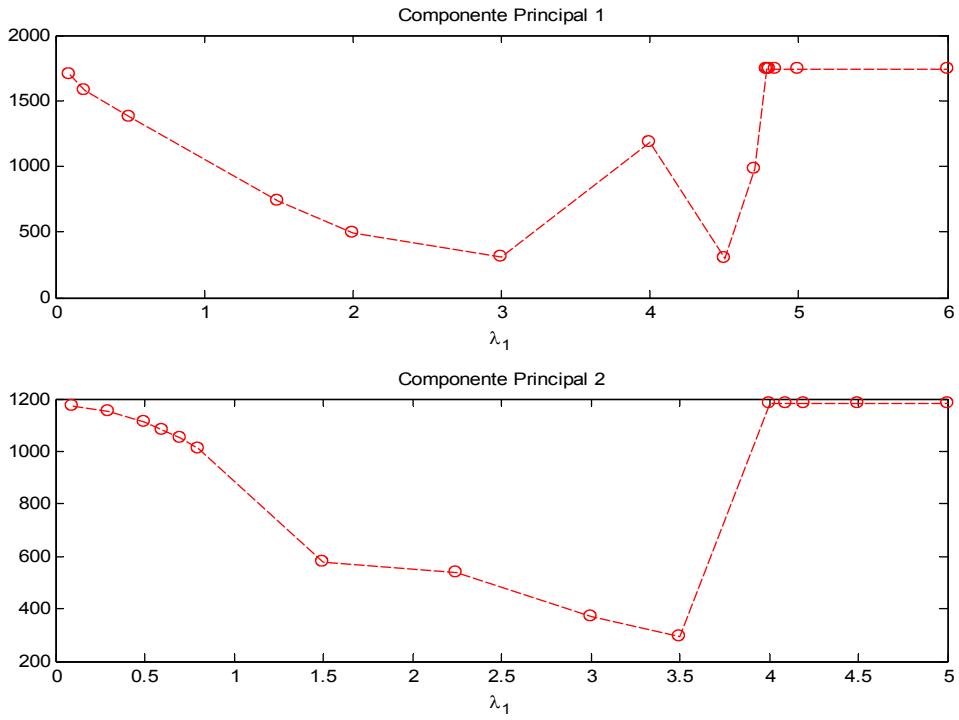

Figura 10: Varianza de las componentes 1 y 2 según la λ_1 utilizada.

En la Figura 10 se ha representado la evolución de la varianza según la λ_1 empleada. Se pueden observar claramente los puntos exactos donde se ha calculado la varianza para que de esta forma se pueda elegir fácilmente la λ_1 más apropiada.

Sin embargo, como se ha explicado anteriormente, también hay que tener en cuenta el número de valores nulos y no nulos de los coeficientes en cada vector de proyección, para obtener solo las cargas de los factores asociados a cada componente.

Para estudiar los cargas nulas, en la Figura 11 se muestra la evolución del valor de cada coeficiente para cada una de los puntos λ_1 usados en la Figura 10. Los valores de las cargas están codificados en colores para un análisis más sencillo. Los colores fríos (azules) representan un valor nulo o cercano a cero mientras que los colores calientes (rojo) representan valores cercanos a 1 dependiendo de su intensidad. Se han representados valores absolutos porque para este análisis es indiferente un valor de +1 o -1 dado que lo que nos interesa es conocer los valores que son nulos para cada λ_1 . El color amarillo que poseen bastantes coeficientes para algún valor de λ_1 representa un valor de 0.5.

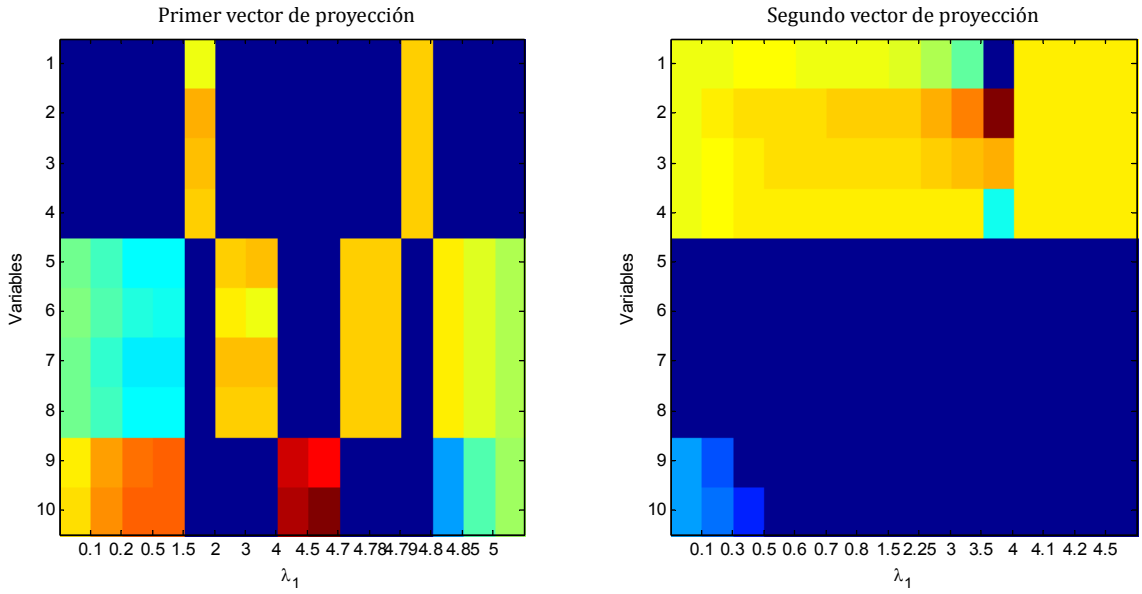

Figura 11: Evolución de los valores, codificados en colores, de las variables para diferentes λ_1 .

Finalmente en la Figura 12 recoge la información obtenida en las Figuras 10 y 11 mostrando el número de valores no nulos en cada vector de proyección, así como el porcentaje de varianza ajustada para casa componente. Se puede observar como la evolución del número de coeficientes no nulos y del valor del porcentaje de varianza explicada está íntimamente relacionada, ya que cuantas más cargas no nulas haya en un vector de proyección más información contendrá la componente principal asociada y por lo tanto mayor será el porcentaje de varianza explicada.

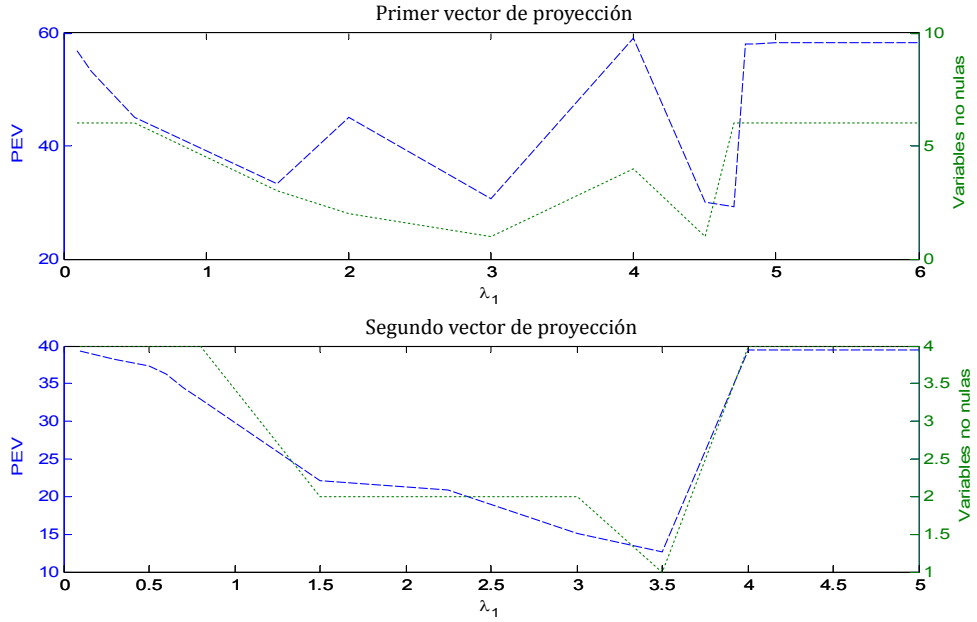

Figura 12: Evolución de la PEV y el número de componentes dispersas según aumenta λ_1

Una vez seleccionadas las lambdas para cada una de las dos primeras de las componentes principales, en este caso λ_1 = (4.759, 4.14), se procede a obtener los valores de las nuevas variables contenidas en ellas. Los resultados se detallan en la Tabla 7, donde también se muestran las tres primeras componentes principales halladas con el PCA original a modo de comparación.

	PCA			SPCA	
	PC1	PC2	PC3	PC1	PC2
X_1	0,133	-0,474	-0,086	0,0	0,5
X_2	0,133	-0,474	-0,086	0,0	0,5
X_3	0,133	-0,474	-0,087	0,0	0,5
X_4	0,133	-0,474	-0,087	0,0	0,5
X_5	-0,388	-0,159	0,272	0,5	0,0
X_6	-0,388	-0,159	0,271	0,5	0,0
X_7	-0,388	-0,159	0,272	0,5	0,0
X_8	-0,388	-0,159	0,273	0,5	0,0
X_9	-0,404	-0,008	-0,580	0,0	0,0
X_{10}	-0,404	-0,008	-0,581	0,0	0,0
Varianza Ajustada (%)	59.13	39.7	0.40725	40.35	39.33

Tabla 7: *Resultados de la simulación 4.1 cargas y varianza.*

En la Tabla 7 se puede observar como el SPCA obtiene la solución ideal dispersa que era de esperar para las dos primeras componentes al identificar correctamente los conjuntos de cargas asociadas el mismo factor de oculto. En SPC1 tenemos el factor V_2 y sus coeficientes asociados a X_5, X_6, X_7, X_8, mientras que en SPC2 tenemos X_1, X_2, X_3, X_4 generadas a partir de V_1.

La varianza explicada en las dos primeras PC es menor en el SPCA que en el PCA ya que el SPCA se centra en quedarse con las variables más importantes para cada componente en vez de quedarse con la mayor cantidad de información posible. De esta forma las cargas que no son importantes para la componente principal son nulas y la interpretación de los resultados es mucho más sencilla al saber exactamente que variables aportan información a cada PC.

Además de los parámetros λ_2 y λ_1 también hay que seleccionar otros parámetros para el correcto funcionamiento del algoritmo SPCA. Por una parte tenemos aquellos parámetros que determinan la condición de parada del algoritmo. Estos son el número máximo de iteraciones del algoritmo y la similitud entre **u** y **v** que indica la convergencia del algoritmo.

Estos dos parámetros están contrapuestos porque cuantas más iteraciones se realicen mejor será la similitud entre **u** y **v**, y más tiempo durará la ejecución. Por estos motivos hay que encontrar un compromiso entre ambos valores. La elección de este compromiso no esta sujeto a ninguna regla específica dependiendo del comportamiento del algoritmo SPCA para cada problema en particular y cambiando con cada conjunto de datos. Por ejemplo, la elección de un valor de 10^{-5} para la diferencia entre **u** y **v** y un máximo de 10^{5} iteraciones parece dar una precisión suficiente sin aumentar considerablemente el tiempo de ejecución para este ejemplo.

4.1.1 Inicialización del algoritmo

El otro parámetro que se puede modificar con el objetivo de mejorar el comportamiento del SPCA es la inicialización de **v**. La solución trivial es inicializar **v** con todas sus variables a un mismo valor trivial (0, 0'5 o 1). Otra forma de inicializar **v** es utilizar la solución del PCA original [30]. Hay que especificar que la inicialización a partir de la solución del PCA original que se utiliza es la **v** final que se obtiene cuando se hallan las PC y no las PC en si mismas.

Con la intención de dilucidar cuál de las inicializaciones propuestas provoca un mejor comportamiento del algoritmo SPCA se han comparado los tiempos de convergencia del algoritmo y el tiempo que tardaban en encontrar la misma solución. Son dos comprobaciones diferentes porque el tiempo de convergencia está determinado por la superación de la similitud entre **u** y **v** de un determinado umbral preestablecido. Mientras que el tiempo de ejecución depende de lo se tarde en encontrar una misma solución válida común para todas la inicializaciones sin depender de ningún límite de convergencia ni número máximo de iteraciones del algoritmo. En ambos experimentos se compararan los tiempos de las inicializaciones propuestas con el tiempo promediado de 100 inicializaciones aleatorias.

Se continua trabajando con el mismo problema artificial anterior por lo que se ha establecido como solución común para todas las inicializaciones la solución del apartado anterior para el algoritmo SPCA, incluida en la Tabla 7. Ahora también se va a estudiar la tercera componente principal para la cual el valor de los coeficientes del vector de proyección asociado es nulo para todas las cargas excepto para las dos últimas que tienen un valor de 0'707.

En primer lugar se estudiarán los tiempos de convergencia para cada una de las tres primeras componentes principales dispersas y el tiempo total empleado para cada inicialización.

Inicialización	A partir del PCA	Todo 0	Todo a 0.5	Todo a 1	Aleatorio
PC1	2,1959	1,5387	1,6451	1,5475	1,5068
PC2	1,5475	1,4595	1,5911	1,5435	1,4397
PC3	1,6323	1,5310	1,6119	1,6532	1,4629
Total	5,3933	4,5401	4,8606	4,7550	4,4222

Tabla 8: Tiempo de convergencia (en segundos) empleado por el SPCA para hallar cada una de las tres primeras componentes principales dependiendo de la inicialización de ***v****.*

En la Tabla 8 se puede observar como para la primera componente dispersa la inicialización fija que provoca un tiempo de ejecución menor es la inicialización con todas las variables a cero. Es lógico pues seis de las diez variables se hacen cero. La segunda inicialización con la que menos tiempo emplea la ejecución es con todas las componentes a 0'5 ya que las cuatro variables que no son nulas adquieren dicho valor. La inicialización usando el PCA no da buenos resultados debido a la gran diferencia entre los resultados del PCA y del SPCA. Las inicializaciones aleatorias ofrecen una amplia gama de resultados pero que en término medio su tiempo es inferior al resto de inicializaciones. Dependerá de la proximidad que exista entre la inicilización aleatoria y el valor de convergencia de **v**.

Los tiempos de convergencia para las componentes dispersas PC2 y PC3 son menores que al hallar la PC1 para todas las inicializaciones. El promedio del tiempo de convergencia al usar las inicializaciones aleatorias continúa siendo inferior al del resto de inicializaciones. Tanto en PC2 como PC3 el menor tiempo en las inicializaciones fijas es la inicializacion puesta por todo a 0, al igual que para PC1. Mientras que la inicialización utilizando la solución del PCA sigue sin mejorar al resto de inicializaciones fijas.

En la Tabla 8 también se muestran los tiempos totales de convergencia para cada una de las inicializaciones. El promedio del tiempo de las inicializaciones aleatorias ofrece el mejor resultado, seguido de cerca del tiempo de convergencia de la inicialización con todas las componentes a cero. Mientras, la ejecución cuando se ha utilizado el resultado del PCA original es la que emplea un mayor tiempo de convergencia bastante elevado.

A continuación se presentan las gráficas con los tiempos de ejecución resultantes para las distintas inicializaciones utilizadas en el experimento anterior cuando se busca un resultado idéntico. Como se explicó anteriormente el resultado que se pretende alcanzar es

el obtenido en el apartado anterior, y que se muestra en la Tabla 9, por lo que se utilizan los mismos parámetros (iteraciones mínimas y máximas, límite de convergencia, λ_1 y λ_2). La diferencia con el tiempo de convergencia estudiado antes es que ahora se ha modificado el algoritmo del SPCA de forma que no dependa de la convergencia entre **u** y **v** sino de alcanzar el resultado a este problema que ya se conoce. De esta forma se examinará el tiempo de ejecución del algoritmo para un resultado dado sin la influencia de parámetros externos.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
PC1	0	0	0	0	0,5	0,5	0,5	0,5	0	0
PC2	0,5	0,5	0,5	0,5	0	0	0	0	0	0
PC3	0	0	0	0	0	0	0	0	0,71	0,71

Tabla 9: *Solución común de las cargas de las tres primeras componentes principales del ejemplo 4.1 para todas las inicializaciones de* ***v****.*

En la Tabla 10 se muestra el tiempo de ejecución del algoritmo que tarda en hallar la solución común para cada una de las tres primeras componentes dispersas y el tiempo total conjunto de las tres.

Inicialización	A partir del PCA	Todo 0	Todo a 0.5	Todo a 1	Aleatorio
PC1	1,5378	1,3913	1,4631	1,4268	1,4584
PC2	1,3798	1,4327	1,4679	1,4891	1,4568
PC3	1,3950	1,5261	1,3835	1,4132	1,4187
Total	4,3265	4,3583	4,3255	4,3415	4,3434

Tabla 10: Tiempo empleado por el SPCA para hallar cada una de las tres primeras componentes principales dependiendo de la inicialización de ***v****.*

En general los tiempos de las inicializaciones son inferiores a los tiempos del caso anterior cuando se estudiaba el tiempo de convergencia. Para la primera componente la inicialización que ofrece un menor tiempo es en la que todas las variables de **v** son nulas. La inicialización a partir de la solución del PCA original continúa siendo la que emplea un mayor tiempo de ejecución para la primera componente aunque es la inicialización que más se reduce su tiempo comparado con los de convergencia.

Esta tendencia continúa para la segunda y la tercera componte, en las que la inicialización a partir del PCA original ofrece tiempos muy competitivos, incluso siendo el mejor tiempo para la PC2.

Las inicializaciones con todo a 1 y 0,5 también mejoran aunque en menor medida. Respecto a las inicializaciones aleatorias hay que destacar que para este caso no mejoran los

tiempos de las inicializaciones triviales como para el tiempo de convergencia. Ahora el tiempo promedio obtenido de las inicializaciones aleatorias no es el inferior en ninguna de las componentes principales.

Finalmente, respecto a los tiempos totales de ejecución vemos que todos están muy próximos entre si. La inicialización con todo **v** a 0,5 da un tiempo de 4,3255 siendo la mejor, pero la diferencia con la inicialización con todo a 0, que obtiene el peor tiempo total, es de 3.28 centésimas. Por lo que se puede concluir que para este caso la influencia de la inicialización no es relevante para la ejecución del algoritmo.

Cabe destacar que para este caso los tiempos de ejecución son menores que los tiempos de convergencia del caso anterior. Esto se explica porque ahora la condición de parada es menos estricta, al basarnos en los resultados requeridos no es necesario que la convergencia entre **v** y **u** sea muy restrictiva.

4.1.2 Tipo de algoritmo

Durante la ejecución de las pruebas en este ejemplo sintético siempre se ha utilizado la implementación del algoritmo SPCA deflactado. Esto es debido a que la solución que proporciona mantiene la ortogonalidad entre las componentes principales, mientras que la implementación del algoritmo SPCA en bloque no asegura que esto ocurra. Otra ventaja de usar el SPCA deflactado frente al SPCA en bloque es que facilita calcular el porcentaje de varianza proyectada, ya que al ser las componentes principales ortogonales se puede hacer directamente.

