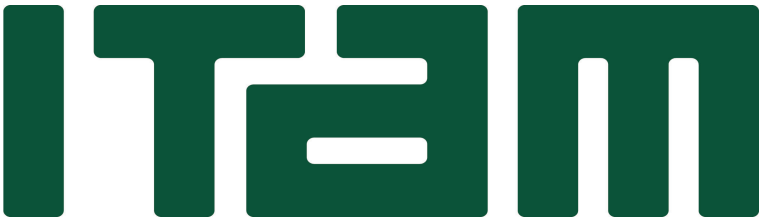
Instituto Tecnológico Autónomo de México



OPTIMIZACIÓN DEL COCIENTE DE LA TRAZA PARA MÁQUINAS DE APRENDIZAJE

TESIS

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS

PRESENTA

SALVADOR GARCÍA GONZÁLEZ

ASESOR

DR. ZEFERINO PARADA GARCÍA

MÉXICO, D.F. 2016

"Con fundamento en los artículos 21 y 27 de la Ley Federal de Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**OPTIMIZACIÓN DEL COCIENTE DE LA TRAZA PARA MÁQUINAS DE APRENDIZAJE**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la biblioteca Raúl Baillères Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una contraprestación"

SALVADOR GARCÍA GONZÁLEZ

_____

Fecha

_____

Firma

Índice general

Índice de figuras

Prólogo

En esta tesis se aborda el problema de optimización planteado por el Análisis Discriminante Lineal de Fisher (ADLF); el cuál, se utilizará para resolver problemas de clasificación. El ADLF busca maximizar un cociente de la forma $Tr(V^T AV)/Tr(V^T BV)$ sobre el conjunto de matrices ortogonales V con p columnas y A, B matrices positivas definidas. Este problema era considerado computacionalmente difícil de resolver, por lo que era reemplazado por otras versiones simplificadas [9] [10]. En esta tesis se busca demostrar que el ADLF, resuelto a través del método de Newton y del algoritmo de Lanczos, se puede resolver fácilmente sin necesidad de versiones simplificadas. Con esto, se logra una precisión comparable con otros algoritmos de clasificación lineales y con un tiempo de cómputo similar.

La tesis consta de tres capítulos y las conclusiones. En el primer capítulo se presenta el problema del ADLF y su solución. En el segundo, se enuncia el método Newton-Lanczos, así como la teoría asociada. En el tercer capítulo se presentan los resultados numéricos. Al final, se presentan las conclusiones a las que se llegaron una vez hechos los experimentos.

Se comienza el primer capítulo introduciendo al ADLF dentro del contexto de aprendizaje de máquina; en particular, como un problema de clasificación lineal. Después, se busca la solución cuando $V \in \mathbb{R}^n$; es decir, cuando $p = 1$. Como siguiente paso, se proporciona la generalización a p dimensiones. Para finalizar el capítulo, se demuestra que el ADLF es equivalente a un problema escalar, por lo que se puede expresar en términos de una $f(\rho)$ y una ρ unidimensional. Una vez dada la solución, se enuncian las condiciones de existencia y un intervalo en donde se encuentra el valor óptimo.

El segundo capítulo aborda el método para resolver el ADLF: Newton-Lanczos. Al inicio, se da una breve presentación de la teoría que sustenta a los métodos de Lanczos, su costo computacional y las ventajas que tienen sobre los métodos tradicionales. Después, se enuncia el algoritmo para alcanzar la solución óptima: el algoritmo de Newton-Lanczos. Al tener como base el método de Newton, se requiere del cómputo de la derivada de $f(\rho)$, por lo que se calcula en este capítulo. Al final, se proporcionan las condiciones necesarias de optimalidad.

En el tercer capítulo se presentan los experimentos numéricos sobre las bases JAFFE y MNIST. Se da una breve introducción de su preprocesamiento, para continuar con un ejemplo donde se proyecta a una dimensión de tamaño 20. Al final, se compara la precisión del ADLF vía Newton-Lacnzos contra el Análisis Discriminante Lineal (ADL) y la Regresión Logística Múltiple (RLM) para diferentes p. Al final, se realiza una comparación del tiempo de cómputo.

Para todo el proyecto se utilizó el lenguaje de programación R y la paquetería *ProjectTemplate*, que sirve para gestionar proyectos de análisis. Para asegurar la portabilidad y reproducibilidad del proyecto se utilizó la paquetería *Packrat*. Por último, para los cálculos, se utilizó la biblioteca de *LAPACK (Fortran)* en su versión para OS X 10.11.4 (*liblapack.dylib)*.

Capítulo 1

Discriminante Lineal de Fisher

En este capítulo se hablará del Análisis Discriminante Lineal de Fisher (ADLF), el cual busca optimizar un cociente de la forma $Tr(V^TAV)/Tr(V^TBV)$ sobre el conjunto de matrices ortogonales V con A, B matrices positivas definidas. Para resolver el ADLF, se han presentado en libros de aprendizaje estadístico y clasificación de patrones formulaciones alternas al problema original, ya que este era considerado computacionalmente muy costoso [10] [9]. Planteamientos como: maximizar la traza de la matriz de dispersión entre clases sujeto a una restricción sobre la matriz de dispersión interna, maximizar la traza del cociente de matrices; o bien, maximizar el cociente de determinantes, han sido formulaciones presentadas en distintos textos [3] [7] [8] [5]. Al final, todas estas propuestas resultan ser versiones simplificadas del problema.

En este capítulo se resolverá el problema original del ADLF a través del método de Newton-Lanczos, el cual resulta ser computacionalmente eficiente. En la primer sección del capítulo se contextualizará al problema dentro del área de Aprendizaje de Máquina. En la segunda parte, se proporcionará la teoría correspondiente para seguir con facilidad el texto. Por último, en la tercera parte, se plantea la solución al problema de ADLF para una dimensión, se generaliza a p dimensiones y se proporcionan las condiciones para la existencia de la solución.

1.1. Aprendizaje de máquina

El Aprendizaje de Máquina toma como base dos áreas de investigación: la Ciencia de la Computación y la Estadística. De la primera, retoma las preguntas: ¿Cómo se pueden construir máquinas que resuelvan problemas? Y ¿Qué problemas son tratables o intratables? De la segunda, toma las preguntas: ¿Qué puede ser inferido de los datos? ¿Bajo que supuestos del modelo? Y ¿Con qué confiabilidad? [8]. El esfuerzo por resolver estas preguntas da como resultado una disciplina enfocada a construir teorías estadístico-computacional de los procesos de aprendizaje.

1.1.1. Procesos de aprendizaje

Se dice que una máquina "aprende"dada una tarea (T), una medición del rendimiento (R) y un tipo de experiencia (E) si el sistema mejora confiablemente su rendimiento (R) en la tarea (T) dada la experiencia (E) [8]. Es decir, se modela una estructura con los datos proporcionados de manera que el rendimiento en la tarea mejora conforme más información recibe. La diversidad de las tareas, así como el campo de aplicaciones es muy diverso, por ejemplo:

- *Clasificación de spam/no-spam*, en el que (E) son los correos, (T) el clasificar correctamente el *spam* y (R) el porcentaje de correos correctamente clasificados.
- *Reconocimiento/Clasificación facial*, en el que (E) son los rostros de distintas personas, (T) el reconocimiento o clasificación de los rostros y (R) el porcentaje de rostros correctamente reconocidos o clasificados.

Los procesos de aprendizaje tienen diversas aplicaciones y distintos supuestos, por lo que han surgido clasificaciones para analizarlos en conjunto. La utilizada en este texto es la propuesta por T. Hastie [7], la cual divide a los métodos en dos grupos: aprendizaje supervisado y aprendizaje no supervisado. El primero, supone la presencia de una variable de salida que actúa como guía en la construcción de la estructura. Ejemplos de éste es la regresión lineal, los árboles de decisión y las máquinas de soporte vectorial. Por otra parte, el aprendizaje no supervisado solo cuenta con la información de las variables independientes; por

ejemplo, análisis de conglomerados, reglas de asociación y reducción dimensional.

Después de esta primer clasificación, se subclasifica a los métodos de aprendizaje supervisado de acuerdo al tipo de variable de salida (Figura 1.1). [1] Cuando se trata de una variable cuantitativa recibe el nombre de regresión, mientras que en el caso de cualitativas se le llama clasificación. Por otra parte, el aprendizaje no supervisado tiene dos ramas en las que el texto hace énfasis [7]: Segmentación en el caso que se desee asignar un grupo a cada individuo, de manera que los grupos sean homogéneos entre sí; o bien, reducción dimensional cuando solamente se desea proyectar a los individuos en un espacio de menor dimensión, de manera que se cumplan características especiales en dicho subespacio.