A continuación se presentan en la Tabla 9 los valores de los coeficientes asociados a las tres primeras componentes dispersas halladas tanto con el SPCA deflactado como con el SPCA en bloque. Las λ_1 han sido seleccionadas mediante el mismo proceso explicado para el SPCA deflactado al comienzo de esta sección. Para el SPCA deflactado se han utilizado λ_1 = {4'798, 4'1, 2'75} y para el SPCA en bloque han sido usadas λ_1 = {4'795, 4'5, 2'5}. Las λ_1 no son exactamente iguales en ambos implementaciones del SPCA aunque si muy parecidas. Esto se debe a que λ_1 afecta a la convergencia y al ser está abordada de forma diferente por cada método se requieren diferentes λ_1.

	SPCA Deflactado			SPCA en Bloque		
	PC1	PC2	PC3	PC1	PC2	PC3
X_1	0,0	0,5	0,0	0,0	0,5	-0,08
X_2	0,0	0,5	0,0	0,0	0,5	-0,08
X_3	0,0	0,5	0,0	0,0	0,5	-0,08
X_4	0,0	0,5	0,0	0,0	0,5	-0,08
X_5	0,5	0,0	0,0	-0,5	0,0	-0,36
X_6	0,5	0,0	0,0	-0,5	0,0	-0,36
X_7	0,5	0,0	0,0	-0,5	0,0	-0,36
X_8	0,5	0,0	0,0	-0,5	0,0	-0,36
X_9	0,0	0,0	0,707	0,0	0,0	0,48
X_{10}	0,0	0,0	0,707	0,0	0,0	0,48
Varianza Ajustada (%)	40.35	39.33	20.31	48,48	47,71	3,8

Tabla 11: *Comparación entre las PC del SPCA Deflactado y del SPCA en bloque*

En la Tabla 9 se puede observar como los vectores de proyección del SPCA deflactado son claramente ortogonales mientras que las componentes del SPCA en bloque no lo son en su totalidad. El tercer vector de proyección dispersa de la versión en bloque es el que rompe la ortogonalidad. Para este ejemplo no tendría excesiva relevancia ya que solo son importantes las dos primeras componentes dispersas al contener la información de los dos factores ocultos (V_1 y V_2). Sin embargo, para otros problemas donde se requieran la utilización de un mayor número de componentes principales dispersas si que sería de gran ayuda utilizar el SPCA deflactado en lugar del SPCA en bloque.

4.2 Simulación en problemas con más variables que observaciones

En este apartado se va a estudiar el funcionamiento del algoritmo SPCA cuando los datos poseen un número mucho mayor de variables que de observaciones. Para datos con este problema entra en juego el parámetro λ_2, al que se le asignará un valor positivo. De esta forma, se puede estudiar el conjunto completo de variables en vez de solo un número máximo igual al número de observaciones, gracias la uso de *la red elástica* explicada teóricamente en el apartado 3.2.2 Sin embargo, nos encontraremos con el inconveniente del aumento considerable del coste computacional al requerirse un número elevado de cargas

no nulas en los nuevos vectores de proyección con coeficientes dispersos.

En esta simulación con datos artificiales se ha planteado un escenario similar al del apartado 4.1. Pero a diferencia de éste se quiere comprobar si el algoritmo SPCA puede funcionar con un menor número de observaciones que de variables. Se buscará el correcto funcionamiento del SPCA al emplear un máximo de 50 observaciones y 500 variables.

Las variables son generadas a partir de cinco factores ocultos. En primer lugar se crean los cinco factores ocultos de la siguiente forma:

$$V_1 \sim N(0, 280), \qquad V_2 \sim N(0, 260), \qquad V_3 \sim N(0, 300)$$

$$V_5 = -0.3V_1 + 0.925V_2 + \varepsilon, \qquad V_4 = -0.3V_1 + 0.925V_2 + \varepsilon, \qquad \varepsilon \sim N(0, 300)$$

V_1, V_2, V_3 y ε son independientes.

A continuación se generan 500 variables

$$X_i = V_1 + \varepsilon^1_i, \qquad \varepsilon_i^1 \sim N(0, 1), \quad i = 1, ...,100$$

$$X_i = V_2 + \varepsilon^2_i, \qquad \varepsilon_i^2 \sim N(0, 1), \quad i = 101, ..., 200$$

$$X_i = V_3 + \varepsilon^3_i, \qquad \varepsilon_i^3 \sim N(0, 1), \quad i = 201, ..., 300$$

$$X_i = V_4 + \varepsilon^4_i, \qquad \varepsilon_i^3 \sim N(0, 1), \quad i = 301, ..., 400$$

$$X_i = V_5 + \varepsilon^5_i, \qquad \varepsilon_i^3 \sim N(0, 1), \quad i = 401, ..., 500$$

$\{\varepsilon^j_i\}$ son independientes, j = 1,2, ..., 5, i = 1, 2, ..., 500.

Cada factor tendrá asociadas la misma cantidad de variables en cada observación, en este caso son 100 variables. Mientras que la varianza de cada uno de los factores es respectivamente 290, 270, 320, 180 y 170.

El objetivo de este experimento es comprobar el funcionamiento del uso de λ_2 en el SPCA cuando el número de observaciones es bastante menor que el número de variables a estudiar. El análisis estará centrado en como evolucionan el número de cargas no nulas según varían λ_2 y λ_1 porque son estos coeficientes los que determinan la información en cada componente. Para ello se van a estudiar las dos primeras componentes dispersas.

El factor oculto V_3 tiene la mayor varianza por lo que es el más importante y las cargas asociadas a él están distribuidas entre X_{201} y X_{300}. Por este motivo en el primer vector de proyección disperso deberían ser estas variables las únicas no nulas para el valor adecuado de λ_2 y λ_1. Un caso similar ocurrirá en la segunda componente, en la que las únicas cargas no nulas en el vector de proyección asociado para los valores correctos de λ_2 y λ_1 deben ser los coeficientes asociados al segundo factor oculto con mayor varianza, V_1. Las cargas asociadas al factor V_1 está distribuidas entre X_1 y X_{100}.

Solamente se verificará el comportamiento de las dos primeras componentes dispersas porque en la tercera componente el estudio sería el mismo, por lo tanto repetitivo y no aportaría nueva información. El resto de componentes no se analizarán debido a que ya no contendrán información importante, al estar está contenida en las tres primeras componentes asociadas con los tres factores ocultos independientes.

Se va a comenzar el análisis probando como evolucionan las dos primeras componentes dispersas cuando λ_2 es nula. Es decir, se usará únicamente la regularización L_1 *lasso* por lo que existirá un límite de variables que utilizará el algoritmo. Dicho límite está determinado por la cantidad de observaciones que tengan los datos usados cuando su número sean menor que el número de las variables.

En primer lugar se ejecutará el algoritmo para 1000, 500, 250, 100 y 50 observaciones y λ_2 es nula.

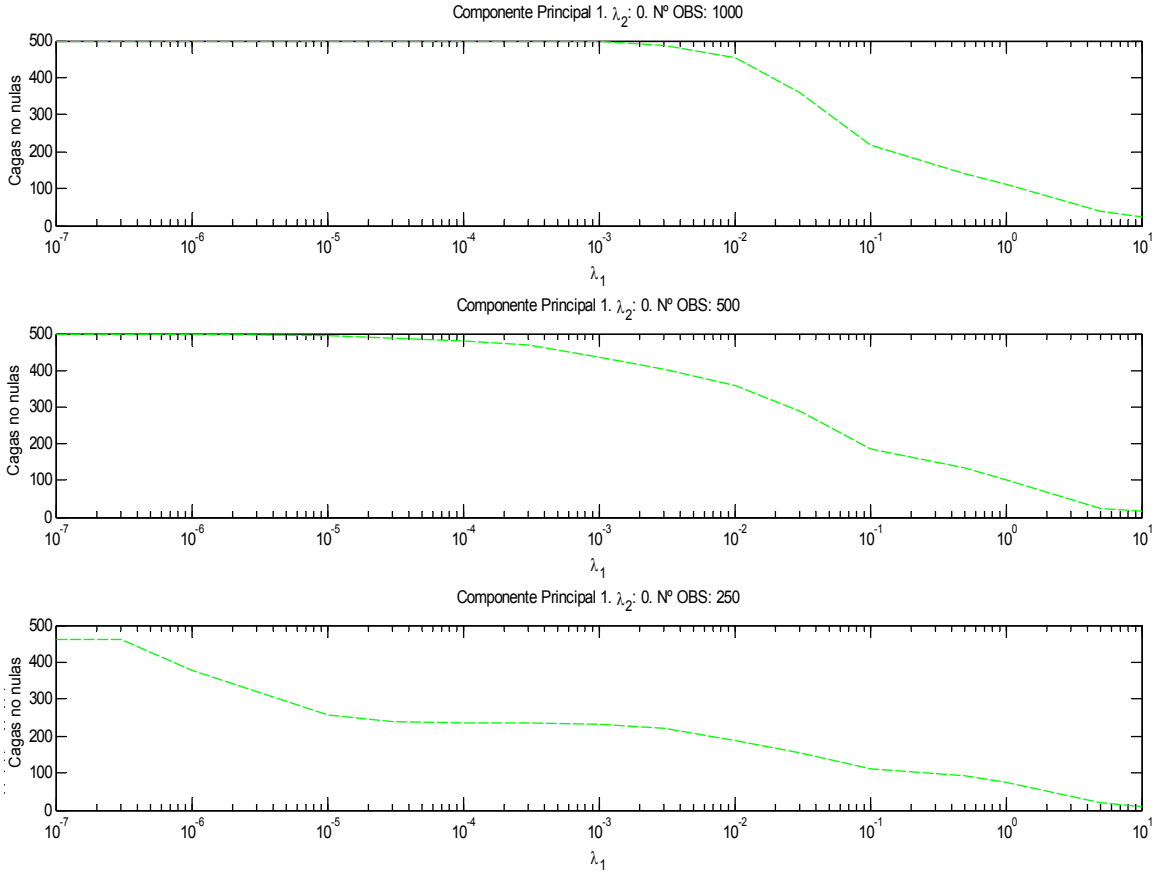

Figura 13: Evolución del número de cargas no nulas en la SPC1 para 1000, 500 y 250 observaciones.

En la Figura 21 se muestra la evolución del número de cargas no nulas dependiendo de λ_1 para la primera componente principal dispersa utilizando 1000, 500 y 250 observaciones. Se puede ver como en los dos primeros casos, en los que hay más observaciones que variables, se utilizan todas las variables para valores pequeños de λ_1 y a medida que esté

valor aumenta se reducen suavemente el número de variables no nulas.

En la Figura 22 se puede comprobar como para el caso de 500 observaciones si estudiamos los valores de todas las cargas de la primera componente principal dispersa para el valor adecuado de λ_1 en el que se obtienen 100 coeficientes no nulos. Estas cargas no nulas coinciden con los coeficientes asociados al factor oculto con mayor varianza, en este caso V_3.

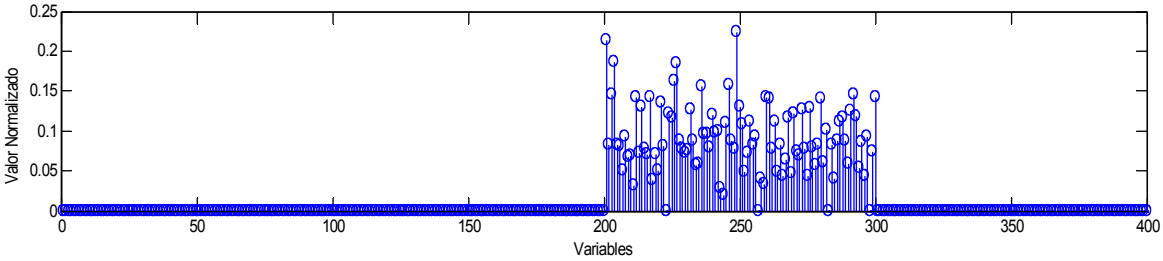

Figura 14: Valor normalizado de todas las cargas de la SPC1 para λ1 igual a 0.45 y 500 observaciones.

Sin embargo, cuando se usan solamente 250 observaciones, un número menor que el de variables en cada observación, las cargas utilizadas por el algoritmo se reducen bruscamente al aumentar λ_1. Esto ocurre en torno al valor de λ_1 de 10^{-5}, donde el algoritmo emplea 250 cargas. Este número de cargas no nulas se mantiene hasta el valor de λ_1 de 10^{-2} cuando prosigue la reducción de cargas no nulas al igual que ocurre en los casos de 1000 y 500 observaciones.

Por lo tanto, se puede establecer que existe un límite en el que se utilizan como máximo tantas variables como observaciones contengan los datos de la simulación. El rango de λ_1 en el que se establece este límite está determinado en la parte inferior por el valor de λ_1 para él que el algoritmo funciona como si λ_1 fuera cero, es decir, no tiene influencia en el funcionamiento del algoritmo. Este valor de λ_1 se encuentra entorno a 10^{-5}. Mientras que el valor superior del rango del límite está determinado por el valor para el cual el algoritmo trabaja con menos variables no nulas que el número de observaciones. En este caso a partir de λ_1 equivalente a 10^{-2} se obtendrían menos de 250 cargas no nulas.

Se puede determinar que existe un límite que establece el número máximo de variables a utilizar por el algoritmo y que está definido por el número de observaciones, como era de esperar, para casos en los que se trabaja con un menor número de observaciones que de variables.

Este mismo comportamiento ocurre si seguimos reduciendo el número de observaciones del conjunto de datos que empleamos en el SPCA.

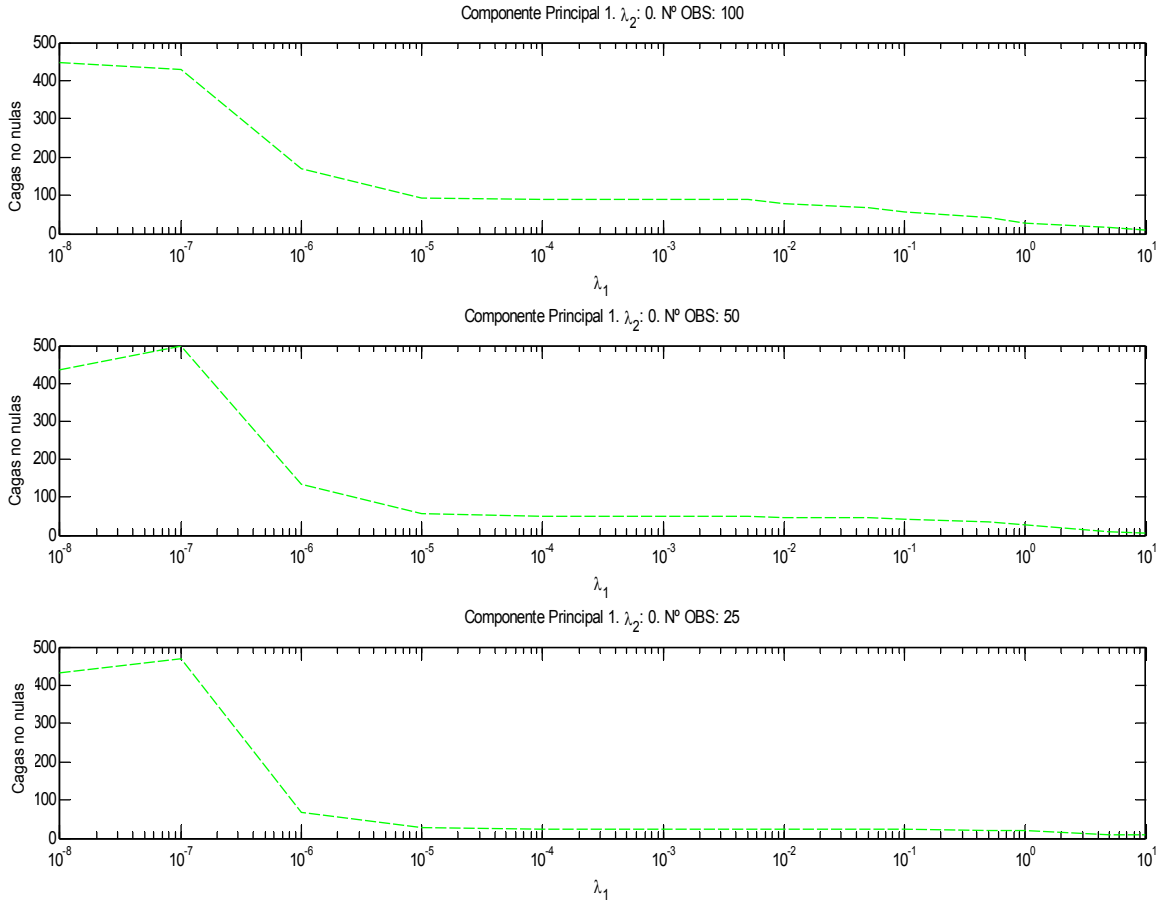

Figura 15: Evolución del número de cargas no nulas en la SPC1 para 100, 50 y 25 observaciones.

Se encuentra un comportamiento similar al del caso de 250 observaciones en los casos de 100, 50 y 25 observaciones. En la Figura 23 se han dibujado las evoluciones de estos casos en los que también existe un límite que determina el número máximo de variables no nulas que utiliza el algoritmo SPCA. También se puede comprobar que este límite coincide con el número de observaciones en cada caso. Además, al igual que en el caso de 250 observaciones el límite comienza en torno a λ_1 igual a 10^{-5} y terminara en el punto a partir del cual el algoritmo use menos variables que observaciones. Este punto será diferente para cada caso, aumentando cuantas menos observaciones estén siendo utilizadas.

En las siguientes gráficas se va a estudiar el comportamiento de la segunda componente principal dispersa. Esta componente está relacionada con el factor oculto V_1, el segundo factor con mayor varianza.

En las Figuras 24, 25 y 26 se muestran la evolución del número de cargas no nulas según aumenta el parámetro λ_1 para 1000, 500, 250, 100, 50 y 25 observaciones. Al igual que ocurría para el caso de la primera componente dispersa para los casos con mayor número de observaciones que de variables no se aprecia ningún límite y el número de

cargas no nulas desciende suavemente a medida que aumenta λ_1. Mientras que para los casos en los que se trabaja con un menor número de observaciones que de variables existe un límite de cargas no nulas equivalente al número de observaciones del caso.

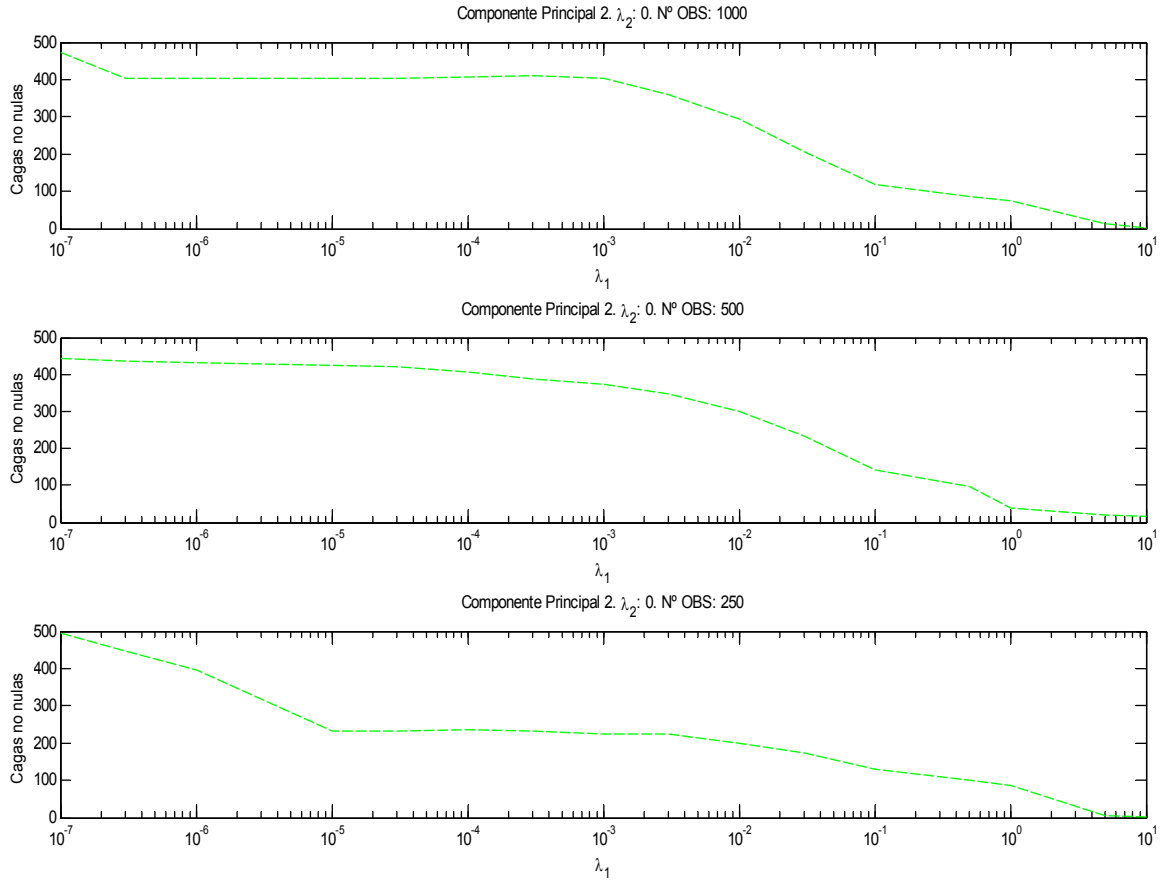

Figura 16: Evolución del número de cargas no nulas en la SPC2 para 1000, 500 y 250 observaciones.

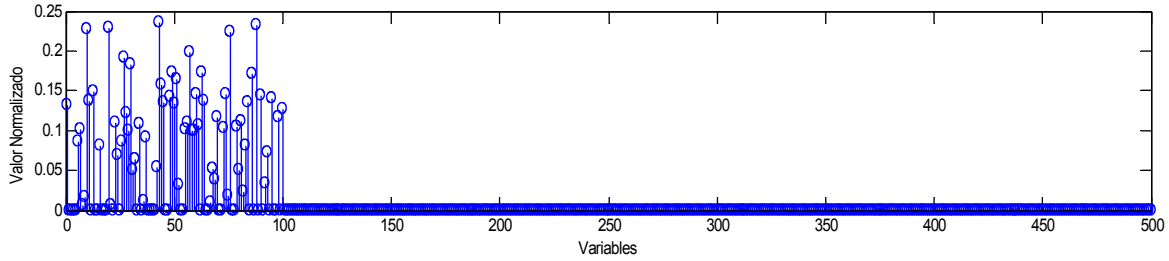

Figura 17: Valor normalizado de todas las variables de la SPC2 para λ_1 igual a 0.45 y 500 observaciones.

En la Figura 25 se muestra el valor normalizado de las variables de la segunda componente dispersa para el caso de 500 observaciones, habiendo se seleccionado λ_1 con valor 0.45 para que solo haya entorno a 100 cargas distintas de cero. Como se puede comprobar en la Figura 25 estás variables no nulas están entre las 100 primeras, es decir aquellas generadas a partir del factor oculto V_1. De esta forma queda demostrado que si se selecciona adecuadamente λ_1 la segunda componente dispersa se queda únicamente con las variables asociadas al segundo factor con mayor importancia, en este caso V_1.

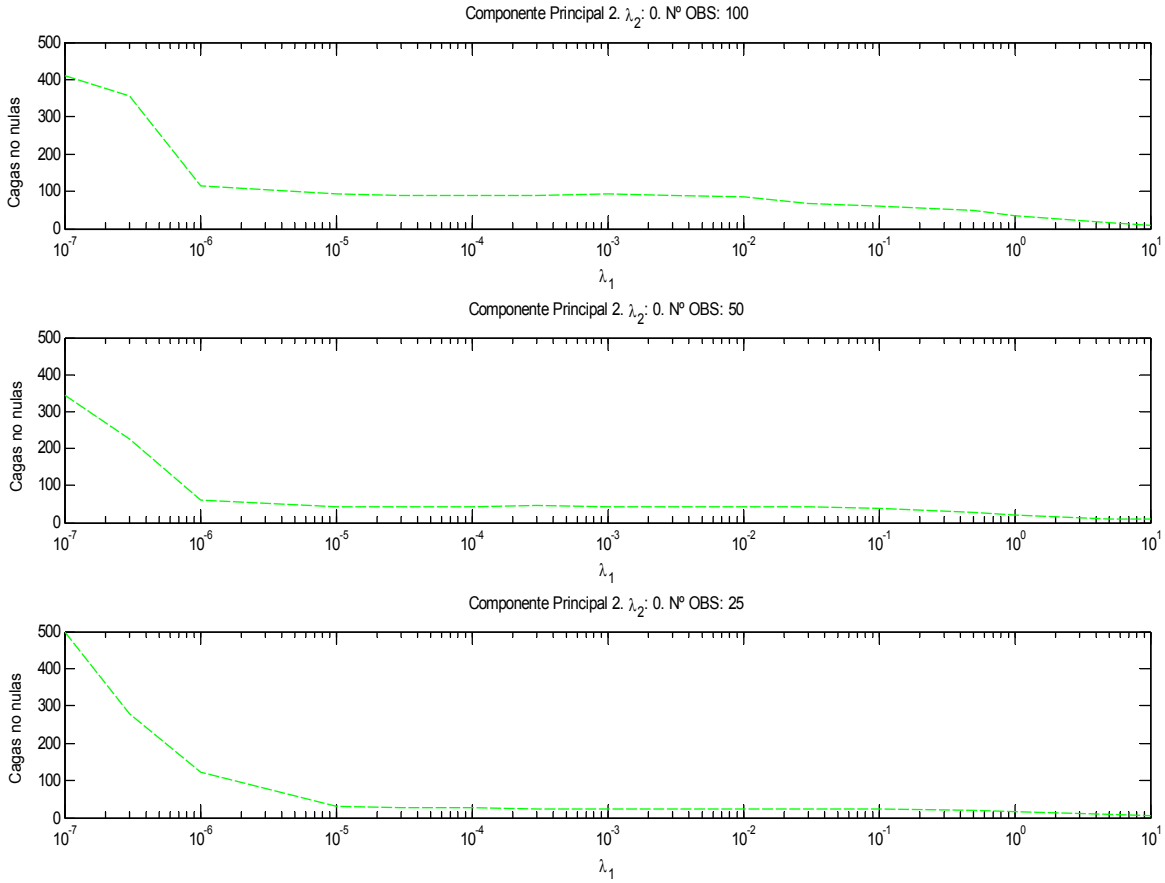

Figura 18: Evolución del número de cargas no nulas en la SPC2 para 100, 50 y 25 observaciones.

A continuación se repetirá el experimento para λ_2 mayor que 0. Se busca comprobar si con un número menor de observaciones que de variables, al introducir λ_2 el algoritmo SPCA puede funcionar sin limitaciones. Se estudiará el comportamiento del número de variables no nulas para diferentes λ_1, λ_2 y 50 observaciones. Se ha seleccionado 50 observaciones al ser un número diez veces inferior al número de variables generadas para este experimento, suficientemente pequeño para estudiar la influencia del parámetro λ_2.

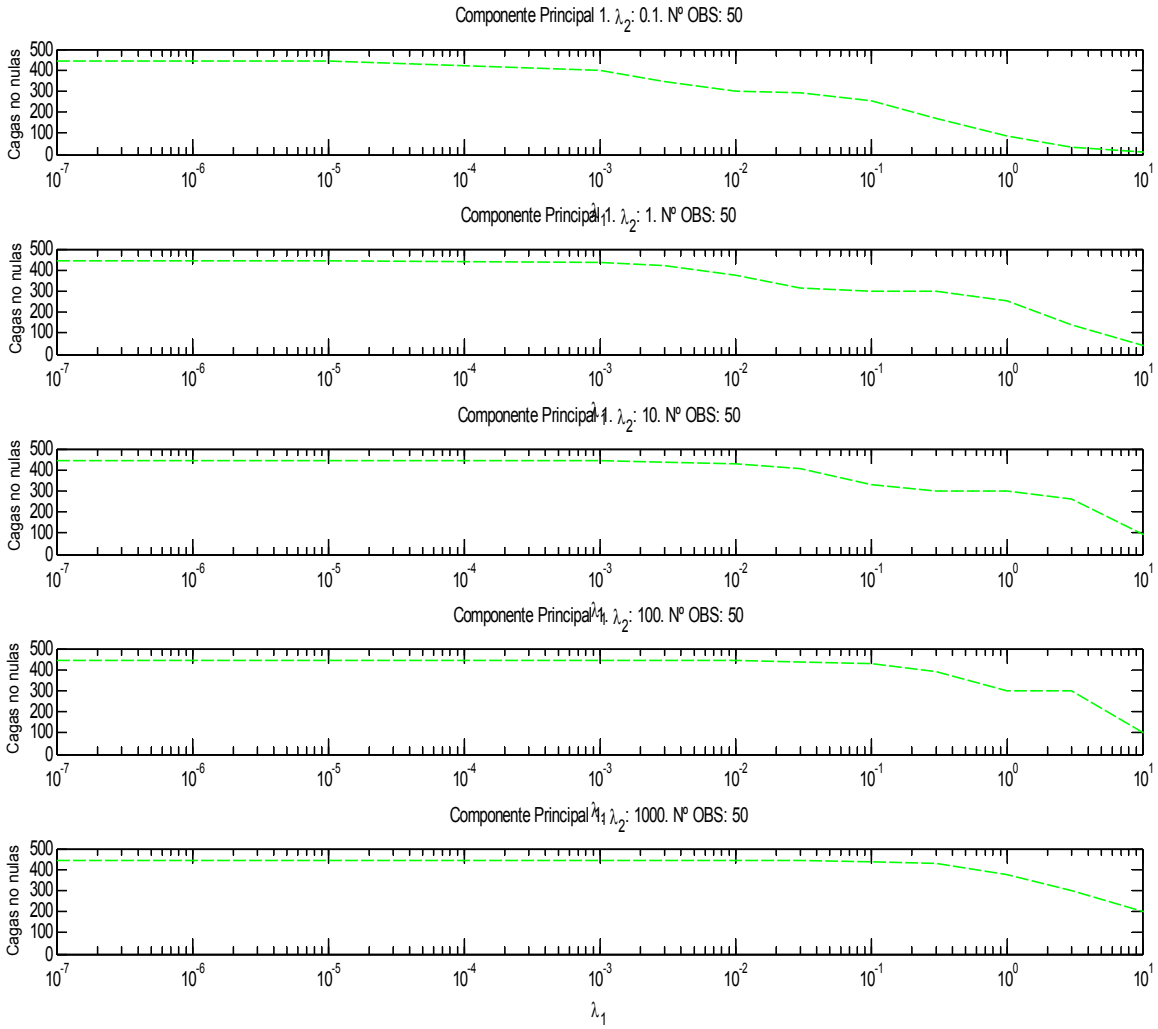

Figura 19: Evolución del número de cargas no nulas de SPC1 para diferentes λ_1 y λ_2 utilizando 50 observaciones.

En la Figura 27 se muestra la evolución del número de cargas no nulas cuando empleamos solo 50 observaciones. Para todos los valores de λ_2 utilizados podemos ver como el algoritmo es capaz de seleccionar todas las variables sin existir ningún límite que se lo impida como ocurre en el caso de λ_2 nula.

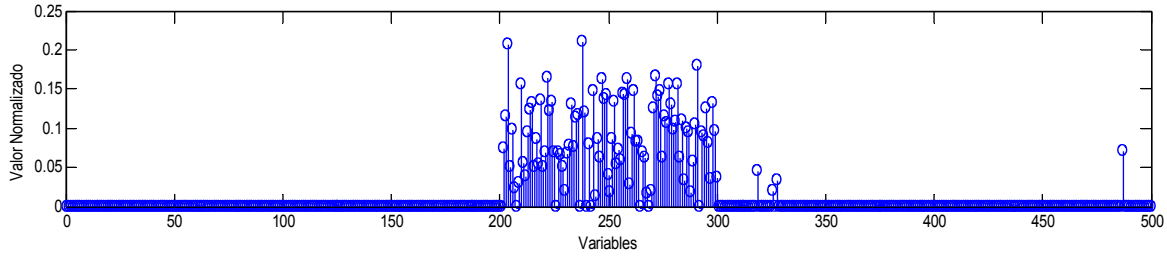

Figura 20: Valor Normalizado de las variables de la SPC1 para λ1 igual a 0.06, λ2 igual a 0.1 y 50 observaciones.

Para terminar de comprobar el funcionamiento cuando hay menos observaciones que variables y utilizamos la *red elástica* al emplear λ_2 se han representado en la Figura 28 los

valores de las cargas cuando tenemos en torno a 100 cargas no nulas.

Se puede observar el 90% de las cargas no nulas pertenecen a la franja entre 201 y 300, lo que significa que están asociadas al factor oculto V_3. Los errores que se ven están determinados por el ruido o la cercanía en la varianza de los factores ocultos. Hay que tener en cuenta que para este caso utilizamos muy pocas observaciones por lo que tenemos muy pocos datos por variable provocando que sea más fácil confundir los factores y no seleccionar el correcto.

Como para λ_2 nula, también se ha comprobado el funcionamiento para la segunda componente dispersa y de esta forma asegurarnos que los resultados obtenidos para la primera componente no son casuales.

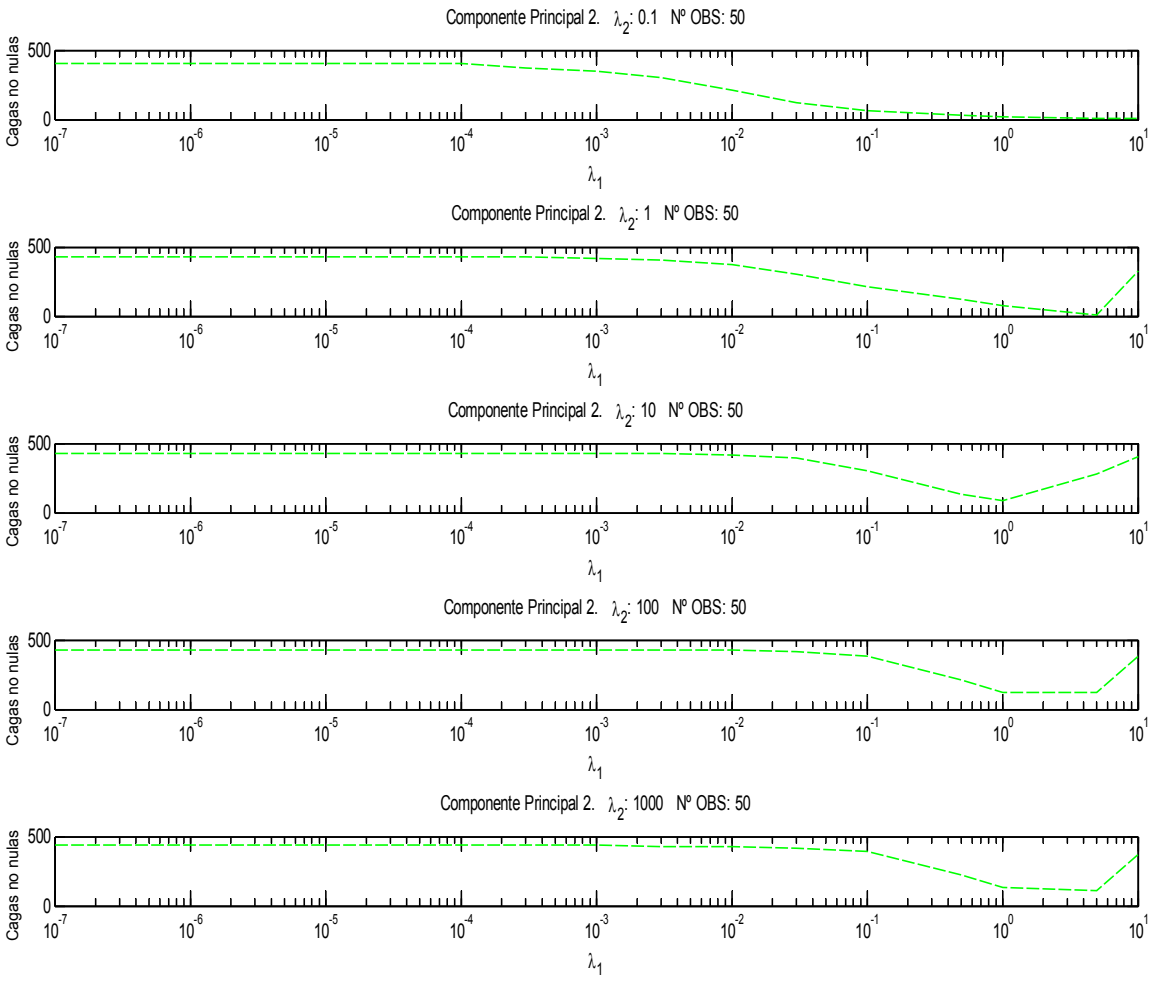

Figura 21: Evolución del número de cargas no nulas de SPC2 para diferentes λ1 y λ2 utilizando 50 observaciones.

En la Figura 29 están dibujadas las curvas con la evolución de las cargas no nulas según aumenta λ_1 para λ_2 igual a 0.1, 1, 10, 100 y 1000. Se puede verificar que el comportamiento

de estas curvas es similar al que siguen las curvas de SPC1 representadas en la Figura 27. La diferencia más notable es que, excepto para λ_2 igual a 0.1, se llega al mínimo de cargas no nulas en el rango empleado de λ_1. A partir de este mínimo la cantidad de variables no nulas aumenta rápidamente hasta volver al máximo de cargas no nulas posibles, es decir, todas.

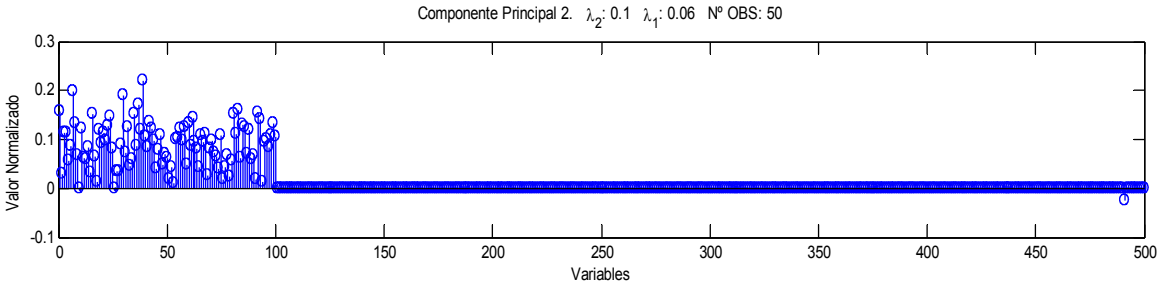

Figura 22: Valor normalizado para las variables de SPC2 para λ_2 igual a 0.1 y λ_1 igual a 0.06

En la Figura 30 se presentá el valor normalizado para las variables de la segunda componente dispersa. Se ha seleccionado el punto aceptable para λ_2 igual a 0.1 y λ_1 igual 0.06 que proporcione alrededor de 100 cargas no nulas. Se puede observar como prácticamente todas las cargas no nulas se encuentran entre 1 y 100, lo que quiere decir que están asociadas al factor oculto V_1. Por lo tanto, el algoritmo funciona correctamente al seleccionar las variables asociadas al segundo factor con mayor variable para la segunda componente principal dispersa.