El problema de ADLF pertenece a la rama de aprendizaje supervisado, en particular a los métodos clasificación. Alternativas para esta finalidad, y que sigan asumiento linealidad en la frontera, son la Regresión Logística, el Análisis Discriminante Lineal y las Máquinas de Soporte Vectorial.

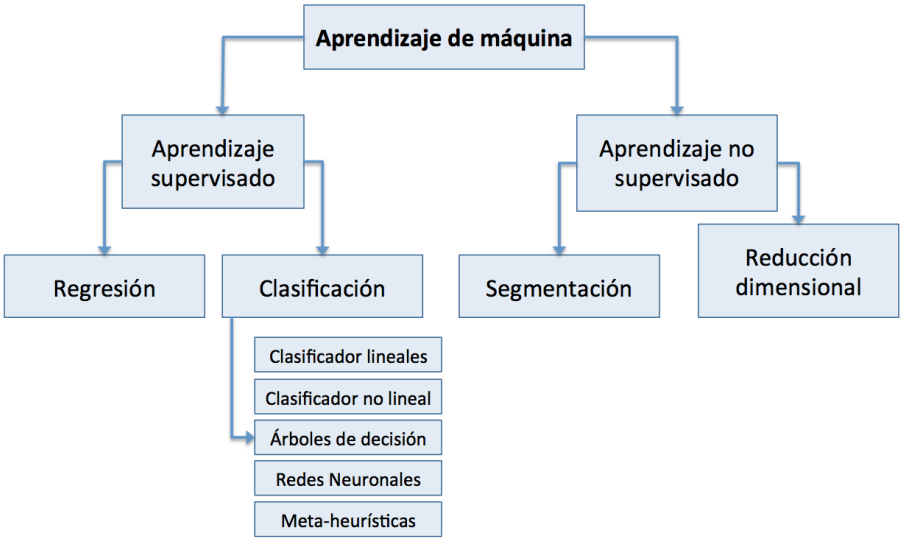


Figura 1.1: Distintos enfoques para resolver problemas de clasificación en el área de Aprendizaje de Máquina.

[1] A lo largo del texto se usará indiferentemente variable de entrada como "input" o variable independiente y variable de salida como "output" o variable dependiente

1.2. Matrices de dispersión

Se comenzará definiendo la nomenclatura necesaria para la sección. Sea x_i el individuo i que pertenece a la clase y_i, N_k el número de personas en la clase k, N el número total de personas y $w_i = V^T x_i$; es decir, los datos proyectados con la matriz V. Entonces, se definen las medias de grupo k como μ_k y la media de todos los datos x_i como μ:

$$\mu_k = \frac{1}{N_k} \sum_{\substack{i=1 \\ y_i=k}}^{N} x_i \tag{1.1}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1.2}$$

Por otro lado, se definen las medias correspondientes a los datos proyectados w_i:

$$\widetilde{\mu_k} = \frac{1}{N_k} \sum_{\substack{i=1 \\ y_i=k}}^{N} w_i \tag{1.3}$$

$$\widetilde{\mu} = \frac{1}{N} \sum_{i=1}^{N} w_i \tag{1.4}$$

El ADLF hace amplio uso de las matrices de dispersión, en específico de la matriz de covarianza, la matriz de dispersión de todos los individuos, la matriz de dispersión interna y la matriz de dispersión entre clases. Es importante analizar a profundidad la terminología y las fórmulas que se usarán a lo largo de la tesis para entender la lógica detrás de la formulación.

Sea Σ la matriz de covarianza (*Covariance Matrix)* de todos los individuos. Se define como $\widehat{\Sigma}$ al estimador insesgado de Σ el cual está escalada entre $N-1$:

$$\widehat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T \tag{1.5}$$

Si esta matriz no está escalada por $N-1$ entonces se le conoce como matriz de dispersión (*Scatter Matrix*), en esta tesis se representará como S_T, con el subíndice T que significa que está tomando en cuenta a todos los individuos:

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T \tag{1.6}$$

Cuando solo se toman a los individuos de una clase particular k, se puede encontrar su correspondiente matriz de dispersión, representada como S_k, con el subíndice k simbolizando que está tomando en cuenta solo a los individuos de la clase k:

$$S_k = \sum_{\substack{i=1 \\ y_i=k}}^{N} (x_k - \mu)(x_k - \mu)^T$$

De esta manera se define la matriz de dispersión interna (*Within-class scatter matrix*) como la suma sobre k de todas las matrices de dispersión de cada clase:

$$S_I = \sum_{k=1}^{K} \sum_{\substack{i=1 \\ y_i=k}}^{N} (x_i - \mu_k)(x_i - \mu_k)^T \tag{1.7}$$

Ahora solo falta definir la matriz de dispersión entre clases (*Between-class scatter matrix*) como la suma de diferencias al cuadrado de las medias de clase contra la media de todos los datos multiplicada por el número de individuos en cada clase N_k:

$$S_E = \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{1.8}$$

Entre la matriz de dispersión interna S_I, la matriz de dispersión entre clases S_E y la matriz de dispersión total S_T existe una relación importante. Se cumple que $S_T = S_I + S_E$; es decir, la dispersión de las medias de grupos con la media global más la dispersión de cada clase individual es igual a la dispersión de los datos sin la información de las clases. Los datos de la figura 1.2 representan las distancias que toman en cuenta cada una de estas matrices. Para ejemplificar esta relación se generaron 10 datos por clase suponiendo distribuciones normales (El coeficiente de correlación de los datos generados es -0.005):

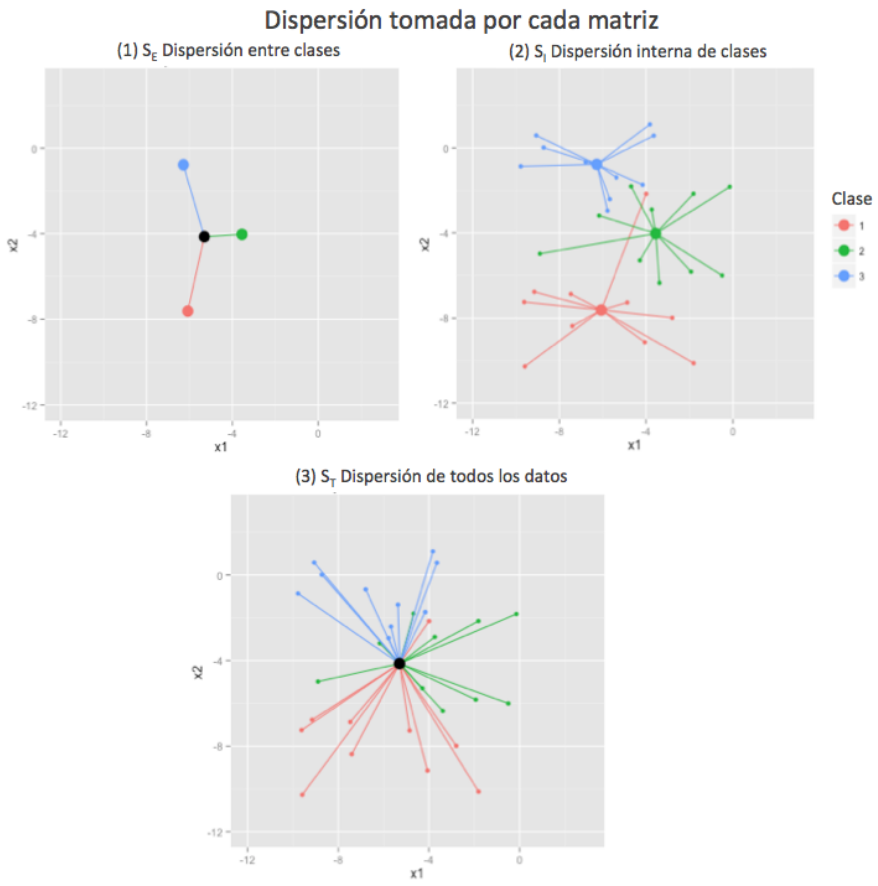


Figura 1.2: En la gráfica (1) se representa la S_E, es decir las distancias al cuadrado entre la media de todos los datos (Punto negro) y las medias de cada clase (Puntos de color gruesos). La gráfica (2) representa S_I; es decir, la distancia al cuadrado de los individuos a la media de su clase. La gráfica (3) representa S_T, la dispersión de los datos con respecto a la media de todos.

Clase	**Distribución x1**	**Distribución x2**
1	N(-5, 2.5)	N(-8, 2)
2	N(-3, 2.5)	N(-4, 2)
3	N(-7, 2.5)	N(-1, 2)

Calculando las matrices de dispersión de acuerdo a las fórmulas (1.6), (1.7), (1.8):

S_I	S_E	S_T
$\begin{bmatrix} 186.05 & 2.78 \\ 2.78 & 94.58 \end{bmatrix}$	$\begin{bmatrix} 46.13 & -4.15 \\ -4.15 & 234.57 \end{bmatrix}$	$\begin{bmatrix} 232.18 & -1.36 \\ -1.36 & 329.16 \end{bmatrix}$

De este ejemplo numérico se puede ver que al sumar la dispersión interna S_I y la dispersión entre clases S_E da como resultado la dispersión de todos los individuos S_T. En general este resultado se cumple, por lo que a continuación se enuncia esta relación muy fácil de demostrar.

Proposición 1.1. *Sea S_E la matriz de dispersión entre clases, S_I la matriz de dispersión interna y S_T la matriz de dispersión de los datos, entonces se tiene que cumplir la siguiente igualdad: $S_T = S_I + S_E$*

Un problema muy común que surge en problemas de aprendizaje estadístico es que el costo computacional puede volverse intratable conforme la dimensionalidad de los individuos crece. En el ADLF se requiere hacer el cómputo de las matrices de dispersión de los individuos constantemente (o bien calcular la inversa de matrices de alta dimensionalidad), cálculos que para grandes dimensiones son sumamente costosos. Existen distintas maneras para hacer frente a este problema, uno de ellos involucra el PCA (Principal Component Analysis) en el preprocesamiento de los datos. Este método es fácil de calcular y solo requiere computar una vez la de matriz de dispersión [9]. Debido a la finalidad de esta tesis no se profundizará en más técnicas para hacer frente a este problema, pero en textos como [7], [3] aparecen distintos métodos para reducción de dimensionalidad.

Retomando el problema de cociente de trazas, lo que se busca es encontrar la proyección que mantenga juntos individuos de una clase al mismo tiempo que separa las medias de distintas clases. Una vez obtenida esta proyección se puede encontrar un hiperplano separador de los datos, o bien algún criterio para asignar la clase de pertenencia.

El problema de optimización se puede plantear como:

$$\max_{\substack{V\in\mathbb{R}^{n\times p}\\ V^TCV=I}} \frac{Tr(V^TS_EV)}{Tr(V^TS_IV)} \tag{1.9}$$

La solución a este problema no tiene una forma cerrada, por lo que en la literatura se buscan formulaciones alternas para resolverlo de una manera más sencilla [10] [5], algunos ejemplos de estas formulaciones son:

$$\max_{\substack{V\in\mathbb{R}^{n\times p}\\ V^TS_IV=I}} Tr(V^TS_EV) \tag{1.10}$$

$$\max_{\substack{V\in\mathbb{R}^{n\times p}\\ V^TCV=I}} Tr\left(\frac{V^TS_EV}{V^TS_IV}\right) \tag{1.11}$$

$$\max_{\substack{V\in\mathbb{R}^{n\times p}\\ V^TCV=I}} \frac{|V^TS_EV|}{|V^TS_IV|} \tag{1.12}$$

Con $|\bullet| = det(\bullet)$ y $Tr(\bullet) = Traza(\bullet)$.

En la siguiente parte de este capítulo se resolverá el problema original (1.9) para $p = 1$, para lo cual se introduce el cociente generalizado de Rayleigh. Para la generalización a p dimensiones solo se plantea el problema y se proporcionan los casos en que la solución existe y es única. Seguido de esto se definirá una función $f(\rho)$ la cual sirve para encontrar el óptimo por métodos iterativos. Por último haciendo uso de los eigenvalores de S_I y S_E se darán cotas inferiores y superiores al óptimo.

1.3. Problema del cociente de trazas

El problema del cociente de trazas (Trace ratio problem) es fácil de ver cuando $V \in \mathbb{R}^{n\times p}$ proyecta a un espacio de pocas dimensiones. Por ejemplo, cuando $p = 2$ se desea obtener la mejor proyección sobre un plano y cuando $p = 1$ sobre una recta. Para ejemplificar esta situación se creo un conjunto sintético donde cada $x_i \in \mathbb{R}^3$. Las distribuciones son normales y se proyectan en $\mathbb{R}^2$ y $\mathbb{R}^1$. Los datos se pueden observar en la figura 1.3.

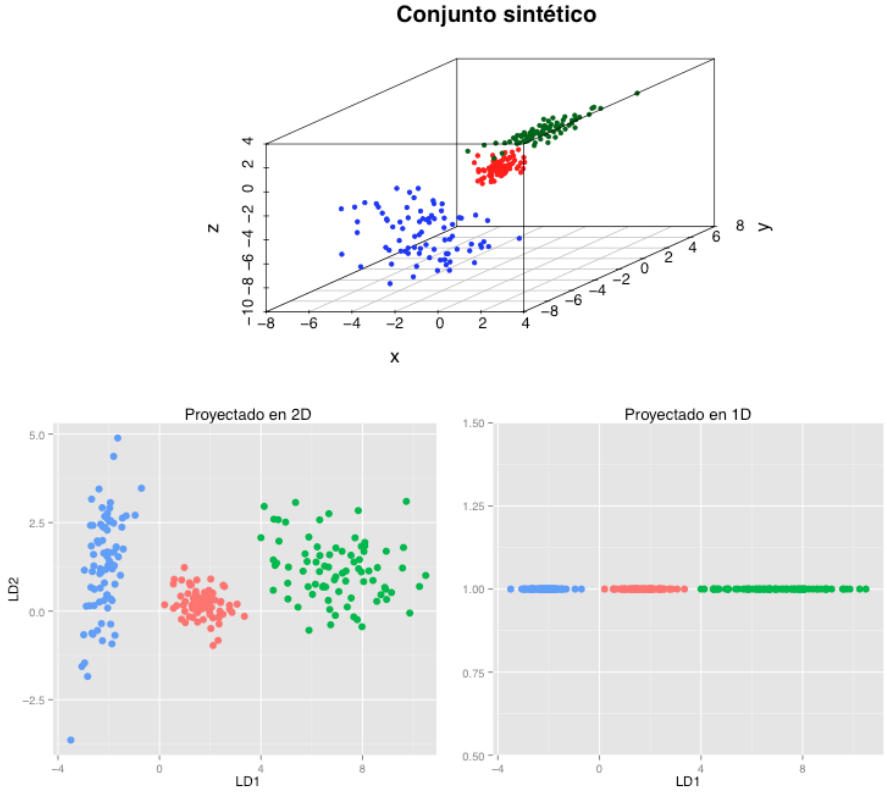


Figura 1.3: En la gráfica de arriba se muestran los datos originales en $\mathbb{R}^3$ los cuales fueron generados a través de distribuciones normales con distintas medias y varianzas. En la gráfica de abajo a la izquierda se muestra la mejor proyección en $\mathbb{R}^2$ y abajo a la derecha la mejor proyección en $\mathbb{R}$

1.3.1. Solución cuando p = 1

El problema (1.9) toma la siguiente forma cuando $V \in \mathbb{R}^n$. Se nombrara v a este proyector de una dimensión ya que resulta ser solo un vector:

$$\max_{v \in \mathbb{R}^n} \frac{v^T S_E v}{v^T S_I v} \tag{1.13}$$

Se tiene que $x_i \in \mathbb{R}^n$ son los individuos originales con $i = 1, ..., N$. Entonces sean $w_i \in \mathbb{R}$ los individuos proyectados por el vector v de manera que $w_i = v^T x_i$. De esta manera es conveniente definir $\widehat{\mu}_k = v^T \mu_k$ y $\widehat{\mu} = v^T \mu$ como la media por

clase y la media total de los datos proyectados.