Los errores que se pueden encontrar son causados por la proximidad en los varianza de los factores así como al ruido añadido a las muestras. También se pueden encontrar valores nulos entre en el rango 0 y 100, esto es debido a que no se ha obtenido exactamente 100 valores no nulos ya que hallar los puntos exactos es bastante laborioso y tedioso. Además con los valores representados en la Figura 30 se muestra claramente que el algoritmo selecciona las cargas asociadas a las variables del factor V_1.

4.3 Simulación con bases de datos reales

Hasta ahora se han presentado dos ejemplos de simulaciones con datos artificiales creados específicamente para comprobar el funcionamiento del SPCA de distintos puntos de vista. En esta última simulación de este capítulo se va a comprobar el comportamiento del SPCA al ser empleado sobre cuatro bases de datos reales.

Para analizar las ventajas, o inconvenientes, de su aplicación en un entorno real, se incluirá en el estudio una etapa posterior de clasificación y se analizará el error final del clasificador cuando se usan como entrada las proyecciones proporcionadas por el SPCA. Para conocer la calidad del SPCA se compararán sus resultados con los PCA original. También se contemplará el caso del clasificador sin ninguna etapa de preprocesado.

Para comparar los distintos métodos se ha utilizado un clasificador lineal basado en máquinas de vectores de soporte. Se usará específicamente el clasificador SVM definido en la librería LIBSVM creada por Chih-Chung Chang and Chih-Jen Lin [56].

Las bases de datos utilizadas pertenecen al repositorio UCI y son: Optical recognition of handwritten digits (hand), Dermatology (ion), Statlog – Landsat Satellite (landstat) y Waveform Database Generator (wave). Todas estas bases de datos contienen datos reales que puede ser clasificados en dos grupos.

Con el objetivo de seleccionar los valores de los parámetros más adecuados tanto del algoritmo SPCA ($\lambda 1$) como del clasificador (C) se ha empleado una validación cruzada de 5 iteraciones. El parámetro C del clasificador SVM pondera la suavidad de la solución con el número de errores permitidos

A continuación se muestran las gráficas de la evolución del error de clasificación dependiendo del número de componentes principales utilizadas. Para todos los conjuntos de datos se han estudiado las primeras diez componentes principales, al contener estas más del 99% de la varianza de los datos de entrada.

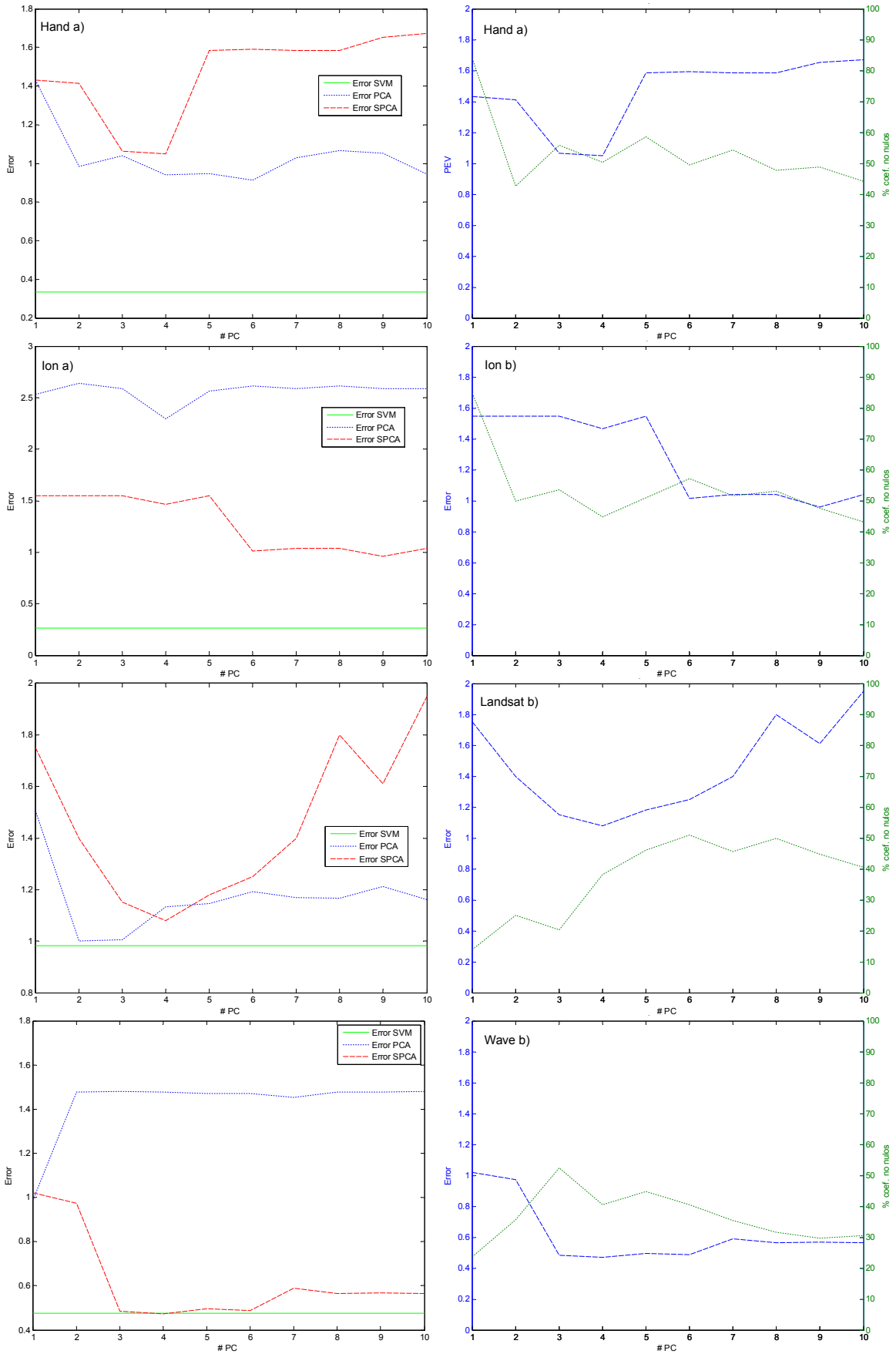

Figura 23: Evolución de clasificación según el número de componentes principales utilizadas para el SPCA, PCA y usando únicamente el clasificador SVM, gráficas a). Error de clasificación y porcentaje de cargas no nulas en las componentes principales según el número de componentes principales del SPCA, gráficas b).

La primera conclusión que se puede extraer de la Figura 31, es que en las gráficas a) el menor error cuadrático medio siempre se obtiene al aplicar directamente en el clasificador SVM. Esto es lógico pues tanto el PCA como el SPCA son técnicas no supervisadas que no tienen en cuenta las etapas posteriores del sistema, en este caso el clasificador SVM. Además como uno de sus objetivos es trabajar con un espacio de datos más reducido, la información disponible en los datos que entran en el clasificador es menor para los dos casos con preprocesado de los datos.

Otra conclusión que se obtiene de las gráficas a) es que un aumento de las componentes principales utilizadas en el PCA y el SPCA no implica una mejora del error en el clasificador. Por lo general, las curvas de error alcanzan su mínimo para el uso de 3 a 5 componentes principales, estando este error muy cercano al error cometido por el clasificador directo. Hay que tener en cuenta que tanto la forma de la curva como el mínimo error determinado según el número de componentes varían con cada conjunto de datos.

Entre la aplicación del algoritmo PCA y del SPCA se encuentran pocas diferencias resaltables más allá de que dependiendo del conjunto de datos funcionará mejor uno que otro. La principal es que el PCA necesita menos componentes principales que el SPCA para alcanzar su mínimo error. Esto es debido a que al tener todas las cargas distintas de cero, las primeras componentes (sobre todo la primera) acumularán más información en el PCA que en el SPCA. Sin embargo, el SPCA utilizando, generalmente, una componente más alcanza su mínimo de error y al usar cargas nulas se mejora considerablemente la interpretación de los resultados. De esta forma, se puede conocer que variables originales del espacio de entrada aportan información a cada componente principal.

En las gráficas b), que muestran el error al utilizar el SPCA y el porcentaje de cargas no nulas en las componentes principales, se puede extraer de su análisis que el menor error se suele encontrar cuando hay más componentes no nulas. Este resultado es razonable ya que cuantos más coeficientes sean distintos de cero más información habrá en el nuevo espacio. Sin embargo, como el objetivo del SPCA es anular las cargas para poder interpretar mejor la información que aportan al nuevo espacio las variables originales, cuando exista una etapa posterior de clasificación habrá que encontrar un compromiso entre el error requerido y el número de cargas nulas.

Por lo tanto, tras observar estos resultados se puede determinar que ni el PCA ni el SPCA logran una mejora cuantitativa en el error cuadrático medio respecto de la clasificación directa de los datos sin ninguna etapa de preprocesado. Sin embargo, si se

utilizan un número específico de componentes principales en la etapa de preprocesado para cada caso, se obtiene un error muy cercano al del clasificador directo usando un espacio de datos muy reducido comparado con el espacio inicial de datos.

Es evidente que para conjuntos de datos con pocas variables introducir una etapa de preprocesado de este tipo no es eficiente, pero en la actualidad las bases de datos suelen contar con un número muy elevado de dimensiones lo que hace que la introducción de etapas como el PCA o SPCA no solo son útiles sino necesarias. Gracias al empleo de menos dimensiones de entrada, estas etapas de preprocesado facilitan el entrenamiento del clasificador al reducir su coste de entrenamiento. Además si en el ámbito de la base de datos es necesario interpretar las variables en el resto del sistema, con el uso del SPCA se obtiene una reducción considerable de las dimensiones mientras se conoce que variables originales intervienen en cada componente principal.

Capítulo 5

Experimentos Ɛ-PCA

En los siguientes apartados se detallaran cuatro simulaciones diferentes en las que se comprobará el funcionamiento del Ɛ-PCA. Se prestará especial atención a su comportamiento frente al ruido impulsivo.

El primer experimento es muy útil para comprender el funcionamiento de este algoritmo. Únicamente se emplearán dos dimensiones, dos variables, por lo que los resultados obtenidos podrán ser representados gráficamente con facilidad. En el segundo experimento se utilizarán tres dimensiones y en el tercero se aumentará el número de variables hasta 10. Finalmente, como en el capítulo anterior, se estudiara el comportamiento del Ɛ-PCA al usar bases de datos reales.

A la hora de implementar el Ɛ-PCA en MATLAB se ha utilizado el software específico CVX para la parte que resuelve la expresión (3.19). El CVX es un sistema de optimización convexa basado en MATLAB [57] y [58].

5.1 Simulación en un ejemplo sintético bidimensional

Este primer experimento tiene como objetivo comparar de la forma más sencilla posible el funcionamiento del algoritmo PCA y el Ɛ-PCA, cuando una de las variables de entrada está dañada por ruido impulsivo. Para esta simulación se van a emplear datos artificiales en los que cada observación solo estará compuesta por dos variables. De esta forma los datos y las componentes se podrán representar gráficamente con facilidad permitiendo un análisis más claro que si solo se utilizarán los resultados numéricos.

La simulación que se va a generar para este experimento consta de un factor oculto que estará escalado por dos valores para dar cada una de las variables. El modelo que se utilizará en esta simulación tendrá 200 observaciones y se presenta a continuación:

$$V \sim N(0, 500^2) + \varepsilon, \qquad \varepsilon \sim N(0, 1)$$
$$X_1 = 0.2V + \varepsilon_1, \qquad X_2 = 0.8V + \varepsilon_2, \qquad \varepsilon_1, \varepsilon_2 \sim N(0, 50^2)$$

Este experimento constará de dos partes y en ambas el ruido impulsivo será añadido a la variable X_2. En primer lugar se van a comparar los métodos PCA y Ɛ-PCA para el caso en el que no hay ruido en los datos y para cuando si se hay ruido en una variable. De está forma veremos como funciona el Ɛ-PCA de una manera superficial sin prestar atención a sus parámetros, para lo que se usarán valores fijos.

Sin embargo, en la segunda parte de este experimento se estudiará el comportamiento del Ɛ-PCA cuando se varían los valores de sus parámetros para diferentes escenarios de ruido. Estos resultados se confrontarán con las soluciones derivadas del PCA. Finalmente, se analizará cual de los métodos presenta mejores resultados, para cada situación.

En este experimento se presentaran los resultados de dos formas. Por una parte, se mostrarán gráficamente las direcciones de proyección dadas por las componentes principales correspondientes a cada algoritmo. Por otra parte, se mostrarán los resultados numéricos que establecerán la desviación de cada componente sobre la dirección óptima. Esta dirección optima estará definida como la dirección de máxima varianza para los datos sin ruido. Por lo tanto esta dirección vendrá determinada por las componentes principales del PCA cuando se utilicen los datos sin ruido. La desviación entre la dirección óptima y la dirección que presenten las componentes del PCA y del Ɛ-PCA para los datos con ruido será determinada, como en los capítulos anteriores, por la diferencia coseno.

A continuación se muestra la Figura 24 donde se han representados los datos sin ruido junto con las dos componentes principales de cada método. Los valores de los parámetros seleccionados para llevar a cabo el Ɛ-PCA son C igual a 1 y ε a 1. Como se verá en la segunda parte de este experimento estos parámetros no son los óptimos pero ofrecen un resultado suficientemente bueno para mostrar el funcionamiento del Ɛ-PCA.

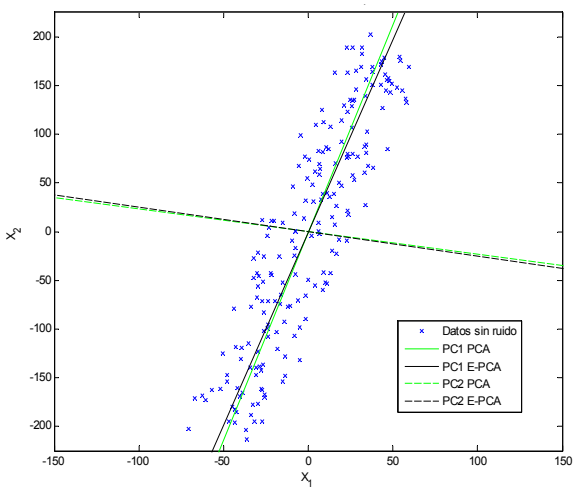

Figura 24: Representación de los datos sin ruido así como de las PC halladas mediante PCA y Ɛ-PCA.

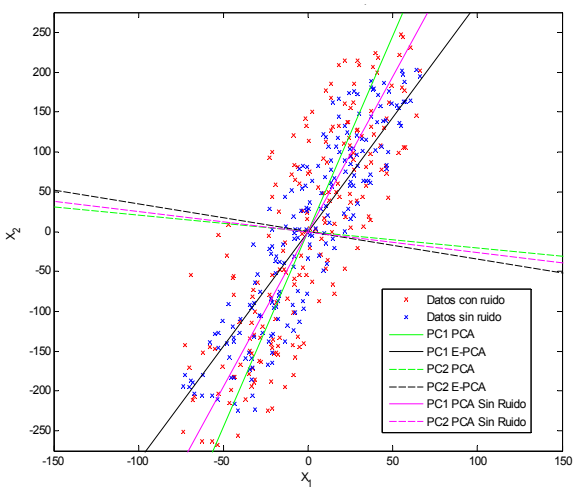

Figura 25: Representación de los datos con ruido y sin ruido. Además de las dos primeras componentes principales de PCA y E-PCA para el caso con ruido.

Se puede observar en la Figura 24 como las componentes principales del PCA tienen las direcciones de máxima varianza para cada una de las dos variables. Se comprueba como la primera componente principal sigue la dirección de la máxima varianza, que en esta simulación es próxima a la dirección de X_2. Por lo que la segunda componente, ortogonal a la anterior, es próxima a la dirección de la variable X_1.

La conclusión que se extrae de esta primera parte del experimento es que en un escenario sin ruido, el comportamiento del Ɛ-PCA es prácticamente similar al del PCA. Aunque como es de esperar, las diferencias entre ambos algoritmos vienen dadas por el propio modelo del algoritmo Ɛ-PCA y por la selección de sus parámetros.

En la Tabla 12 se muestran los resultados numéricos de esta simulación y se puede comprobar la desviación que sufren las componentes del Ɛ-PCA respecto a las del PCA. También se ha calculado la varianza de cada componente. Vuelve a quedar demostrado, que en el caso sin ruido, el Ɛ-PCA tiene un comportamiento similar al del PCA, ya que proyectan prácticamente la misma varianza.

	PCA		E-PCA	
	PC1	PC2	PC1	PC2
X_1	-0.2281	0.9736	0.2447	-0.9695
X_2	-0.9736	-0.2281	0.9695	0.2447
Varianza (%)	98.6568	1.3431	98.6285	1.3714
Diferencia Coseno	1	1	-0.9998	0.9998

Tabla 12: *Resultados del PCA y del Ɛ-PCA para datos sin ruido. Se muestran las cargas y la varianza de cada componente, así como la desviación respecto de la dirección de máxima varianza.*

En la Figura 25 están las componentes principales cuando se introduce ruido impulsivo sobre la segunda variable. Este ruido tiene la misma potencia que la variable X_2. Además se muestran también las componentes PCA cuando no hay ruido para no perder la referencia de cual es el objetivo. Este objetivo no será otro que las componentes E-PCA cuando halla ruido se desvíen muy poco de las direcciones de máxima variación sin ruido.

En la Figura 25 y en la Tabla 13 se observa como tanto las componentes del PCA como las del Ɛ-PCA se han desviado al introducir ruido. Ambos algoritmos se desvían de la solución del PCA sin ruido, aunque en direcciones opuestas. Este hecho muestra que al introducir ruido impulsivo, como era de esperar, cada algoritmo proporciona una solución diferente, ya que minimizan costes distintos.

	PCA		E-PCA	
	PC1	PC2	PC1	PC2
X_1	-0.1997	-0.9798	0.3285	0.9444
X_2	-0.9798	0.1997	0.9444	-0.3285
Varianza(%)	98.108	1.8919	96.3985	3.6014
Diferencia Coseno	0.9987	0.9987	-0.9965	-0.9965

Tabla 13: *Resultados del PCA y del Ɛ-PCA para datos con ruido impulsivo. Se muestran las cargas y la varianza de cada componente, así como la desviación respecto de la dirección de máxima varianza.*

En el segundo experimento de esta sección, veremos si ajustando correctamente lo parámetros del Ɛ-PCA podemos conseguir que su solución esté alineada con la óptima.

En primer lugar, se van a representar las componentes principales del PCA y del Ɛ-PCA cuando han utilizado datos con ruido impulsivo. El ruido impulsivo tendrá una potencia que variará entre la misma, el doble y el cuádruple que la poseen los datos de X_2, al ser esta la variable sobre la que volveremos a añadir el ruido impulsivo. El comportamiento del Ɛ-PCA se estudiará para ε igual a 1, 0.001 y 1e-7, mientras que el valor de C oscilará entre 1, 10 y 100. Estos valores han sido elegidos porque muestran resultados suficientemente claros y

específicos. También se mostrarán las componentes principales del PCA sin ruido impulsivo para poder ver el resultado óptimo que se pretende alcanzar.