Matriz de dispersión entre clases Φ_E de los individuos proyectados w_i:

$$\Phi_E = \sum_{k=1}^{K} N_k(\widehat{\mu}_k - \widehat{\mu})^2$$

$$\Phi_E = \sum_{k=1}^{K} N_k(v^T\mu_k - v^T\mu)^2$$

$$\Phi_E = \sum_{k=1}^{K} N_k v^T(\mu_k - \mu)(\mu_k - \mu)^T v$$

Por distributividad en matrices se cumple que $vAv+vBv = v(A+B)v$, entonces:

$$\Phi_E = v^T\big[\sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T\big]v \tag{1.14}$$

Matriz de dispersión intra clase Φ_I de los individuos proyectados w_i:

$$\Phi_I = \sum_{k=1}^{K}\sum_{\substack{i=1\\ y_i=k}}^{N} (w_i - \widehat{\mu}_k)^2$$

$$\Phi_I = \sum_{k=1}^{K}\sum_{\substack{i=1\\ y_i=k}}^{N} (v_i^T x_i - v_i^T\mu_k)^2$$

$$\Phi_I = \sum_{k=1}^{K}\sum_{\substack{i=1\\ y_i=k}}^{N} v^T(x_i - \mu_k)(x_i - \mu_k)^T v$$

Usando de nuevo la distibutividad de matrices:

$$\Phi_I = v^T\big[\sum_{k=1}^{K}\sum_{\substack{i=1\\ y_i=k}}^{N} (x_i - \mu_k)(x_i - \mu_k)^T\big]v \tag{1.15}$$

Las fórmulas de Φ_I y Φ_E de los individuos w_i se pueden expresar en función de las matrices de dispersión intra clase y entre clases S_I y S_E de los individuos originales x_i. De esta manera:

$$\Phi_E = f(S_E) = v^T S_E v$$
$$\Phi_I = f(S_I) = v^T S_I v$$

Se tiene que $\Phi_I, \Phi_E \in \mathbb{R}$, entonces maximizar el cociente $\frac{\Phi_E}{\Phi_I}$ con respecto a v tiene como resultado una proyección que conserva cerca a los individuos pertenecientes a la misma clase, mientras que aleja a los centros de cada clase. Para el caso de una dimensión se puede encontrar una solución cerrada. La teoría asociada a este problema de maximización esta relacionada con el Cociente Generalizado de Rayleigh, el cual bajo las condiciones enunciadas de este caso, se puede transformar a un Cociente de Rayleight. Usando la proposición 1.2 se puede obtener la solución a este último.

Proposición 1.2. *La solución a la maximización del Cociente de Rayleigh:*

$$\max_{v \in \mathbb{R}^n} \frac{v^T A v}{v^T v}$$

cuando A es simétrica, es obtenida cuando v es el eigenvector asociado al eigenvalor más grande de la matriz A.

1.3.2. Generalización a p dimensiones

Para dimensiones más grandes de v, el Cociente Generalizado de Rayleigh no puede ser escrito en general como el Cociente de Rayleigh, por lo que la solución planteada en el capítulo anterior no es de utilidad. Esto genera la dificultad de no tener una solución cerrada, por lo que se han propuesto métodos iterativos y planteamientos alternos a la solución.

La generalización a p dimensiones implica que los individuos $x_i \in \mathbb{R}^n$ son proyectados ahora por la matriz $V = (V_1|V_2|...|V_p)$, de manera que $w_i = V^T x_i$ con $w_i \in \mathbb{R}^p$ y $V_j \in \mathbb{R}^n$. De esta manera las matrices Φ_I y Φ_E se definen como sigue:

$$\Phi_E = \sum_{k=1}^{K} N_k ||\widehat{\mu}_k - \widehat{\mu}||_2^2$$

$$\Phi_E = \sum_{k=1}^{K} N_k ||V^T \mu_k - V^T \mu||_2^2$$

$$\Phi_E = \sum_{k=1}^{K} N_k ||V^T(\mu_k - \mu)||_2^2$$

$$\Phi_E = \sum_{k=1}^{K} N_k \big[(V_1^T(\mu_k - \mu))^2 + (V_2^T(\mu_k - \mu))^2 + ... + (V_p^T(\mu_k - \mu))^2\big] \quad (1.16)$$

De esta expresión hay que destacar que $V_1^T(\mu_k - \mu)$ es un escalar, ya que $V_1 \in \mathbb{R}^n$ y $(\mu_k - \mu) \in \mathbb{R}^n$. Otra fórmula equivalente y que es comúnmente usada por sus propiedades algebraicas consiste en la siguiente expresión:

$$\Phi_E = \sum_{k=1}^{K} N_k Tr\big[V^T(\mu_k - \mu)(\mu_k - \mu)^T V\big] \quad (1.17)$$

Para ejemplificarla se toma una clase $k = k_1$. Al desarrollar $(\bullet) = V^T(\mu_1 - \mu)(\mu_1 - \mu)^T V$ se tiene una matriz en $\mathbb{R}^{p \times p}$ igual a:

$$(\bullet) = \begin{pmatrix} V_1^T(\mu_1 - \mu) \\ V_2^T(\mu_1 - \mu) \\ \vdots \\ V_p^T(\mu_1 - \mu) \end{pmatrix} \begin{pmatrix} (\mu_1 - \mu)^T V_1 & (\mu_1 - \mu)^T V_2 & \dots & (\mu_1 - \mu)^T V_p \end{pmatrix}$$

$$(\bullet) = \begin{pmatrix} V_1^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_1 & \dots & V_1^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_p \\ V_2^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_1 & \dots & V_2^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_p \\ \vdots & \ddots & \vdots \\ V_p^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_1 & \dots & V_p^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_p \end{pmatrix}$$

$$(\bullet) = \begin{pmatrix} (V_1^T(\mu_1 - \mu))^2 & \dots & V_1^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_p \\ V_2^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_1 & \dots & V_2^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_p \\ \vdots & \ddots & \vdots \\ V_p^T(\mu_1 - \mu)(\mu_1 - \mu)^T V_1 & \dots & (V_p^T(\mu_1 - \mu))^2 \end{pmatrix}$$

Por lo tanto al calcular la traza de la matriz de $p \times p$ desarrollada arriba, se tiene que $Tr(V^T(\mu_1 - \mu)(\mu_1 - \mu)^T V)$ es equivalente a:

$$Tr(\bullet) = (V_1^T(\mu_1 - \mu))^2 + (V_2^T(\mu_1 - \mu))^2 + \ldots + (V_p^T(\mu_1 - \mu))^2$$

Al generalizar a las K clases y usando la propiedad de linealidad en la traza; es decir, $Tr(A + B) = Tr(A) + Tr(B)$, entonces se puede escribir de la siguiente manera:

$$\Phi_E = Tr \sum_{k=1}^{K} N_k \big[V^T(\mu_k - \mu)(\mu_k - \mu)^T V\big]$$

Esta expresión es equivalente a (1.16). Como paso final se factoriza V^T y V sobre todos los sumandos, lo que nos llevaría a lo siguiente:

$$\Phi_E = Tr(V^T \sum_{k=1}^{K} N_k \big[(\mu_k - \mu)(\mu_k - \mu)^T\big] V)$$

o, expresada en términos de $S_E = \sum_{k=1}^{K} N_k \big[(\mu_k - \mu)(\mu_k - \mu)^T\big]$

$$\Phi_E = Tr(V^T S_E V) \tag{1.18}$$

Similarmente se puede llegar a la formulación de la varianza intra-clase Φ_I.

$$\Phi_I = Tr(V^T S_I V) \tag{1.19}$$

1.3.3. Existencia de la solución

Para demostrar la existencia y unicidad de la solución, las matrices S_I y S_E deben cumplir ciertas características. Sean $A = S_E$ y $B = S_I$, la primer condición que se les impone es que sean positivas definidas. La razón que apoya la restricción está relacionada con la forma de la función a maximizar, que es un cociente. Como B se encuentra en el denominador, se tiene que evitar que $Tr(V^T BV) = 0$, ya que con este valor se indetermina la función objetivo [9].

T.T. Ngo propone generalizar el estudio a las matrices positivas semidefinidas. Para esto se deben encontrar los casos en que $Tr(V^T BV)$ toma el valor de 0.

Si se diagonaliza a la matriz $B = Q\Lambda_B Q^T$ con Q ortogonal y Λ_B una matriz diagonal con entradas iguales a los eigenvalores de B, entonces:

$$Tr(\Lambda_B) = \lambda_{B_1} + \lambda_{B_2} + ... + \lambda_{B_n} \qquad con \qquad \widehat{V} = Q^T V$$

De este modo $\widehat{V} = (\widehat{V}_1|\widehat{V}_2|...|\widehat{V}_p)$ y cada $\widehat{V}_i^T = (\widehat{V}_{i1}, \widehat{V}_{i2}, ..., \widehat{V}_{in})$ es un vector renglón. De esta manera la matriz $\widehat{V}^T$:

$$\widehat{V}^T = \begin{pmatrix} \widehat{V}_1^T \\ \widehat{V}_2^T \\ \vdots \\ \widehat{V}_p^T \end{pmatrix} = \begin{pmatrix} \widehat{V}_{11} & \widehat{V}_{12} & \dots & \widehat{V}_{1n} \\ \widehat{V}_{21} & \widehat{V}_{22} & \dots & \widehat{V}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{V}_{p1} & \widehat{V}_{p2} & \dots & \widehat{V}_{pn} \end{pmatrix}$$

Por lo que la traza que involucra a B tiene la siguiente forma:

$$Tr(V^T B V) = Tr(V^T Q \Lambda_B Q^T V)$$

$$Tr(V^T B V) = Tr(\widehat{V}^T \Lambda_B \widehat{V})$$

$$V^T B V = \begin{pmatrix} \widehat{V}_1^T \\ \widehat{V}_2^T \\ \vdots \\ \widehat{V}_p^T \end{pmatrix} \begin{pmatrix} \lambda_{B_1} & 0 & \dots & 0 \\ 0 & \lambda_{B_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{B_n} \end{pmatrix} \begin{pmatrix} \widehat{V}_1|\widehat{V}_2|\dots|\widehat{V}_p \end{pmatrix}$$

Desarrollando la multiplicación de matrices, y calculando la traza resulta en los siguientes sumandos:

$$\begin{aligned} Tr(V^T B V) = & \lambda_{B_1}\widehat{V}_{11}^2 + \lambda_{B_2}\widehat{V}_{12}^2 + \dots + \lambda_{B_n}\widehat{V}_{1n}^2 + \\ & \lambda_{B_1}\widehat{V}_{21}^2 + \lambda_{B_2}\widehat{V}_{22}^2 + \dots + \lambda_{B_n}\widehat{V}_{2n}^2 + \\ & \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ & \lambda_{B_1}\widehat{V}_{p1}^2 + \lambda_{B_2}\widehat{V}_{p2}^2 + \dots + \lambda_{B_n}\widehat{V}_{pn}^2. \end{aligned}$$

Es fácil de ver que la expresión de arriba tiene $p \times n$ sumandos, por lo que se puede expresar en términos de dos sumatorias. La primera de $j = 1, ..., p$ y la segunda de $i = 1, ...n$:

$$Tr(V^T BV) = \sum_{j=1}^{p} \sum_{i=1}^{n} \lambda_{B_i} \widehat{V}_{ji}^2$$

$$Tr(V^T BV) = \sum_{i=1}^{n} \lambda_{B_i} \sum_{j=1}^{p} \widehat{V}_{ji}^2 \tag{1.20}$$

De la última expresión se separa la sumatoria sobre i. De esta manera, para cada elemento i se tienen dos factores:

$$(i)\lambda_{B_i} \tag{1.21}$$

$$(ii) \sum_{j=1}^{p} \widehat{V}_{ji}^2 \tag{1.22}$$

La idea para que $Tr(V^T BV)$ sea positivo, es que al menos uno de los sumandos sea positivo. Si (1.21) y (1.22) son ambos distintos de cero para al menos una i, entonces se cumple esta condición. Esta idea está expresada en el Lema 1.1.

Lema 1.1. *Sea B positiva semidefinida y $V \in \mathbb{R}^{n \times p}$. Si B tiene a lo más $p-1$ eigenvalores iguales a 0, entonces $Tr(V^T BV) = Tr(\widehat{V}^T \Lambda_B \widehat{V}) \neq 0$ para cualquier matriz ortogonal V.*

Demostración. Sea $\widehat{V} = [\widehat{V}_1|...|\widehat{V}_p]$ tal que $\widehat{V}\widehat{V} = V^T QQ^T V = V^T I_n V = I_p$. De esta manera se puede construir una matriz $\widehat{V}' \in \mathbb{R}^{p \times p}$ seleccionando p de los n renglones de $\widehat{V}$ tal que $\widehat{V}'$ sea no singular. $\widehat{V}'$ tiene la propiedad de no contener eigenvalores iguales a 0; como consecuencia, sus renglones y columnas no contienen al vector $\widehat{0}$. Al no contenerlo,se sabe que al menos hay p renglones de $\widehat{V}$ tal que $\sum_{j=1}^{p} \widehat{V}_{ji}^2 \neq 0$ para cada uno de ellos. Por otra parte en el lema se supone que la matriz B tiene a lo más $p-1$ eigenvalores iguales 0 por lo que al menos un elemento de la sumatoria es distinto de cero. □

Analizando a mayor profundidad el resultado anterior, se sabe que hay $n-p+1$ eigenvalores de B positivos ($\lambda_{B_i} \neq 0$) y p renglones de $\widehat{V}$ que tienen norma

distinta de cero. Al calcular la sumatoria (1.20), se tiene que al menos una combinación de λ_{B_i} y uno de los p renglones cumplen que su multiplicación tiene signo positivo. Para ejemplificar esta situación sean C_i con $i = 1, ..., n-p+1$ los eigenvalores de B y K_j con $j = 1, ..., p$ la norma de los renglones de $\widehat{V}$ que son distintos de 0.

i	λ_{B_i}	$\sum_{j=1}^{p} \widehat{V}_{ji}^2$	$\lambda_{B_i} \sum_{j=1}^{p} \widehat{V}_{ji}^2$
1	C_1	0	0
2	C_2	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-p$	C_{n-p}	0	0
$n-p+1$	C_{n-p+1}	K_p	$C_{n-p+1}K_p$
$n-p+2$	0	K_{p-1}	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	0	K_2	0
n	0	K_1	0

Con esta combinación se tiene que al menos hay un sumando de $\sum_{i=1}^{n} \lambda_{B_i} \sum_{j=1}^{p} \widehat{V}_{ji}^2$ distinto de cero, por lo que $Tr(V^T BV) \neq 0$. Bajo estas condiciones se garantiza que el denominador sea mayor a 0, solo falta asegurarse que el numerador sea menor a infinito.

Lema 1.2. *Sea* $U_p = \{V \in \mathbb{R}^{n \times p} | V^T V = I_p\}$ *un conjunto compacto con* $V = (v_1, v_2, ..., v_p)$

Demostración. Se tiene que U_p es un conjunto cerrado porque contiene a todos sus puntos límite; por otro lado, U_p también es acotado bajo la norma 2 y la norma de Frobenius:

Tomando la norma-2 y la norma de Frobenius de V:

$$\begin{aligned}
||V||_2 =& Max\{||V_x||_2 \quad | \quad ||x||_2 = 1\} \\
=& ||V_x||_2^2 \\
=& (Vx)^T(Vx) \\
=& x^T V^T V x \\
=& x^T x = 1 \\
||V||_F =& \sum_F^p ||v_i|| = p
\end{aligned}$$

Entonces se tiene U_p es cerrado y acotado, por lo que U_p es compacto. □

Con este resultado se tiene que $Tr(V^T AV)$ toma un valor finito ya que todas sus entradas son finitas.