En primer lugar se empleará ruido impulsivo con la misma potencia que X_2. Las gráficas resultantes para los valores del coste (C) y ε se encuentran en la Figura 26.

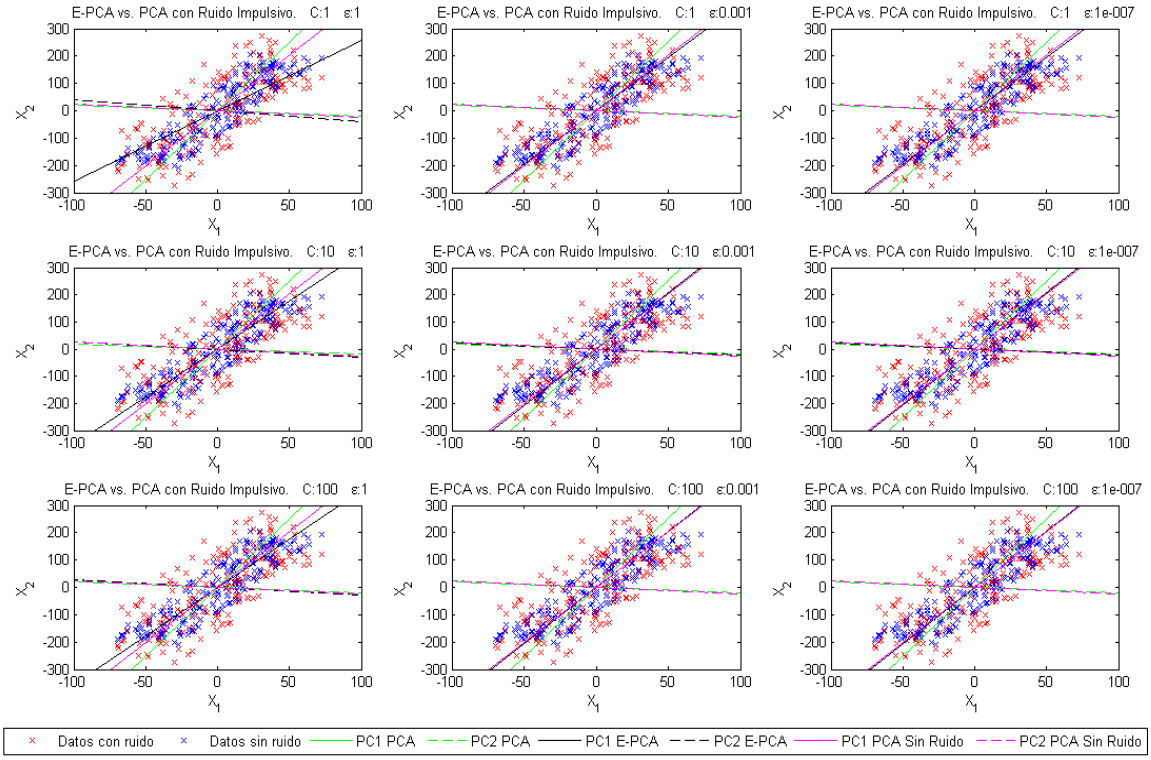

Figura 26: Gráficas de las componentes principales de PCA y Ɛ-PCA para datos con ruido impulsivo con potencia igual a la de X2.

En la Tabla 14 se muestran los resultados numéricos de la desviación. Se puede comprobar, tanto en la Figura 26 como en la Tabla 14, que para este caso el parámetro más influyente es ε porque para valores inferiores a 0'001 el Ɛ-PCA encuentra componentes que tienen dirección óptima. Para un valor de ε igual a 1 los resultados no son los óptimos. Esto es debido a que ε tiene un muy alto y permite que se toleren errores de hasta un valor absoluto de 1. Sin embargo, para los costes de 10 y 100, y el resto de valores de ε, la desviación es menor que la de las componentes del PCA (Tabla 17). En cuanto los errores penalizan, el problema de optimización tiene en cuenta los errores y empieza a recolocarse. Como el ruido es impulsivo y el coste es lineal con los errores, esas soluciones tienden a la óptima.

C	ƐPC1	ƐPC2	ƐPC1	ƐPC2	ƐPC1	ƐPC2
ε	1		1,00E-003		1,00E-007	
1	0.9918	-0.9918	1	-1	1	-1
10	0.9994	-0.9994	1	1	1	1
100	0.9995	-0.9995	1	-1	1	-1

Tabla 14: *Desviación de ƐPC1 y ƐPC2 con ruido impulsivo con potencia igual a X_2, respecto de PC1 y PC2 sin ruido.*

Este comportamiento se puede analizar mejor cuando se aumenta la potencia de ruido impulsivo. A continuación se utilizará ruido impulsivo con el doble de potencia que los datos en X_2.

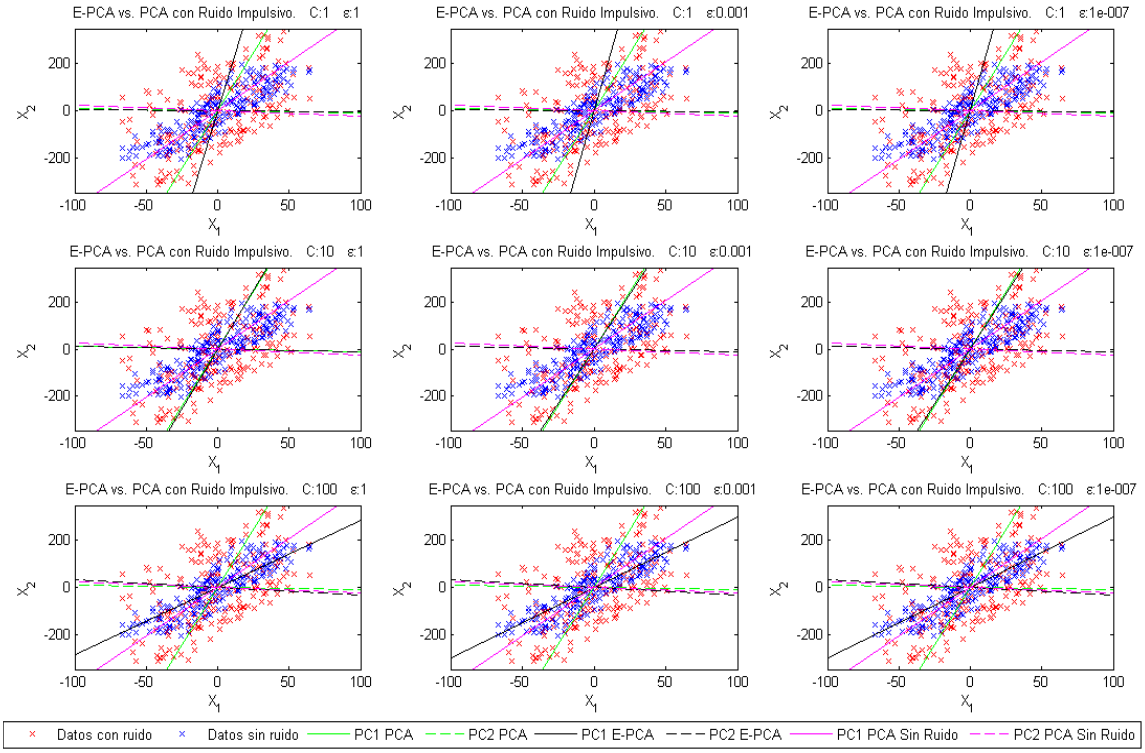

Figura 27: Gráficas de las componentes principales de PCA y Ɛ-PCA para datos con ruido impulsivo con potencia doble a la de X_2.

Al observar la Figura 27 se comprueba que un ruido impulsivo con el doble de potencia que X_2 tiene un mayor influencia en el comportamiento del Ɛ-PCA que en el caso anterior. Ahora el parámetro con más peso en el comportamiento de la dirección de las componentes del Ɛ-PCA es C. Para las simulaciones con una C de 100 las componentes Ɛ-PCA tienen una desviación respecto de la solución óptima claramente menor que las componentes del PCA.

El Ɛ-PCA es capaz de proporcionar la solución similar a la óptima (aquella que coincide con el caso sin ruido), pero para ellos es necesario ajustar adecuadamente los valores de ε y

C. Fijando un ε suficientemente bajo (0.001 ó 1e-7), sólo hay que preocuparse por ajustar adecuadamente C.

C	ΕPC1	ΕPC2	ΕPC1	ΕPC2	ΕPC1	ΕPC2
ε	1		1,00E-003		1,00E-007	
1	-0.9828	-0.9828	-0.9822	-0.9822	-0.9822	-0.9822
10	-0.9905	0.9905	-0.9914	-0.9914	-0.9914	-0.9914
100	0.9951	0.9951	0.9965	0.9965	0.9965	0.9965

Tabla 15: *Desviación de Ε PC1 y Ε PC2 con ruido impulsivo con potencia doble que X_2, respecto de PC1 y PC2 sin ruido para ruido.*

En la Tablar 15 se ve como C proporciona los mayores cambios de dirección, mientras que ε sirve para afinar los resultados. Aunque para los valores de ε iguales a 10^{-3} y 10^{-7} no hay diferencias. Esto es debido a que la mayoría de las desviaciones de los datos son mayores que estos valores de ε y serán controladas por el Ε-PCA a través de coste.

También se puede sacar de la Tablar 15 que las menores desviaciones se logran para una C de 100 y ε de 10^{-3} o 10^{-7}. Mientras que se alcanzan mejores resultados que en el PCA cuando se emplea C igual a 100 o para C igual a 10 y ε con un valor de 10^{-3} o 10^{-7}.

Para terminar de afianzar estos resultados se va a realizar una última serie de simulaciones aumentando el ruido impulsivo hasta cuatro veces la potencia de la variable X_2. En la Figura 28 y en la Tabla 16 se presentan los resultados de estas simulaciones.

Como para el caso del ruido impulsivo con el doble de la potencia, el parámetro más crítico es C. De nuevo eligiendo ε suficientemente bajo y ajustando convenientemente C (en este caso en torno a 100) se alcanza la solución óptima con una desviación menor que las componentes del PCA.

El parámetro ε tienen poca influencia en los cambios de la desviación por lo que es muy fácil de ajustar. Esto es debido a la naturaleza intrínseca del problema presentado al estar diseñado de tal forma que para valores pequeños de ε la desviación de los datos sea manejada por el parámetro C. Se puede ver claramente este efecto en la Tabla 16 al conseguir ε menores cambios en la desviación cuanto más grande es C.

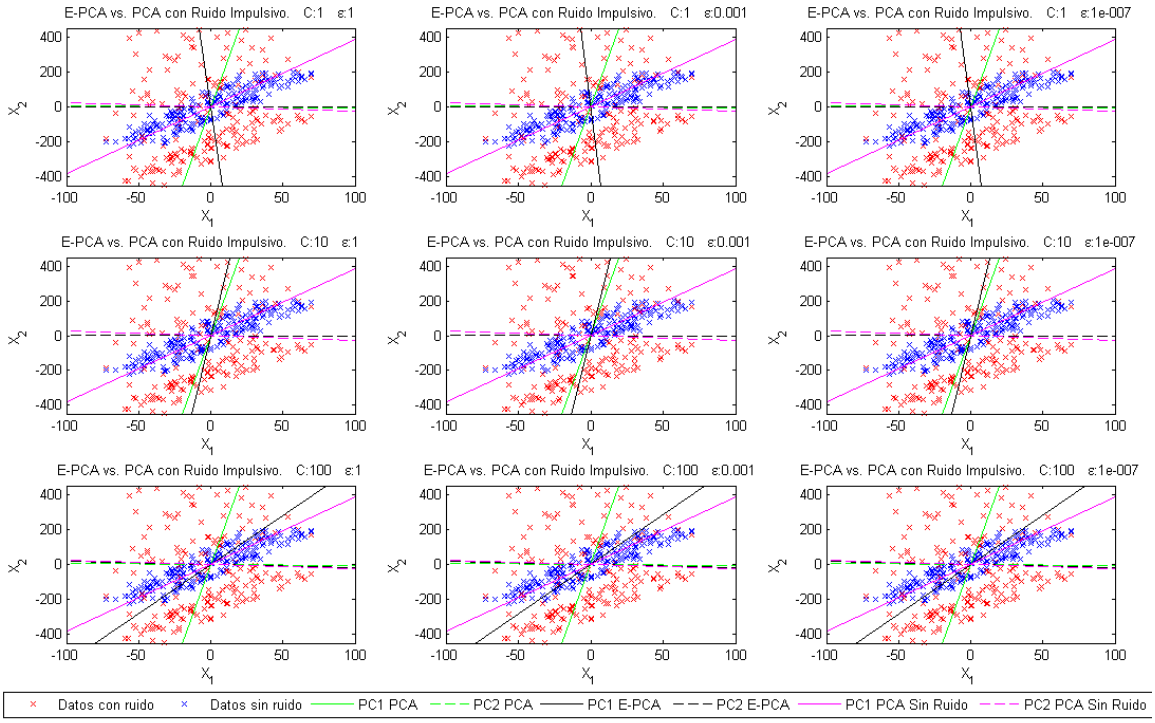

Figura 28: Gráficas de las componentes principales de PCA y Ɛ-PCA para datos con ruido impulsivo con potencia cuádruple a la de X_2.

C	ƐPC1	ƐPC2	ƐPC1	ƐPC2	ƐPC1	ƐPC2
ε	1		1,00E-003		1,00E-007	
1	0.9630	0.9630	0.9635	0.9635	0.9635	0.9635
10	0.9746	-0.9746	0.9747	0.9747	0.9747	0.9747
100	-0.9969	0.9969	-0.9968	-0.9968	-0.9968	-0.9968

Tabla 16: *Desviación de ƐPC1 y ƐPC2 con ruido impulsivo con potencia cuádruple a X_2, respecto de PC1 y PC2 sin ruido.*

En la Tabla 17 se encuentran las desviaciones respecto del resultado óptimo de las componentes principales del PCA según el ruido impulsivo utilizado. Si se comparan estos resultados con los ofrecidos en las Tablas 14, 15 y 16 se puede concluir que para todo las potencias de ruido usadas se han encontrado resultados mejores en el Ɛ-PCA. Además a medida que aumentaba el ruido impulsivo la distancia entre la menor desviación lograda por el Ɛ-PCA y el PCA aumentaba haciendo más valiosas las componentes Ɛ-PCA. También hay que decir que la mínima desviación alcanzada por el Ɛ-PCA para los dos casos con más ruido es similar.

Escala de potencia	PC1	PC2
1	0.9989	-0.9989
2	0.9909	-0.9909
4	0.9777	0.9777

Tabla 17: *Desviación de las componentes principales del PCA para los casos con ruido impulsivo respecto del caso sin ruido.*

Hasta este punto, en esta se ha estudiado el ruido impulsivo sobre todos los datos. A continuación se presenta la Tabla 18 con la evolución de la desviación cuando aplicamos un ruido impulsivo de cincuenta veces la potencia de la señal mientra se varía el número de muestras afectadas. Como se pretende analizar el comportamiento del Ɛ-PCA exclusivamente cuando se varía las muestras con ruido, se utilizaran valores fijos de ε y C. Se ha elegido valores intermedios: ε igual a 10^{-3} y C igual a 10. Además solo se estudiará la primera componente porque como se ha visto durante esta sección la segunda componente tiene una desviación similar.

Porcentaje datos con ruido	PCA	E-PCA
0%	1	0.94069
25%	0.85574	0.93559
50%	0.7588	0.91207
75%	0.51286	0.88918
100%	0.51642	0.85072

Tabla 18: *Desviación de las componentes principales del PCA y del E-PCA según el porcentaje de datos con ruido impulsivo con una potencia 100 veces superior a la de la señal.*

Al haberse elegido un ruido impulsivo con una potencia alta se puede ver de forma sencilla como los resultados del PCA empeoran más rápidamente que los del Ɛ-PCA a medida que se aumenta el número de datos afectados por el ruido.

La conclusión que se obtiene tras este experimento es que el PCA produce las componentes principales óptimas para los casos en los que no hay ruido impulsivo. Sin embargo, para los casos con ruido impulsivo el Ɛ-PCA puede ofrecer mejores componentes si se encuentra la combinación adecuada de los parámetros ε y C. Además a medida que el ruido impulsivo es mayor el PCA elabora peores componentes mientras que el Ɛ-PCA podrá ofrecer componentes sin que el ruido impulsivo le afecte tan gravemente.

5.2 Simulación en un ejemplo sintético tridimensional

Este segundo experimento sigue en la línea del primer experimento realizado en el apartado 5.1. En esta simulación también se emplearan datos artificiales pero ahora cada observación consta de tres variables. Se estudiará el funcionamiento del Ɛ-PCA en contraposición con el del PCA si los datos presentan ruido impulsivo en dos de las tres variables.

La simulación que se va a generar para este experimento consta de un factor oculto que estará escalado por tres valores para formar cada una de las variables. El modelo que se utilizará en esta simulación tendrá 200 observaciones y se presenta a continuación:

$$V \sim N(0, 500^2) + \varepsilon, \qquad \varepsilon \sim N(0, 1)$$
$$X_1 = 0.2V + \varepsilon_1,$$
$$X_2 = 0.8V + \varepsilon_2,$$
$$X_3 = 0.5V + \varepsilon_3,$$
$$\varepsilon_1, \varepsilon_2, \varepsilon_3 \sim N(0, 50^2)$$

Este experimento constará de dos partes. En primer lugar se van a comparar los métodos PCA y Ɛ-PCA para el caso en él que no hay ruido en los datos, y para cuando si hay ruido en una variable. Será una primera aproximación a lo que pretendemos estudiar en este experimento puesto que usaremos unos parámetros fijos para el Ɛ-PCA.

En la segunda parte se estudiará el comportamiento del Ɛ-PCA cuando se varían los valores de sus parámetros para diferentes escenarios de ruido. Estos resultados se confrontarán con las soluciones derivadas del PCA. Finalmente se analizará para cada situación el método que de mejores resultados.

En la Figura 29 se han representado los datos sin ruido y las dos primeras componentes principales de los métodos PCA y Ɛ-PCA. Al haberse empleado datos sin ruido la dirección de las componentes PCA es la dirección óptima. Como se ve en la Figura 29 la primer componente del Ɛ-PCA esta cercana a la componente PCA pero está desviada. Al igual que ocurría en la sección anterior,en el caso sin ruido la solución del Ɛ-PCA es parecida a la ideal, pero hay ligeras diferencias.