Lema 1.3. *Sean A y B dos matrices simétricas tales que B es positiva semi-definida con rango mayor que $n - p$; es decir, que tenga al menos $n - p + 1$ eigenvalores distintos de cero. Entonces el cociente* (1.9) *admite un máximo con valor ρ^* [9].*

Demostración. Tomando el resultado del lema 1.1 se tiene que $Tr(V^T BV) \neq 0$; por otra parte, $V \in U_p$ que es un conjunto compacto. Con estas dos observaciones, el valor de (1.9) es distinto de infinito. Entonces el cociente (1.9) admite un máximo con valor ρ^* y que tiene como argumento V^{**}. □

1.3.4. Equivalencia con un problema escalar

Valor en el óptimo. Del lema 1.3 se sabe que existe una matriz $V^{**} \in U_p$ tal que (1.9) alcanza el valor máximo ρ^*. Expresando esta idea se tiene que:

$$\frac{Tr(V^{T**} AV^{**})}{Tr(V^{T**} BV^{**})} = \rho^* \tag{1.23}$$

Entonces para cualquier otra matriz $V \in U_p$:

$$\frac{Tr(V^T AV)}{Tr(V^T BV)} \leq \rho^* \tag{1.24}$$

Como la traza es un operador lineal y por la propiedad distributiva de las matrices entonces (1.24) es equivalente a:

$$Tr(V^T AV) - \rho^* Tr(V^T BV) \leq 0$$

$$Tr(V^T AV - \rho^* V^T BV) \leq 0$$

$$Tr(V^T (A - \rho^* B)V) \leq 0 \tag{1.25}$$

y el resultado equivalente para (1.23):

$$Tr(V^{T**}(A - \rho^* B)V^{**}) = 0 \tag{1.26}$$

Para facilitar la lectura, de aquí en adelante se define la función $G(\rho) = A - \rho B$. Maximizar el lado izquierdo de la desigualdad (1.25) sujeto a $V^T V = I$ es equivalente a maximizar un problema de eigenvalores generalizado. Usando lo establecido en el apéndice A, se sabe que el valor máximo de este problema dado ρ^*:

$$\max_{\substack{V \in \mathbb{R}^{n \times p} \\ V^T V = I}} Tr(V^T G(\rho^*)V) = \lambda_{G(\rho^*)_1} + \lambda_{G(\rho^*)_2} + ... + \lambda_{G(\rho^*)_p} \tag{1.27}$$

Con $\lambda_{G(\rho^*)_1} \geq \lambda_{G(\rho^*)_2} \geq ... \geq \lambda_{G(\rho^*)_p}$ los p eigenvalores más grandes de $G(\rho^*)$. De esta manera el valor óptimo de (1.27) es simplemente la suma de los p eigenvalores más grandes de esta matriz, y V^{**} el conjunto de correspondientes eigenvectores. Para obtener este valor y la matriz, el primer paso es encontrar a ρ^*, ya que teniéndolo es inmediato calcular V^{**}. Dada esta premisa, se puede ver que el problema a resolver se reduce a buscar el valor óptimo de ρ. Para esto se define la función $f(\rho)$ sobre todo $\mathbb{R}$, tal que $f(\rho)$ es continua sobre su argumento ρ:

$$f(\rho) = \max_{V^T V = I} Tr(V^T (G(\rho))V) \tag{1.28}$$

Es conveniente examinar $f(\rho)$ con dos objetivos, el primero es estimar la dificultad de calcular el valor de $f(\rho)$ y el segundo es encontrar la maximización

adecuada para obtener ρ^*. Respecto al primer punto, la manera de calcular $f(\rho)$ en cada punto es equivalente a (1.27), pero en lugar de usar los eigenvalores de $G(\rho^*)$ se usan los de $G(\rho)$. En particular se llamará $V(\rho)^*$ al argumento que resuelve (1.28) [2]. Sean $\lambda_{G(\rho)_1} \geq \lambda_{G(\rho)_2} \geq ... \geq \lambda_{G(\rho)_n}$ los n eigenvalores de $G(\rho)$. Con esta notación $f(\rho)$ toma el valor de:

$$f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2} + ... + \lambda_{G(\rho)_p} \tag{1.29}$$

Para el segundo punto, la idea es iterar hasta obtener el valor de ρ^*. Por esto, es conveniente analizar como se comporta la función con respecto a su argumento. A continuación se presentan dos propiedades de $f(\rho)$. Para demostrarlas, primero se enuncia el teorema 8.1.5 de [6].

Teorema 1.1. *Sean X y $X + E$ matrices simétricas $n \times n$, y λ_{X_k}, λ_{E_k} los k-ésimos eigenvalores más grandes de X y E respectivamente. De esta manera λ_{X_k} es el k-ésimo eigenvalor más grande de X y λ_{E_1} el más grande de E. Con estas definiciones se cumple lo siguiente:*

$$\lambda_{X_k} + \lambda_{E_n} \leq \lambda_{(X+E)_k} \leq \lambda_{X_k} + \lambda_{E_1} \tag{1.30}$$

Se procede a enunciar este lema acerca de la función $f(\rho)$

Lema 1.4. *La función $f(\rho) = \max_{V^T V = I} Tr(V^T(A - \rho B)V)$ cumple las siguientes dos propiedades:*

(1) f es una función no creciente de ρ
(2) $f(\rho) = 0$ si y solo si $\rho = \rho^*$

Para probar la parte (1) del lema 1.4 se comparan los valores de $f(\rho)$ para ρ_1 y ρ_2 con $\rho_2 \geq \rho_1$. Como se desea demostrar que f es una función no creciente de ρ, se busca que $f(\rho_2) \leq f(\rho_1)$. Se definen las matrices $Y = X + E$ y $E = Y - X$, para después restarles λ_{X_k}, entonces (1.29) se puede escribir como:

$$\lambda_{(X)_k} + \lambda_{(Y-X)_n} \leq \lambda_{(Y)_k} \leq \lambda_{(X)_k} + \lambda_{(Y-X)_1}$$

[2] Se utilizó la nomenclatura de $V(\rho^*)$ porque estos eigenvectores dependen del valor de ρ en la matriz $A - \rho B$.

$$\lambda_{(Y-X)_n} \leq \lambda_{(Y)_k} - \lambda_{(X)_k} \leq \lambda_{(Y-X)_1}$$

La expresión en el centro de estas desigualdades es la resta del eigenvalor k-ésimo de la matriz X y Y, por lo que si se realiza la suma de $k = 1$ a $k = p$ se tiene lo siguiente:

$$p\lambda_{(Y-X)_n} \leq \sum_{k=1}^{p} \lambda_{(Y)_k} - \sum_{k=1}^{p} \lambda_{(X)_k} \leq p\lambda_{(Y-X)_1} \tag{1.31}$$

Definiendo $X = A - \rho_2 B$ y $Y = A - \rho_1 B$, entonces (1.31) toma la siguiente forma:

$$p\lambda_{(B(\rho_2-\rho_1))_n} \leq \sum_{k=1}^{p} \lambda_{(A-\rho_1 B)_k} - \sum_{k=1}^{p} \lambda_{(A-\rho_2 B)_k} \leq p\lambda_{(B(\rho_2-\rho_1))_1} \tag{1.32}$$

Retomando el resultado (1.29), y sustituyéndolo en (1.32) la desigualdad queda de la siguiente forma:

$$p\lambda_{(B(\rho_2-\rho_1))_n} \leq f(\rho_1) - f(\rho_2) \leq p\lambda_{(B(\rho_2-\rho_1))_1} \tag{1.33}$$

En la parte izquierda de la desigualdad anterior, se puede a determinar el signo que toman los eigenvalores $\lambda_{(B(\rho_2-\rho_1))_n}$. Si $(\rho_2 - \rho_1) \geq 0$ entonces la matriz $(\rho_2 - \rho_1)B$ es positiva semidefinida; por lo tanto, todos sus eigenvalores son mayores o iguales a 0:

$$0 \leq p\lambda_{(B(\rho_2-\rho_1))_n} \leq f(\rho_1) - f(\rho_2) \leq p\lambda_{(B(\rho_2-\rho_1))_1}$$

$$0 \leq f(\rho_1) - f(\rho_2) \tag{1.34}$$

De esta manera $f(\rho_2) \leq f(\rho_1)$ cuando $\rho_2 \geq \rho_1$

Para probar la parte (2) del lema 1.4, se tiene que demostrar la condición suficiente y la condición necesaria. La demostración de la primera es inmediata, ya que cuando $\rho = \rho^*$, entonces por (1.26), $f(\rho) = 0$. Para la condición necesaria se usará (1.23) y la propiedad que la $Tr(V^T BV) > 0 \ \forall \ V \in U_p$. Se demostrará que, dado $f(\rho^*) = 0$, entonces:

$$(i) \quad f(\rho) < 0 \quad si \quad \rho > \rho^* \tag{1.35}$$

$$(ii) \quad f(\rho) > 0 \quad si \quad \rho < \rho^* \tag{1.36}$$

Caso (i) $\rho > \rho^*$

Se tiene que las siguientes desigualdades se cumplen:

$$\frac{Tr(V^T AV)}{Tr(V^T BV)} \leq \rho^* < \rho$$

Por lo que es equivalente a:

$$Tr(V^T(A - \rho B)V) < 0 \quad \forall V \in U_p$$

De esta manera $f(\rho) < 0$ cuando $\rho > \rho^*$.

(ii) $\rho < \rho^*$

Retomando el resultado (1.24) y suponiendo que $\rho^* > \rho$ entonces existe una V^* tal que:

$$\rho < \frac{Tr(V^{T*} AV^*)}{Tr(V^{T*} BV^*)} \leq \rho^*$$

$$Tr(V^{T*} AV^* - \rho V^{T*} BV^*) > 0 \quad \Longrightarrow \quad \frac{Tr(V^{T*} AV^*)}{Tr(V^{T*} BV^*)} > \rho$$

En particular:

$$\max_{V^T V} \frac{Tr(V^T AV)}{Tr(V^T BV)} > \rho$$

De esta manera $f(\rho) > 0$ cuando $\rho < \rho^*$.

Las ecuaciones (1.35) (1.36), junto con la continuidad de la funcion f, muestran que $f(\rho) = 0$ implica $\rho = \rho^*$ [9]. De esta manera el problema puede ser visto como encontrar la raíz de la función $f(\rho)$. La figura 1.3.4 muestra como es esta función.

Corolario 1.1. *La función* $f(\rho) = \text{máx}_{V^T V=I}\, Tr(V^T(A-\rho B)V)$ *cumple las siguientes condiciones:*

$$f(\rho) > 0 \quad \forall \quad \rho \in (-\text{ínf}, \rho^*)$$
$$f(\rho) < 0 \quad \forall \quad \rho \in (\rho^*, \text{ínf})$$

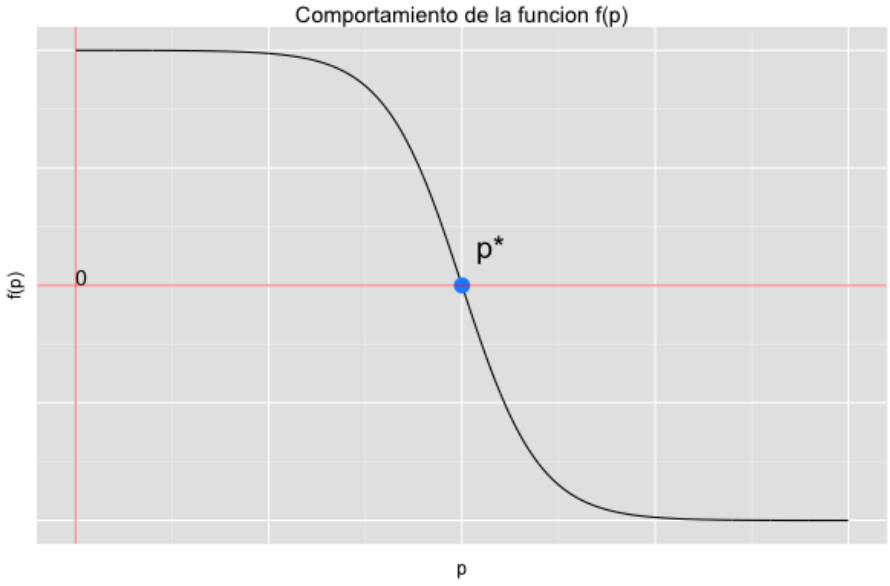


Figura 1.4: La función $f(\rho)$ es no creciente para toda ρ. El valor de $f(\rho) = \lambda_{G(\rho)1} + \lambda_{G(\rho)2} + ... + \lambda_{G(\rho)p}$. $f(\rho^*) = 0$

Ejemplo 1.1. *Para ejemplificar el lema 1.1 se muestran las matrices* $A, B \in \mathbb{R}^{3\times 3}$. *Para el valor de* $f(\rho)$ *se utiliza la propiedad (1.29):*

$$f(\rho) = \lambda_{G(\rho)1} + \lambda_{G(\rho)2} + ... + \lambda_{G(\rho)p}$$

con p *la dimensión a la que se va a proyectar.*

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 1.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

$$G(\rho) = A - \rho B = \begin{pmatrix} 4 - 1.5\rho & 0 & 0 \\ 0 & 6 - 2.5\rho & 0 \\ 0 & 0 & 8 - 5\rho \end{pmatrix}$$

Los eigenvalores de esta matriz son $4 - 1.5\rho$, $6 - 2.5\rho$ *y* $8 - 5\rho$. *Las funciones graficadas de estos eigenvalores se presentan en la figura 1.1.*

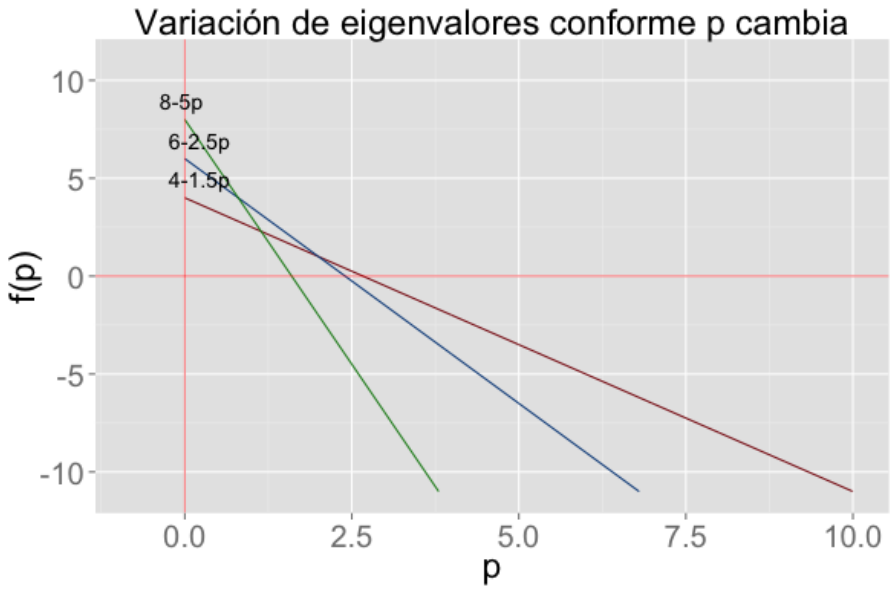


Figura 1.5: Cada línea representa como se comporta cada eigenvalor de $A - \rho B$ cuando se varía ρ.

Cuando se desea que el proyector sea de dimensión 1, entonces se tiene que $f(\rho)$ *es el eigenvalor más grande, cuando sea de dimensión 2, la suma de los dos más grandes y así respectivamente. El valor de* $f(\rho)$ *para estos tres casos está representado en la figura 1.1.*

1.3.5. Localización del óptimo

En la sección anterior se encontró que la maximización al cociente de trazas puede ser visto como el problema de encontrar la raíz de la función $f(\rho) = max_{V^TV=I} Tr(V^T(A - \rho B)V)$. Por esto, encontrar un intervalo (ρ_1, ρ_2) que contenga al valor óptimo ρ^* puede reducir el número de iteraciones del método. Usando el lema 1.4 se sabe que f es una función no creciente de ρ. Por esta razón si se encuentra una ρ_1 y ρ_2 tal que $f(\rho_1) \geq 0$ y $f(\rho_2) \leq 0$ y con la propiedad de continuidad de la funcion $f(\rho)$ entonces se encontró un intervalo que contiene a ρ^*.

En esta tesis se dan cotas para el valor de ρ^*, la primera en función los eigenvalores de una transformación de $B - \rho A$ y la segunda en función de los eigenvalores de B y A [9]. La demostración de cada una requiere del conocimiento del concepto de inercia y del Teorema de la Inercia de Sylvester. Por este motivo se presentan a continuación: [3]

Definición 1.1. *La inercia de una matriz simétrica A es la tripleta de enteros no negativos (m, z, p) donde m, z y p son respectivamente el número de eigenvalores negativos, cero y positivos de A [6].*

Teorema 1.2. *Sea $A \in \mathbb{R}^{n\times n}$ una matriz simétrica y $Z \in \mathbb{R}^{n\times n}$ no singular. Entonces A y Z^TAZ tienen la misma inercia [6].*

Proposición 1.3. *La raíz ρ^* de $f(\rho)$ está localizada en el intervalo (λ_p, λ_1) donde λ_p es el p-ésimo eigenvalor más grande de $Z^T(A - \rho B)Z$.*

Demostración. Sea Z la matriz que diagonaliza a $A - \rho B$ de manera que[4]:

$$\begin{aligned} Z^TAZ &= \Lambda \\ Z^TBZ &= I \end{aligned} \tag{1.37}$$

Con Λ una matriz diagonal y $\mu_1, \ldots, \mu_n$ sus respectivos eigenvalores e I la identidad de tamaño n. Entonces por el teorema 1.2 se sabe que $A - \rho B$ y $Z^T(A - \rho B)Z = \Lambda - \rho I$ tienen el mismo número de eigenvalores positivos, negativos y cero. Por lo tanto, la matriz en cuestión es de la siguiente forma:

[3] La demostración de este teorema puede ser encontrada en [6].