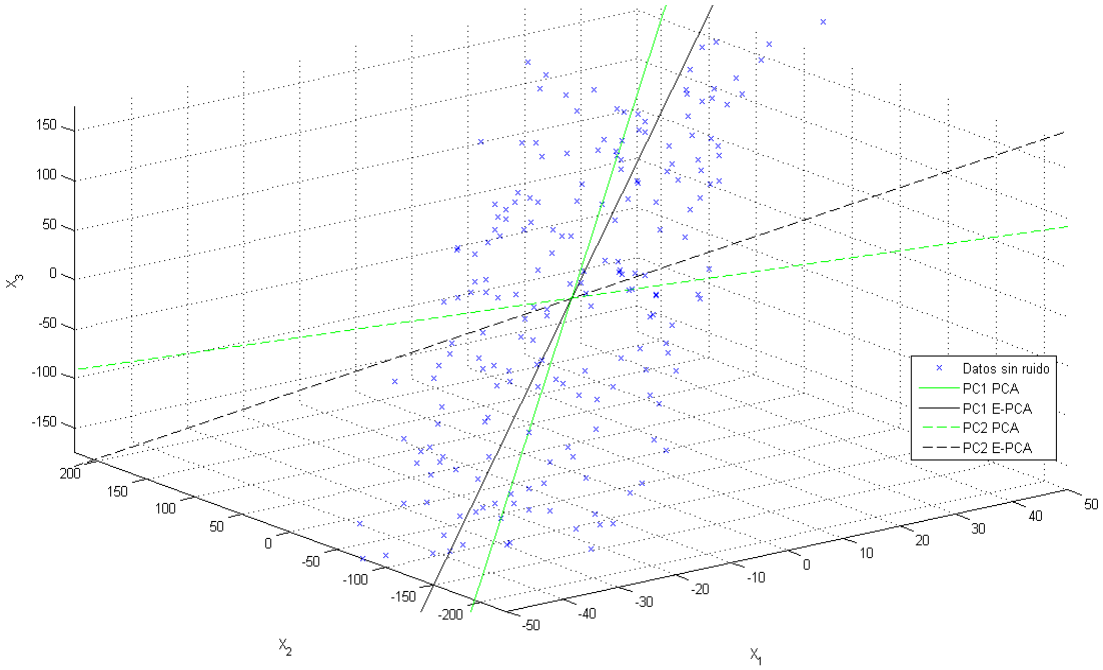

Figura 29: Representación de los datos sin ruido así como de las dos primeras componentes principales halladas mediante PCA y E-PCA

Tanto en el PCA como en el Ɛ-PCA la primera componente contiene respectivamente un 97.8% y un 96.7% de la varianza, como se muestra en la Tabla 19. Las otras dos componentes poseen una cantidad de información despreciable por lo que podrían ser obviadas sin perder prácticamente información. Esto es debido a que aunque el modelo tiene tres variables estas no son independientes y dependen de un único factor oculto. Esto se puede comprobar al observar que el valor absoluto de las variables en la primera componente concuerda con las constantes que escalan el factor oculto para cada variable.

En la Tabla 10 se puede ver también como las variables en la componente principal del Ɛ-PCA se aproximan al valor absoluto de las componentes PCA. Para esta simulación se han utilizado los parámetros ε igual 10^{-5} y el coste a 100, sin embargo se podrían conseguir mejores resultados si se ajustarán mejor estos parámetros.

	PCA			Ɛ-PCA		
	PC1	PC2	PC3	PC1	PC2	PC3
X_1	-0.2080	-0.5392	0.8160	0.3015	-0.8851	0.3542
X_2	-0.8276	-0.3476	-0.4406	0.7825	0.4420	0.4383
X_3	-0.5213	0.7670	0.3739	0.5446	-0.1450	-0.8260
Varianza(%)	97.8883	1,1023	1.0094	96.7955	2.1031	1.1013
Diferencia Coseno	1	1	1	-0.9943	0.2123	-0.2129

Tabla 19: *Resultados del PCA y Ɛ-PCA del para datos sin ruido impulsivo.*

A continuación se le añade ruido impulsivo a X_2 y X_3 con una potencia equivalente a la de lo datos. Se han utilizado los mismos parámetros para Ɛ-PCA que en la simulación sin ruido. En la Tabla 20 y en la Figura 30 se presentan los resultados.

	PCA			E-PCA		
	PC1	PC2	PC3	PC1	PC2	PC3
X_1	-0.1640	-0.6766	-0.9841	0.1824	0.0844	0.9795
X_2	-0.8762	0.4683	0.1138	0.8744	-0.4693	-0.1224
X_3	-0.4531	-0.8809	0.1361	0.4494	0.8789	-0.1595
Varianza(%)	85.8278	12.7743	1.3979	85.797	12.771	1.4310
Diferencia Coseno	0.9970	-0.7087	0.7085	-0.9973	0.6967	-0.6978

Tabla 20: *Resultados para datos con ruido impulsivo.*

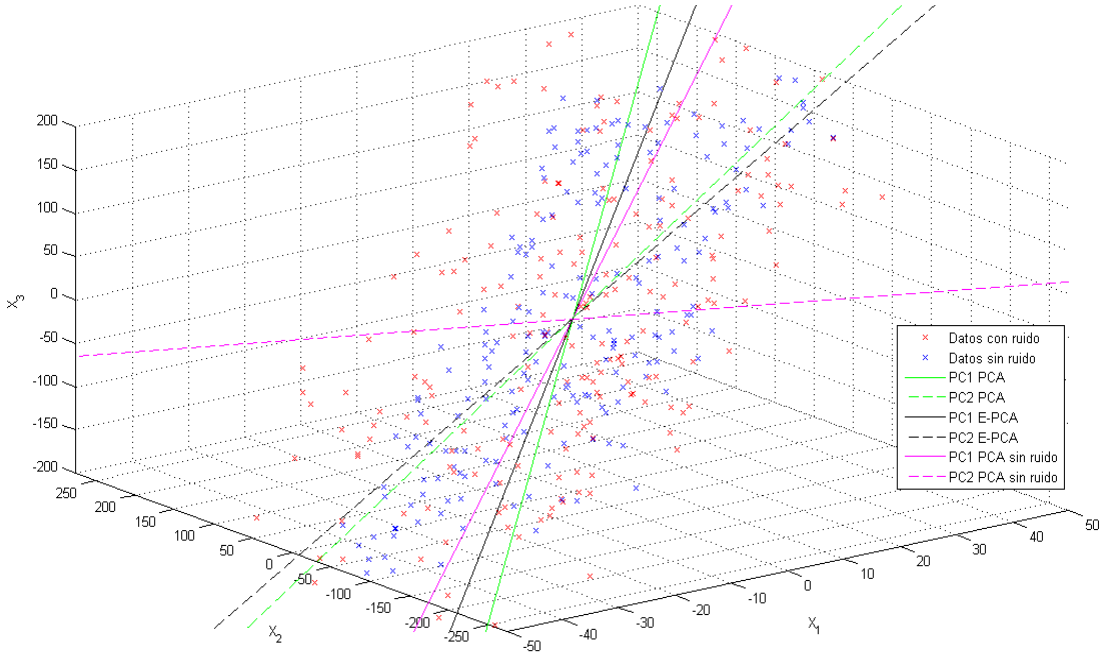

Figura 30: Representación de los datos con ruido y sin ruido. Además de las dos primeras componentes principales de PCA y E-PCA para el caso con ruido.

Al aplicar ruido impulsivo el comportamiento del PCA y del Ɛ-PCA empeoran. Sin embargo, como en el apartado anterior el Ɛ-PCA es más robusto a la presencia del ruido impulsivo. Para unos parámetros ε y C sin los valores óptimos el Ɛ-PCA ofrece mejores resultados que el PCA. En la primera componente principal la desviación es menor.

En la segunda parte de este experimento se va a estudiar como evoluciona la desviación entre la dirección de la primera componente principal del Ɛ-PCA para diferentes valores de ε, C y ruido. Solamente se mostrarán los datos relativos a la primera componente al ser esta la más importante al contener casi toda la información.

En primer lugar se empleará ruido impulsivo sobre las variables X_2 y X_3 con potencia igual a estos datos, como en la primera parte del experimento. Sin embargo, se mostrará la evolución de la diferencia coseno en vez de representar las componentes principales.

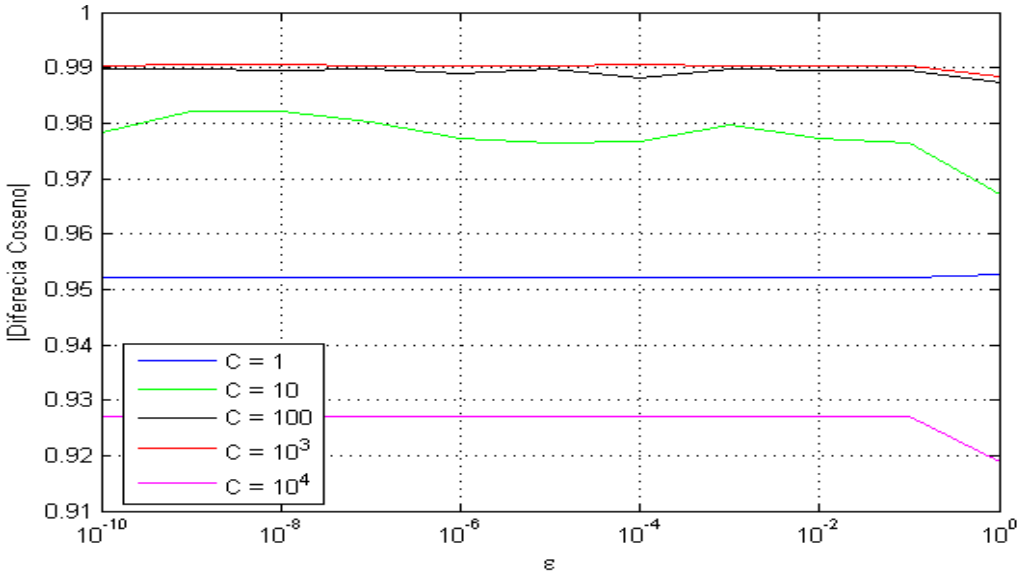

Figura 31: Desviación de la ΕPC1 respecto de la solución óptima cuando se varían C y ε con un ruido impulsivo con la misma potencia que los datos.

En la Figura 31 se puede observar como la evolución de la diferencia coseno para el coste igual a 10^3 es la que ofrece mejores resultados para un ruido impulsivo de la misma potencia que los datos, aunque la del coste 100 está muy cerca. La diferencia coseno para el Ε-PCA con coste 10^3 y ε entre 10^{-10} y 10^{-1} es de 0.9904, mientras que con este ruido impulsivo la distancia coseno de la primera componente principal del PCA es de 0.9271.

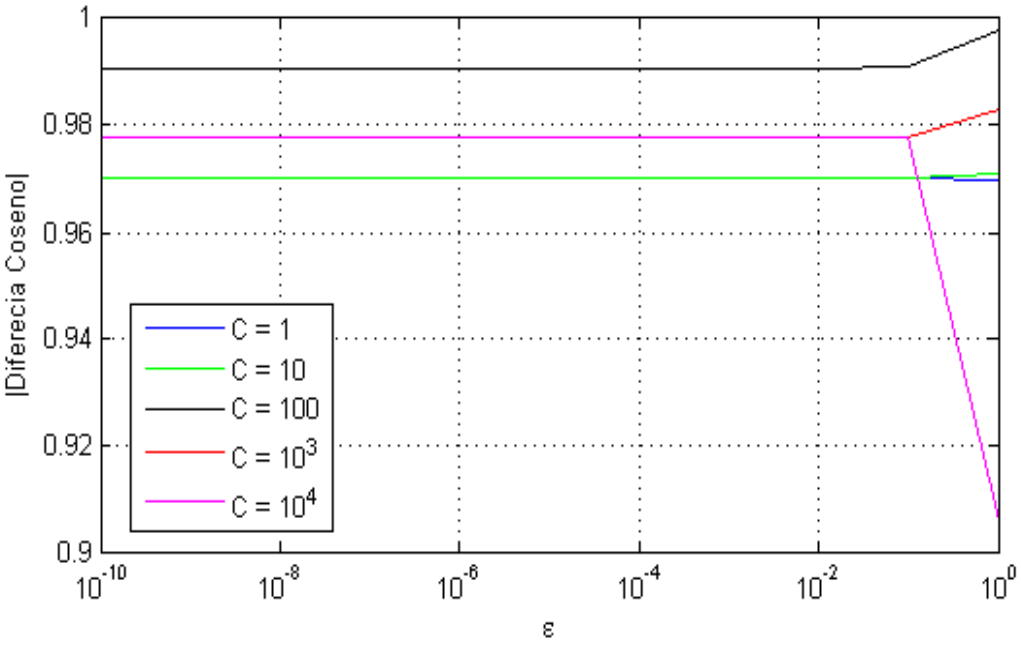

Figura 32: Desviación de la ΕPC1 respecto de la solución óptima cuando se varían C y ε con un ruido impulsivo con potencia doble que los datos.

En la Figura 32 se muestran las evoluciones de la diferencia coseno para un ruido impulsivo de doble potencia que los datos. El coste que ofrece mejores resultados es 100 con una diferencia coseno de 0.9978 para ε igual 1 y 0.9907 para el resto. Mientras que la primera componente del PCA tiene una diferencia coseno de 0.9758.

Finalmente si empleamos una potencia de ruido de cuatro veces la potencia de los datos se obtienen las evoluciones de la desviación de la primera componente del Ɛ-PCA mostradas en la Figura 33.

Para este último caso de ruido impulsivo se consiguen las menores desviaciones para los simulaciones con coste 10^3 y 10^4 que proporcionan la misma diferencia coseno de 0.9391, constante para todos los valores de ε. En este caso la diferencia coseno de la primera componente PCA respecto de la solución óptima es de 0.9148.

Hasta ahora cuando se ha añadido ruido ha sido a todas los datos de X_2 y X_3, por es en la Tabla 21 se muestra como evoluciona la desviación según aumenta el porcentaje de datos afectados por el ruido. Como en el experimento anterior se ha empleado un ruido elevado, en este caso cien veces la potencia de los datos. Como se puede observar el PCA se degrada más rápido que el Ɛ-PCA. Al ser un ruido con una potencia tan alta ya desde el primer momento que se añade los resultados del PCA se deterioran, mientras que los del Ɛ-PCA continúan siendo competitivos.

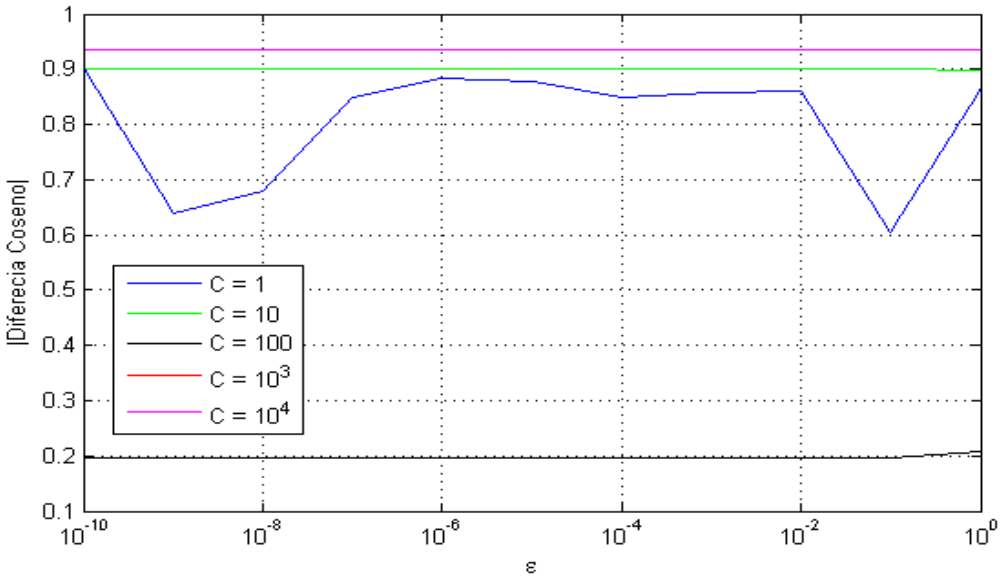

Figura 33: Desviación de la ƐPC1 respecto de la solución óptima cuando se varían C y ε con un ruido impulsivo con potencia cuádruple que los datos.

Porcentaje datos con ruido	PCA			Ɛ-PCA		
	PC1	PC2	PC3	PC1	PC2	PC3
0%	1	1	1	1	0.99528	0.99528
25%	0.75485	0.92188	0.82294	0.93559	0.95531	0.84567
50%	0.69123	0.66895	0.68169	0.91996	0.81858	0.83434
75%	0.51286	0.42396	0.47785	0.90378	0.60233	0.55751
100%	0.42009	0.57629	0.3559	0.8813	0.44357	0.42376

Tabla 21: *Desviación de las componentes principales del PCA y del Ɛ-PCA según el porcentaje de datos con ruido impulsivo con una potencia 100 veces superior a la de la señal.*

En este experimento se ha comprobado que el Ɛ-PCA alcanza buenos resultados si se eligen correctamente los parámetros. Funciona bien para ε bajos, pero C es más crítico y hay que ajustarlo cuidadosamente. Por lo tanto, una buena selección de C y ε permite aislar el comportamiento del Ɛ-PCA frente al ruido impulsivo y acercarnos a la solución ideal.

5.3 Simulación con datos multidimensionales

En este tercer experimento se pretende analizar el comportamiento del Ɛ-PCA sobre datos con varias variables. Para ello se ha elegido un ejemplo artificial conocido al haber sido estudiado anteriormente en el PCA y el SPCA en el apartado 4.1.

En este ejemplo se generaban 10 variables a partir de 3 factores ocultos. Dos de ellos eran independientes mientras que el tercero se formaba a partir de los otros dos. En este experimento se comprobarán los resultados del Ɛ-PCA con los datos del modelo tal cual y a continuación se ira añadiendo ruido impulsivo a las variables. Como el ejemplo tiene 10 variables, estas no se podrán representar gráficamente por lo que se analizará la dirección de las dos primeras componentes principales. Como en los apartados 5.1 y 5.2 la dirección de las componentes principales se estudiará como la desviación respecto de la componentes principales del PCA para datos sin ruido, que será la solución óptima.

Se van a presentar tres escenarios diferentes según la exposición de los datos al ruido impulsivo. Este ruido impulsivo añadido tendrá el doble o el cuádruple de la potencia de los datos. En primer lugar se va a estudiar el caso con datos sin ruido impulsivo.

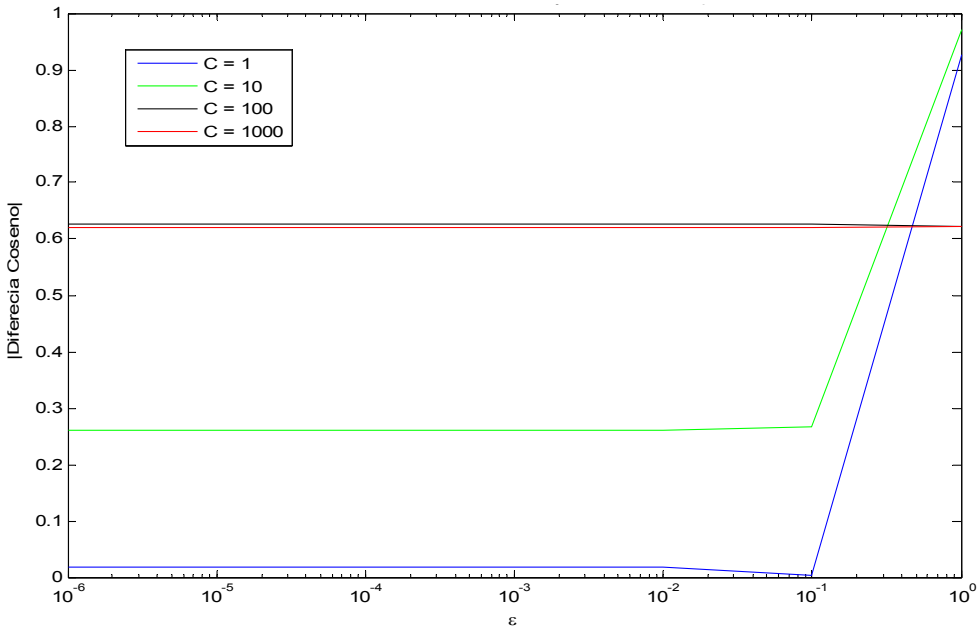

Figura 34: Desviación de la ƐPC1 para diferentes costes y valores de ε sin ruido impulsivo.

Como se observa en la Figura 34 y muestran los resultados en la Tabla 22 el Ɛ-PCA es capaz de ofrecer buenos resultados pero aún así de peor calidad que los resultados el PCA. En la Figura 34 se ve como para el coste 100 y el coste 1000 la desviación de la primera componente del Ɛ-PCA sufre mínimas variaciones respecto a ε. Por lo tanto, como en las secciones anteriores, para valores de ε pequeños es necesario ajustar cuidadosamente C. Sin embargo, tanto en el coste 1 como en el coste 10 existe una gran desviación para todos los valores de ε excepto para 1 cuando presentan una primera componente principal con una dirección bastante cercana a la dirección óptima que posee la primera componente principal del PCA.

Hay que tener en cuenta que en este experimento se está trabajando con 10 dimensiones. Se puede ver que al añadir más dimensiones, los resultados sufren una mayor desviación respecto a la solución ideal del PCA sin ruido, en comparación con los ejemplos anteriores con dos y tres dimensiones.

En el caso sin ruido mostrado en la Figura 34, el Ɛ-PCA empleando un valor de C igual a 10 y ε a 1 entrega el mejor resultado para la Ɛ-PC1. Se pueden observar los valores de las cargas para las dos primeras componentes principales del PCA y del Ɛ-PCA, así como la desviación y la varianza en la Tabla 22.

	PCA		E-PCA	
	PC1	PC2	PC1	PC2
X_1	0.1095	0.4795	0.1380	0.3757
X_2	0.1095	0.4797	0.1378	0.3756
X_3	0.1095	0.4795	0.1381	0.3764
X_4	0.1095	0.4796	0.1379	0.3759
X_5	-0.3938	0.1408	-0.4450	-0.0178
X_6	-0.3937	0.1408	-0.4448	-0.0181
X_7	-0.3936	0.1410	-0.4448	-0.0181
X_8	-0.3937	0.1408	-0.4446	-0.0176
X_9	-0.4074	-0.0144	-0.2572	0.4655
X_{10}	-0.4072	-0.0143	-0.2570	0.4655
Varianza (%)	60.3602	39.6398	72.1892	27.8108
Diferencia Coseno	1	1	0.9706	0.6977

Tabla 22: *Resultados cuando no hay ruido impulsivo para el PCA y Ɛ-PCA empleando C igual a 10 y ε a 1.*

Ahora se introduce ruido impulsivo en los datos. Se realizarán dos simulaciones, la primera tendrá solamente ruido impulsivo en las variables X_4 y X_8. Mientras que la segunda simulación tendrá ruido impulsivo en todas las variables. El ruido impulsivo introducido tendrá una potencia doble a la potencia de las variables a las que se le aplique. Después se repetirán las simulaciones introduciendo ruido impulsivo con potencia cuádruple a la de los datos.

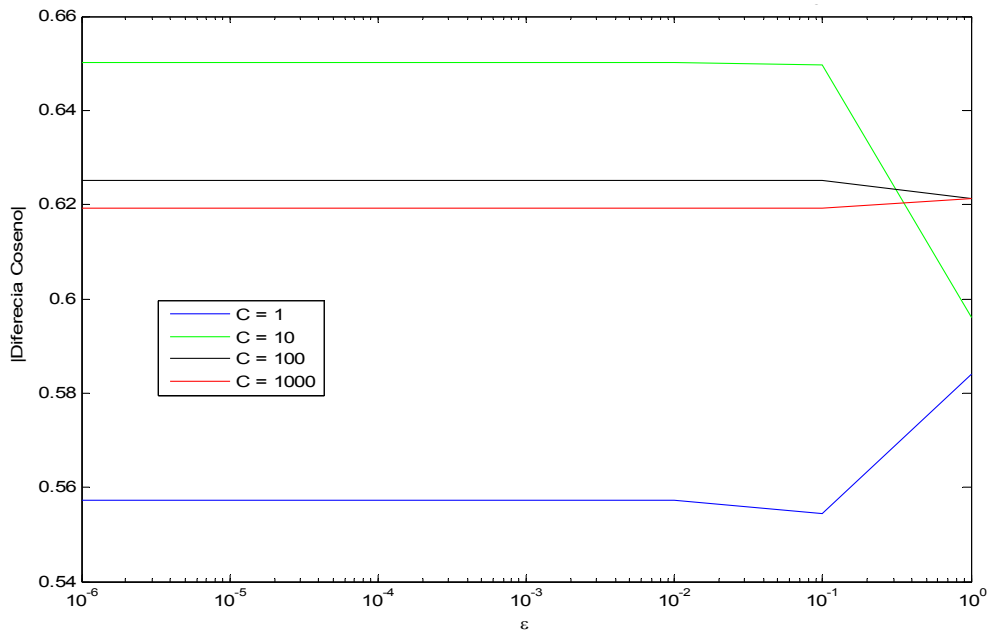

Figura 35: Desviación de Ɛ-PC1 para diferentes costes y valores de ε cuando se introduce ruido impulsivo en X_4 y X_8 de potencia doble a los datos.

En la Figura 35 se observa como la menor desviación no alcanza los resultados para el

caso sin ruido y además está muy por debajo. La menor desviación se alcanza para el coste 10 y cualquier valor de ε inferior a 0.1. Al observar la Tabla 23 se comprueba que los resultados para el PCA también son peores aunque continúan siendo mejores que los del Ɛ-PCA.

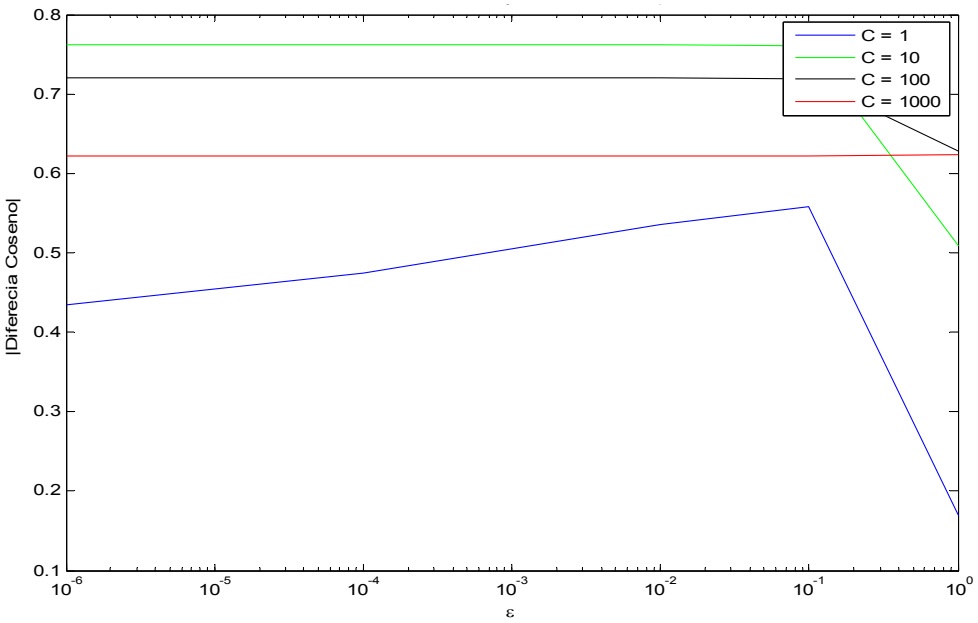

Figura 36: Desviación de Ɛ-PC1 para diferentes costes y valores de ε cuando se introduce ruido impulsivo en todas las variables.

En la Figura 36 se observa como a pesar de añadir ruido impulsivo en todas las variables la primera componente del Ɛ-PCA llega a obtener un menor desviación. Como para el caso sin ruido impulsivo y cuando sólo se introducía el ruido impulsivo en X_4 y X_8 el coste que termina por ofrecer un mejor resultado es el de valor 10.

Si comparamos estos resultados con los del PCA (en la Tabla 23) volvemos a ver que son peores. Los resultados del PCA también mejoran y aunque las componentes no llegan a obtener la dirección óptima se le acercan mucho.

En la Tabla 23 se puede comprobar como afecta el ruido impulsivo en las componentes principales. Al observar las variables X_4 y X_8 cuando solo se añade ruido impulsivo a estas variables se comprueba que no tienen el mismo valor que las otras variables que tienen asociado el mismo valor. Para X_4 son las variables X_1 ,X_2 yX_3, mientas que para X_8 son las variables X_5 ,X_6 yX_7. Para la simulación cuando se introduce ruido impulsivo en todas las variables, los valores de las variables asociadas al mismo factor están son distintos aunque suelen estar cercanos entre si.

	Ruido Impulsivo en X_4 y X_8				Ruido Impulsivo en todo X			
	PCA		Ɛ-PCA		PCA		Ɛ-PCA	
	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2
X_1	0.0458	0.2306	0.1929	0.2376	0.1215	0.4747	0.1836	-0.4676
X_2	0.0457	0.2307	0.1928	0.2379	0.0939	0.4637	0.2054	-0.4394
X_3	0.0458	0.2306	0.1926	0.2381	0.0894	0.4720	0.2145	-0.4109
X_4	0.1394	0.8925	0.3834	0.3101	0.1150	0.4987	0.2165	-0.4514
X_5	-0.2378	0.0240	0.4121	0.0515	-0.3972	0.1796	0.4669	0.1697
X_6	-0.2376	0.0239	0.4120	0.0514	-0.4015	0.1454	0.4456	0.1201
X_7	-0.2376	0.0241	0.4119	0.0517	-0.3887	0.1288	0.4024	0.1436
X_8	-0.8316	0.1939	0.4561	-0.8009	-0.3799	0.1209	0.3797	0.1455
X_9	-0.2378	-0.0468	0.1096	0.2054	-0.4261	-0.0044	0.2779	0.2558
X_{10}	-0.2377	-0.0468	0.1095	0.2056	-0.3987	-0.0576	0.1764	0.2550
Varianza (%)	55.31	44.68	54.94	45.05	54.64	45.35	50.83	49.16
|Diferencia Coseno|	0.8323	0.7988	0.6502	0.3942	0.9992	0.9976	0.7625	0.7744

Tabla 23: *Resultados del PCA y Ɛ-PCA para los casos con ruido impulsivo en X_4 y X_8, y en todas las variables. Los resultados del Ɛ-PCA están obtenido para C = 10 y $\varepsilon=10^{-6}$.*

A continuación repetiremos las simulaciones pero se aumentará el ruido impulsivo hasta cuatro veces la potencia de los datos sin ruido. Ahora solo se mostrará en la Tabla 24 las diferencias coseno. Los valores de las componentes principales seguirán la misma tónica que en la simulación anterior al introducir ruido impulsivo y los resultados que nos pueden aportar información relevante serán únicamente las diferencias coseno.

|Diferencia Coseno|	PCA		Ɛ-PCA	
	PC1	PC2	PC1	PC2
Ruido X_4 y X_8	0.5056	0.5643	0.6403	0.1316
Ruido en todo X	0.9417	0.9366	0.7231	0.6724

Tabla 24: *Resultados del PCA y Ɛ-PCA cuando se añade ruido impulsivo cuádruple a la potencia de los datos .*

En la Tabla 24 se puede observar como el PCA obtiene peores resultado al aumentar el ruido. Aunque cuando se añade ruido impulsivo en todas las variables continúa gozando de una diferencia coseno elevada, no es tan buena que en el caso anterior cuando había un ruido menor. Esto es debido a que como el ruido impulsivo afecta a todas las variables el funcionamiento del PCA no sufre un gran empeoramiento al contrarrestarse los efectos del ruido impulsivo entre las variables asociadas al mismo factor oculto.

Sin embargo, cuando únicamente se añade ruido impulsivo a las variables X_4 y X_8, cada

una asociada a un factor oculto distinto, se deterioran los resultados al tratarse de variables puntuales. Para este caso la diferencia coseno desciende en más de 0.3 para PC1 y 0.23 para PC2. Es en este caso cuando el Ɛ-PCA (C = 1000 y $\varepsilon = 1$) proporciona mejores resultados que el PCA.

Los resultados del Ɛ-PCA han sufrido pocas variaciones al aumentar el valor del ruido impulsivo sobre todos los datos (C = 10 y $\varepsilon = 10^{-6}$). Al estar especialmente diseñado para tratar con puntos desviados de la tendencia general de los datos el aumento del ruido le afecta menos. Como se comprueba en la Tabla 24, el Ɛ-PCA ya es una alternativa al PCA cuando el ruido impulsivo tiene una potencia cuádruple a la de las variables X_4 y X_8 y a medida que aumentemos el valor del ruido impulsivo las diferencias con el PCA se harán mayores. Lo mismo pasa cuando el ruido impulsivo afecta a todas las variables, pero en este caso el ruido tendrá que ser bastante mayor para que el Ɛ-PCA sea competitivo.

En la Tabla 25 se muestran los resultados de añadir un ruido impulsivo con una potencia alta, cien veces la potencia de los datos de entrada. En este caso, no se ha añadido directamente a todas los datos sino que se ha aumentado la proporción de datos afectados progresivamente. Se puede observar que para un ruido tan elevado las desviaciones no sufren grandes variaciones. Aún así el Ɛ-PCA ofrece mejores resultados para todos los casos de ruido impulsivo que el PCA.

%	Ruido Impulsivo en X_4 y X_8				Ruido Impulsivo en todo X			
	PCA		E-PCA		PCA		E-PCA	
	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2
0%	1	1	1	1	1	1	1	1
25%	0.39293	0.4771	0.62236	0.3940	0.27567	0	0.62661	0.30801
50%	0.39357	0.48176	0.60336	0.37082	0.106	0.06403	0.58677	0.16894
75%	0.40115	0.49127	0.57943	0.3044	0.09437	0	0.55028	0.27194
100%	0.39152	0.47793	0.55538	0.43252	0.12845	0.17638	0.60039	0.51765

Tabla 25: *Diferencia coseno entre el PCA original sin ruido, y el PCA y el Ɛ-PCA cuando se añade un ruido impulsivo de 100 veces la potencia de la señal a las variables X_4 y X_8, y a todos el conjunto de datos de entrada según la proporción de datos afectados.*

5.4 Simulación con datos reales

Para completar el estudio del Ɛ-PCA se va a comprobar su funcionamiento sobre cuatro bases de datos reales. De esta forma se podrá conocer las prestaciones que ofrece en un entorno real.

La estructura de esta simulación será similar a la empleada en la sección 4.3. El Ɛ-PCA será la etapa de preprocesado previa a una fase de clasificación. La eficiencia del Ɛ-PCA se medirá mediante el error de clasificación al usar sus datos de salida como la entrada del clasificador. Se compararán sus resultados en cada una de las bases de datos con los resultados al usar el PCA como etapa previa al clasificador y con los resultados de no utilizar ninguna etapa de preprocesado.

El clasificador será un clasificador lineal SVM definido en las librerías LIBSVM empleado anteriormente. Al igual que las bases de datos que vuelve a ser: Optical recognition of handwritten digits (hand), Dermatology (ion), Statlog – Landsat Satellite (landstat) y Waveform Database Generator (wave). Todas estas bases de datos contienen datos reales que deben ser clasificados en dos grupos.

También se empleará validación cruzada de 5 iteraciones para entrenar el parámetro C del clasificador y el los parámetros C y ε del algoritmo Ɛ-PCA.

A continuación se muestran las gráficas de la evolución del error de clasificación dependiendo del número de componentes principales utilizadas. Para todas las bases de datos se han calculado las primeras diez componentes principales, al contener estas más del 99% de la varianza de los datos de entrada.

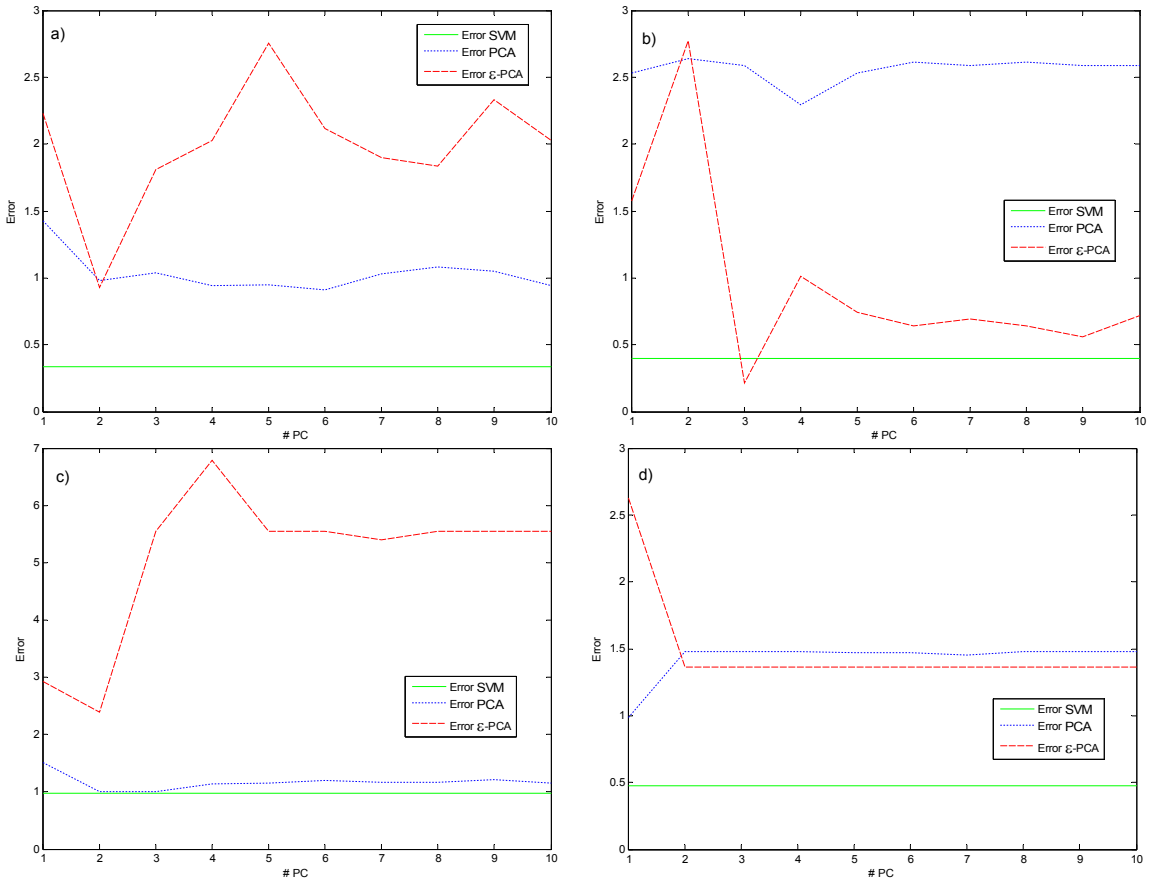

Figura 37: Evolución del error de clasificación según el número de componentes principales utilizadas para el PCA, Ɛ-PCA y usando únicamente el clasificador SVM. En la gráfica a) se ha usado la base de datos hand, en la b) ion, en la c) landstat y en la d) wave.

En primer lugar se puede observar que para todos los casos el error del clasificador directo sin emplear ninguna etapa de preprocesado es el menor. Esto es lógico pues tanto el PCA como el Ɛ-PCA son técnicas no supervisadas que no tienen en cuenta las etapas posteriores del sistema, en este caso el clasificador SVM. Además como uno de sus objetivos es trabajar con un espacio de datos más reducido, la información disponible en los datos que entran en el clasificador es menor para los dos casos con preprocesado de los datos. Este hecho por una parte empeora el error de clasificación, pero por otra, al trabajar con un espacio menor de datos, hace que sea más eficiente entrenar al clasificador.

Es necesario notar que estas bases de datos no tiene un número elevado de variables (hand es la base de datos con más variables, 62). Si se hubieran usado bases de datos con un elevado número de variables, incluso más variables que datos, es muy probable que el escenario fuera diferente y el clasificador funcionara peor sin esta fase de preprocesado. En este ejemplo el objetivo es analizar las prestaciones del Ɛ-PCA frente al PCA, no analizar las ventajas del PCA como etapa de preprocesado. Por lo tanto, hemos preferido emplear

ejemplos sencillos que podamos analizar y procesar con más facilidad.

Respecto a las curvas de error del PCA y del Ɛ-PCA, se puede comprobar que mientras el PCA da como resultado curvas suaves y las del Ɛ-PCA sufren muchas variaciones. También se puede ver que no se puede determinar si el PCA o Ɛ-PCA tiene un comportamiento más útil, dependiendo de cada caso funcionaran mejor o peor. En las gráficas a) y c) el PCA ofrece mejores resultados, en la b) es claramente mejor el Ɛ-PCA, incluso con un pico para el empleo de tres componentes que supera al clasificador directo, y en la d) aunque están muy próximos el Ɛ-PCA da un menor error.

Por lo tanto, la ventaja de la elección del Ɛ-PCA frente a la no utilización de etapas de preprocesado, en este caso el uso directo del clasificador, será una reducción del espacio de datos de trabajo mejorando así la capacidad de entrenamiento del clasificador y disminuyendo su coste de entrenamiento. Mientras que la mejora respecto a otras etapas de preprocesado, especialmente frente al PCA normal, radica exclusivamente para las bases de datos con ruido impulsivo. Para las bases de datos b) y d), en las cuales el Ɛ-PCA ofrece una verdadera mejora, es de suponer que tengan ruido impulsivo por lo que el uso de la función de coste “Ɛ-insensibe” es más adecuado en estos escenarios.

Capítulo 6

Conclusiones y Líneas Futuras

6.1 Conclusiones

Como se ha visto en este documento, el análisis de componentes principales es una técnica no supervisa de extracción de características que hace posible la reducción de las variables de un conjunto de datos de entrada con una pérdida mínima de información. Esta técnica es muy útil, pero tiene varios inconvenientes que pueden ser subsanados con las versiones que hemos propuesto durante este proyecto fin de carrera.

Al abordar el PCA como la minimización del error de reconstrucción se ha desarrollado una formulación que convierte el PCA en un problema de optimización iterativa. De esta manera se pueden introducir cambios fácilmente en el PCA y formular, de manera sencilla, las diferentes extensiones que se han propuesto.

La primera versión presentada ha sido el análisis de componentes principales dispersas. Mediante la introducción en el término de optimización de la regularización L_1 se consigue que las cargas asociadas a cada componente principal tiendan a anularse. Gracias a esta modificación, como hemos visto, las componentes principales no dependen de todas las variables de entrada, mejorando así su interpretación. También se ha comprobado como el parámetro λ_1 condiciona el número de cargas nulas, permitiéndonos controlar el nivel de

interpretabilidad de la solución frente al grado de similitud de la misma con la solución original del PCA.

Asimismo, dentro de esta versión se ha estudiado el uso de la regularización L_2 que en combinación con la regularización L_1 nos ha permitido aplicar el SPCA a conjuntos de datos compuestos por más variables que observaciones. Para este caso es necesario además el control de λ_2.

La segunda versión que se ha presentado ha sido el Ɛ-PCA. Esta técnica modifica la función de coste del PCA por una más robusta frente al ruido impulsivo, la función "ε-insensible" utilizada en las SVM. Con esta nueva formulación hay que ajustar los parámetros C y ε, constatando que una buena elección de los mismos insensibiliza el funcionamiento del algoritmo frente al ruido impulsivo. Se ha constatado que una buena elección de estos parámetros efectivamente aísla el comportamiento frente al ruido impulsivo. Aunque ambos son importantes, ε suele funcionar correctamente con valor bajo para todos los datos por lo que básicamente solo se necesita una elección pormenorizada de C, como ocurre en las SVM. También se ha visto que las ventajas del Ɛ-PCA frente al PCA son más evidentes cuanto más elevado es el ruido impulsivo.

Se ha verificado que el SPCA y el Ɛ-PCA necesitan más componentes principales que el PCA para que los datos de salida contengan la misma varianza. Sin embargo, este incremento del número de componentes principales es muy pequeño comparado con el número de variables de entrada y subsana las desventajas originales del PCA: la interpretación de las componentes principales con el SPCA; y la influencia del ruido impulsivo con el Ɛ-PCA.

Con las simulaciones finales con datos reales en los experimentos de las dos versiones se ha constatado que la introducción en un sistema con un clasificador de una etapa de preprocesado como el PCA, no mejora los resultado en términos de error de clasificación. Sin embargo, proporciona un ahorro computacional muy considerable. Y si en vez del PCA se utilizan sus versiones se pueden alcanzar unos resultado competitivos usando datos con un número de dimensiones muy reducidas fáciles de interpretar o sin la influencia del ruido impulsivo.

6.2 Líneas Futuras

El trabajo llevado a cabo en el presente proyecto fin de carrera puede ser continuado en las siguientes futuras líneas de trabajo:

- Uso de otras técnicas para mejorar la resolución del problema de optimización. En este PFC se ha empleado en MATLAB la toolbox MOSEK. Sin embargo, se puede intentar acelerar el coste computacional mediante otras técnicas.
- Unión del SPCA y del Ɛ-PCA en una nueva extensión del PCA, Ɛ-SPCA, con cargas dispersas y robusta frente al ruido impulsivo. Se utilizaría la regularización de la *red elástica* para las crear las cargas dispersas y la función de coste "ε-insensible" para aislar los resultados del ruido impulsivo.
- Análisis del PCA, SPCA y Ɛ-PCA cuando se utilizan versiones kernel de estos algoritmos. De esta forma se añadiría la características de no-linealidad entre el conjunto de datos de entrada y el conjunto de salida a las características ya existentes en estos métodos.
- En este proyecto se ha partido del algoritmo PCA, que es un método no supervisado que no tiene en cuenta el objetivo del sistema completo, por lo que otra linea de trabajo sería partir de un método supervisado como el PLS, que busque mejorar específicamente las prestaciones de todo el sistema.

Anexo A

Presupuesto

En este último capítulo se presentará el presupuesto necesario para llevar a cabo este proyecto. A partir de la planificación y de los costes asociados se desglosarán todos lo gastos que compondrán el presupuesto final del proyecto.

Planificación

Este proyecto ha sido realizado siguiendo las fases que se presentan a continuación:

- Fase 0. Estudio del estado del arte de las técnicas no supervisadas de extracción de características centrándonos en el análisis de componentes principales y sus versiones.
- Fase1. Estudio en profundidad del PCA.
- Fase 2. Implementación del algoritmo PCA y realización de experimentos.
- Fase 3. Estudio del funcionamiento del algoritmo SPCA e implementación de dicho algoritmo.
- Fase 4. Realización de varios experimentos para analizar el comportamiento del SPCA y para comparar su prestaciones con las del PCA.

- Fase 5. Estudio pormenorizado de las SVM y su aplicación dentro del PCA. Implementación del Ɛ-PCA
- Fase 6. Experimentos con el objetivo de estudiar las características del Ɛ-PCA .
- Fase 7. Realización de la memoria y presentación del proyecto.

A continuación se presenta la Tabla 26 con las fechas de inicio y finalización, y la duración de cada fase del PFC.

	Fecha de inicio	Fecha de finalización	Duración
Fase 0	01/11/2010	20/11/2010	10 días
Fase 1	20/11/2010	29/01/2011	17 días
Fase 2	15/01/2011	22/02/2011	20 días
Fase 3	03/02/2011	10/04/2011	25 días
Fase 4	20/04/2011	15/11/2011	60 días
Fase 5	16/05/2011	12/06/2011	15 días
Fase 6	01/12/2011	22/03/2012	40 días
Fase 7	18/12/2011	11/04/2012	40 días

Tabla 26: Calendario del proyecto. Fechas y duración de cada una de las fases.

Como se puede ver en la Tabla 26, las fases no son excluyentes habiéndose desarrollado varias fases en paralelo.

Finalmente, el número de días empleados en este proyecto ha sido 160. Si la dedicación diaria fue de media de 6 horas, el número total de horas dedicadas a este proyecto suman un total de 960.

Presupuesto

En la siguiente página está detallado el presupuesto a partir de los gastos asociados y la planificación presentada anteriormente.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Andrés Sánchez Mangas

2.- Departamento:

Teoría de la señal y comunicación

3.- Descripción del Proyecto:

- Titulo: **Análisis de Componentes Principales: Versiones dispersas y robustas al ruido impulsivo**
- Duración (meses): **15**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

19.768,50 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Sánchez Mangas, Andrés		Ingeniero	7,30	2.000,00	14.600,00	
					0,00	
		Hombres mes	**7,3**	**Total**	14.600,00	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador Personal	1.000,00	100	15	60	250,00
Router	70,00	100	15	60	17,50
Memoria Flash	25,00	100	15	60	6,25
				Total	273,75

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[a]

Descripción	Empresa	Costes imputable
MATLAB© R2010a	The Math Works, Inc	1.500,00
Material de oficina		100,00
	Total	1.600,00

fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	14.600
Amortización	274
Subcontratación de tareas	0
Costes de funcionamiento	1.600
Costes Indirectos	3.295
Total	19.769

Referencias

[1] N. J. Nilsson, "Introduction to Machine Learning - An Early Draft of a Proposed Textbook". Artificial Intelligence Laboratory, Department of Computer Science, Stanford University. 1996.

[2] I. Guyon, "An Introduction to Variable and Feature Selection". Journal of Machine Learning Research, 3, pp.1157-1182. 2003.

[3] I. Guyon, S. Gunn, M. Nikravesh y L. A. Zdeh, "Feature Extraction: Foundation and Applications". Springer-Verlag Berlin Heidelberg. 2006.

[4] C. M. bishop, "Neural Networks for Pattern Recognition". Oxford University Press. 1995.

[5] D. V. Nguyen y D. M. Rocke, "Tumor classification by partial least squares using microarray gene expression data". Bioinformatics, 18, pp.39-50. 2002.

[6] P. Pellegri, G. Novatti y R. Schettini, "Multispectral loss-less compression using approximation methods". IEEE International Conference on Image Processing, 2005.

[7] J. Arenas-Garcia y K. B. Petersen, “Kernel multivariate analysis in remote sensing feature extraction”. G. Camps-Valls y L. Bruzzone, "Kernels methods for Remote Sensing Data Analysis". John Willey & Sons, Ldt. pp.3-24. 2009.

[8] R. A. Johnson y D. W. Wichern, "Applied Multivariate Statistical Analysis". Prentice Hall. 1986.

[9] I. Jolliffe, "Rotation of principal components: choice of normalization constraints". Journal of Applied Statistics, 22, pp.29-35. 1995.

[10] H. Abdi, "Partial least squares regression and projection on latent structure regression (PLS Regression)". Wiley Interdisciplinary Reviews. Computational Statistics, 2, pp.97-106. 2010.

[11] D. Weenink, "Canonical Correlation Analysis". Proceedings of the Institute of Phonetic Sciences University of Amsterdam, 25, pp.81-99. 2003.

[12] K. Worsley, J. Poline, K. Friston y A. Evans, "Characterizing the response of PET and fMRI data using multivariate linear models (MLM)". NeuroImage, 6, pp.305–319. 1998.

[13] S. Wold, P. Geladi, K. Esbensen y J. Öhman, "Multi-way principal components-and PLS-

analysis". Journal of Chemometrics, 1, pp.41-56. 1989.

[14] P. D. Sampson, A. P. Streissguth, H. M. Barr y F. L. Bookstein, "Neurobehavioral effects of prenatal alcohol: Partial Least Squares analysis". Neurotoxicology and Teratology, 11, pp.477–491. 1989.

[15] R. Rosipal y L. J. Trejo, "Kernel partial least squares regression in reproducing kernel Hilbert space". The Journal of Machine Learning Research, 2, pp.97-123. 2002.

[16] R. W. Preisendorfer y C. D. Mobley, "Principal Component Analysis in Meteorology and Oceanography". Amsterdam: Elsevier. 1988.

[17] E. Beltrami, "Sulle funzioni bilineari". Giornale di Mathematiche di Battaglini, 11. 1973.

[18] M. C. Jordan, "Mémoire sur les Formes Bilinéaires". Journal de Mathematiques Pures et Appliquees, 19, pp.35-54. 1974.

[19] K. Pearson, "On lines and planes of closest fit to systems of points in space". Philosophical Magazine, 6, pp.559-572. 1901.

[20] H. Hotelling, "Analysis of a complex of statistical variables into principal components". Journal of Educational Psychology, 24, 1933.

[21] H. Hotelling, "Simplified calculation of principal components". Psychometrika, 1, pp.27-35. 1936.

[22] M. A. Girshick, "Principal Components". Journal of the American Statistical Association, 31, pp.519-528. 1936.

[23] M. A. Girshick, "On the Sampling Theory of Roots of Determinantal Equations". The Annals of Mathematical Statistics, 10, pp.203-224. 1939.

[24] T. W. Anderson, "Asymptotic theory for principal component analysis". The Annals of Mathematical Statistics, 34, pp.122-148. 1963.

[25] C. R. Rao , "The use and interpretation of principal component analysis in applied research ". Sankhya: The Indian Journal of Statistics, 26, pp.329-358. 1964.

[26] J. N. R. Jeffers, "Two case studies in the application of principal components analysis". Applied Statistics, 16, pp.225-236. 1967.

[27] T. Hastie, R. Tibshirani y J. Friedman, "The Elements of Statiscal Learning; Data mining, Interference and Prediction". Springer Verlag, Nueva York. 2001.

[28] P. Hancock, A. Burton y V. Bruce, "Face processing: human perception and principal components analysis". Memory Cognition, 24, pp.21-40. 1996.

[29] J. Misra, W. Schmitt, D. Hwang, L. Hsiao, S. Gullans y G. Stephanopoulos, "Interactive exploration of microarray gene expression patterns in a reduced dimensional space". Genome Research, 12, pp.1112-1120. 2002.

[30] H. Zou, T. Hastie y R. Tibshirani, "Sparse Principal Component Analysis". Journal of Computational and Graphical Statistics, 15, pp.265-286. 2006.

[31] C. Leng y H. Wang, "On General Adaptive Sparse Principal Component Analysis". Journal of Computational and Graphical Statistics, 1, pp.201-215. 2009.

[32] L. Xu y A. L. Yuille, "Robust principal component analysis by self-organizing rules based

on statistical physics approach". IEEE Transactions on Neural Networks, 6, pp.131-143. 1995.

[33] F. De la Torre y M. Black, "Robust principal component analysis for computer vision". Eighth IEEE International Conference on Computer Vision,1, pp.362-369. 2001.

[34] I. Higuchi y S. Eguchi, "Robust principal component analysis with adaptive selection for tuning parameters". The Journal of Machine Learning Research, 5, pp.453-471. 2004.

[35] C. Croux y G. Haesbroeck, "Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies". Biometrika, 87(3), pp.603-618. 2000.

[36] M. Hubert, P. J. Rousseeuw, y K. Vanden Branden, "ROBPCA: Anew approach to robust principal component analysis". Technometrics, 47, pp.64-79. 2005.

[37] C. Alzate y J. A. K. Suykens, "Kernel Component Analysis Using an Epsilon-Insensitive Robust Loss Function". IEEE Transactions on Neural Networks, 19(9), pp.1583-1598. 2008.

[38] 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.

[39] S. Vines, "Simple principal components". Applied Statistics, 49, pp.441-451. 2000.

[40] J. Cadima y I. Jolliffe, "Loadings and correlations in the interpretation of principal components". Journal of Applied Statistics, 22, pp.203-214. 1995.

[41] I. T. Jolliffe y M. Uddin, "A modified principal component technique based on the lasso". Journal of Computational and Graphical Statistics, 12, pp.531-547. 2003.

[42] R. Tibshirani, "Regresion shrinkage and selection via the lasso". Journal of the Royal Statistics Society, Series B, 58, pp.267-288. 1996.

[43] H. Zou y T. Hastie, "Regression shrinkage and selection via the elastic net, with applications to microarrays". Technical report, Department of Statistics, Stanford Univeristy. 2003. Disponible en http://www.stat.stanford.edu/~hastie/pub.htm

[44] B. Efron, T. Hastie, I. Johnstone y R. Tibshirani, "Least angle regression". The Annals of Statistics, 32, pp.407-499. In press. 2004.

[45] A.N. Tikhonov y V.Y. Arsenin, "Solutions of Ill-posed Problems". W. H. Winston. 1977.

[46] E. López, "Support Vector Regression, Un modelo conectado con la teoría de Robustez". Accesible en www.inca.inf.utfsm.cl/~Erick/downloads/SVR_link_robustez.pdf. 2009.

[47] A. Smola y B. Schölkopf, "A tutorial on support vector regression". Statistics and Computing, 14, pp.199-222. 2004.

[48] V. Vapnik y A. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities". Theory of Probability and its Applications, 16, pp.264-281. 1971.

[49] V. Vapnik y A. Chervonenkis, "A note on one class of perceptrons". Automation and Remote Control, 25. 1964.

[50] V. Vapnik, "Statistical learning theory". John Wiley and Sons, New York. 1998.

[51] K. P. Bennet y O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets". Optimization Methods and Software, 1, pp.23-34. 1992.

[52] C. Cortes y V. Vapnik, "Support vector networks". Machine Learning, 20, pp.273-297. 1995.

[53] J. Shawe-Taylor y N. Cristianini, "Kernel Methods for Pattern Analysis". Cambridge University Press, Cambridge. 2004.

[54] Mosek 2011, "The MOSEK optimization tools version 6.0 (Revision 103), User's manual and reference". Mosek ApS, Denmark, 2011. Software disponible http://mosek.com/

[55] A. d'Aspremont, L. El Ghaoi, M. I. Y. Jordan y G. R. G. Lanckriet, "A direct formulation for Sparse PCA using semidefinite programming". SLAM Review, pp.434-448. 2006.

[56] Chih-Chung Chang y Chih-Jen Lin, "LIBSVM : a library for support vector machines". ACM Transactions on Intelligent Systems and Technology. 2011. Software disponible en http://www.csie.ntu.edu.tw/~cjlin/libsvm

[57] M. Grant y S. Boyd, "Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar)". Springer. Lecture Notes in Control and Information Sciences, pp.95-110. 2008.

[58] M. Grant y S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21". 2011. Disponible en http://cvxr.com/cvx/