[4] El cálculo de esta matriz puede obtenerse en el algoritmo 8.7.1 de [6]

$$\Lambda - \rho I = \begin{pmatrix} \mu_1 & 0 & \dots & 0 \\ 0 & \mu_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mu_n \end{pmatrix} - \rho \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \tag{1.38}$$

Como la matriz V de (1.9) es de tamaño $n \times p$, solo nos interesa saber el signo de los p eigenvalores más grandes. Tomando $\rho = \mu_p$ entonces los elementos de la diagonal de la matriz $\Lambda - \rho I$ son de la forma:

$$\begin{aligned} \mu_i - \mu_p \geq 0 \quad para \quad i \geq p \\ \mu_i - \mu_p \leq 0 \quad para \quad i \leq p \end{aligned} \tag{1.39}$$

Los primeros p elementos tienen la propiedad de ser no negativos, ya que $\lambda_{(A-\rho B)_1} \geq \lambda_{(A-\rho B)_2} \geq ... \geq \lambda_{(A-\rho B)_p}$. Usando el teorema 1.2 se sabe que los primeros p eigenvalores de $A - \rho B$ también son no negativos. Por ende la suma de ellos es mayor o igual que cero.

Por otro lado si se toma $\rho = \mu_1$. Entonces los elementos de la diagonal de la matriz (1.38) son de la forma:

$$\mu_i - \mu_1 \leq 0 \quad \forall \quad i \tag{1.40}$$

Con $i = 1, ..., p$, cada uno de los elementos de la diagonal tiene la propiedad de ser no positivo por el mismo argumento que el caso pasado. Por lo tanto los p eigenvalores más grandes de $\Lambda - \rho I$ y de $A - \rho B$ son no positivos, por lo que su suma es menor o igual que cero:

$$\rho = \mu_p \Rightarrow \sum_{i=1}^{p} (\mu_i - \mu_p) \geq 0 \Rightarrow f(\rho) \geq 0 \tag{1.41}$$

$$\rho = \mu_1 \Rightarrow \sum_{i=1}^{p} (\mu_i - \mu_1) \leq 0 \Rightarrow f(\rho) \leq 0 \tag{1.42}$$

□

Ejemplo 1.2. *Tomando las matrices A,B iguales que en el ejercicio 1.1, se puede encontrar fácilmente a la matriz Z:*

$$Z = \begin{pmatrix} \sqrt{(\frac{1}{1.5})} & 0 & 0 \\ 0 & \sqrt{(\frac{1}{2.5})} & 0 \\ 0 & 0 & \sqrt{(\frac{1}{5})} \end{pmatrix}$$

Con la matriz Z definida de esta manera, $Z^T AZ = \Lambda$ y $Z^T BZ = I$ toman la siguiente forma:

$$\Lambda - \rho I = \begin{pmatrix} \frac{4}{1.5} & 0 & 0 \\ 0 & \frac{6}{2.5} & 0 \\ 0 & 0 & \frac{8}{5} \end{pmatrix} - \rho \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Entonces se puede encontrar un intervalo tal que $\rho^* \in [\rho_1, \rho_2]$:

$$\rho_1 = \mu_p$$
$$\rho_2 = \mu_1$$

Conforme el tamaño de la dimensión a proyectar cambia, las cotas son las siguientes:

$$\begin{aligned} p = 1 \Rightarrow \quad & \rho_1 = \frac{4}{1.5} \quad y \quad \rho_2 = \frac{4}{1.5} \\ p = 2 \Rightarrow \quad & \rho_1 = \frac{4}{1.5} \quad y \quad \rho_2 = \frac{6}{2.5} \\ p = 3 \Rightarrow \quad & \rho_1 = \frac{4}{1.5} \quad y \quad \rho_2 = \frac{8}{5} \end{aligned}$$

Estas cotas se pueden var más facil en la figura 1.2,donde se observa $f(\rho)$ con respecto a p:

Otro intervalo que se ha desarrollado tiene que ver con directamente con los eigenvalores de A y B en lugar de los obtenidos por la matriz $A - \rho B$:

Proposición 1.4. *Sea B positiva definida, entonces la raíz ρ^* de $f(\rho)$ es tal que [9]:*

$$\frac{\sum_{i=1}^{p}\lambda_{A_i}}{\sum_{i=1}^{p}\lambda_{B_i}} \leq \rho^* \leq \frac{\sum_{i=1}^{p}\lambda_{(A)_i}}{\sum_{i=1}^{p}\lambda_{(B)_{n-i+1}}}$$

con λ_{A_i} y λ_{B_i} el i-ésimo eigenvalor más grande de la matriz A y B respectivamente [9].

Demostración. Se tiene la propiedad que para una p dada:

$$\max_{V^TV=I} Tr(V^TAV) = Tr(V^{T*}AV^*) = \sum_{i=1}^{p}\lambda_{A_i} \tag{1.43}$$

Con λ_{A_i} los eigenvalores de A. Como esta V^* maximiza la traza sobre A, entonces no necesariamente maximiza la de B. Al sustituirla en $Tr(V^TBV)$ se tiene que:

$$Tr(V^{T*}BV^*) \leq \sum_{i=1}^{p}\lambda_{B_i} \tag{1.44}$$

Con λ_{B_i} los eigenvalores de B. Al hacer el cociente de (1.43) y (1.44) Se puede acotar inferiormente a ρ^*:

$$\frac{\sum_{i=1}^{p}\lambda_{A_i}}{\sum_{i=1}^{p}\lambda_{B_i}} \leq \frac{Tr(V^{T*}AV^*)}{Tr(V^{T*}BV^*)} \leq \max_{V^TV}\frac{Tr(V^TAV)}{Tr(V^TBV)} = \rho^*$$

Ahora falta acotarlo superiormente. Usando las siguientes propiedades que son derivadas de (1.27):

$$Tr(V^TAV) \leq \sum_{i=1}^{p}\lambda_{A_i} \tag{1.45}$$

$$Tr(V^TBV) \geq \sum_{i=1}^{p}\lambda_{B_{(n-i+1)}} \tag{1.46}$$

La expresión 1.46 es la suma de los p eigenvalores más chicos de B. Dividiendo 1.45 entre 1.46 se tiene que para cualquier matriz ortogonal V:

$$\frac{Tr(V^T AV)}{Tr(V^T BV)} \leq \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_{(n-i+1)}}} \tag{1.47}$$

En particular si se toma $V = V^{**}$ (La matriz con la que se alcanza ρ^*):

$$\rho^* = \frac{Tr(V^{T**} AV^{**})}{Tr(V^{T**} BV^{**})} \leq \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_{(n-i+1)}}} \tag{1.48}$$

□

Ejemplo 1.3. *Para ejemplificar esta cota se usará las matrices A y B de los dos ejemplos anteriores y se muestra en la figura 1.3.5*

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 1.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

(i) Para $p = 1$ la cota es la siguiente:

$$\lambda_{A_1} = 8 \qquad \lambda_{B_1} = 5 \qquad \lambda_{B_3} = 1.5$$

$$\rho_1 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_i}} = \frac{8}{5} \qquad \rho_2 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_{n-i+1}}} = \frac{8}{1.5}$$

(ii) Para $p = 2$ la cota es la siguiente:

$$\sum_{i=1}^{2} \lambda_{A_1} = 14 \qquad \sum_{i=1}^{2} \lambda_{B_1} = \quad 7.5 \qquad \sum_{i=1}^{2} \lambda_{B_{3-i+1}} = 4$$

$$\rho_1 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{(B_i)}} = \frac{14}{7.5} \qquad \rho_2 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_{n-i+1}}} = \frac{14}{4}$$

(iii) Para $p = 3$ la cota es la siguiente:

$$\sum_{i=1}^{3} \lambda_{A_1} = 18 \qquad \sum_{i=1}^{3} \lambda_{B_1} = 9 \qquad \sum_{i=1}^{3} \lambda_{B_{n-i+1}} = 9$$

$$\rho_1 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_i}} = \frac{18}{9} \qquad \rho_2 = \frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_{n-i+1}}} = \frac{18}{9}$$

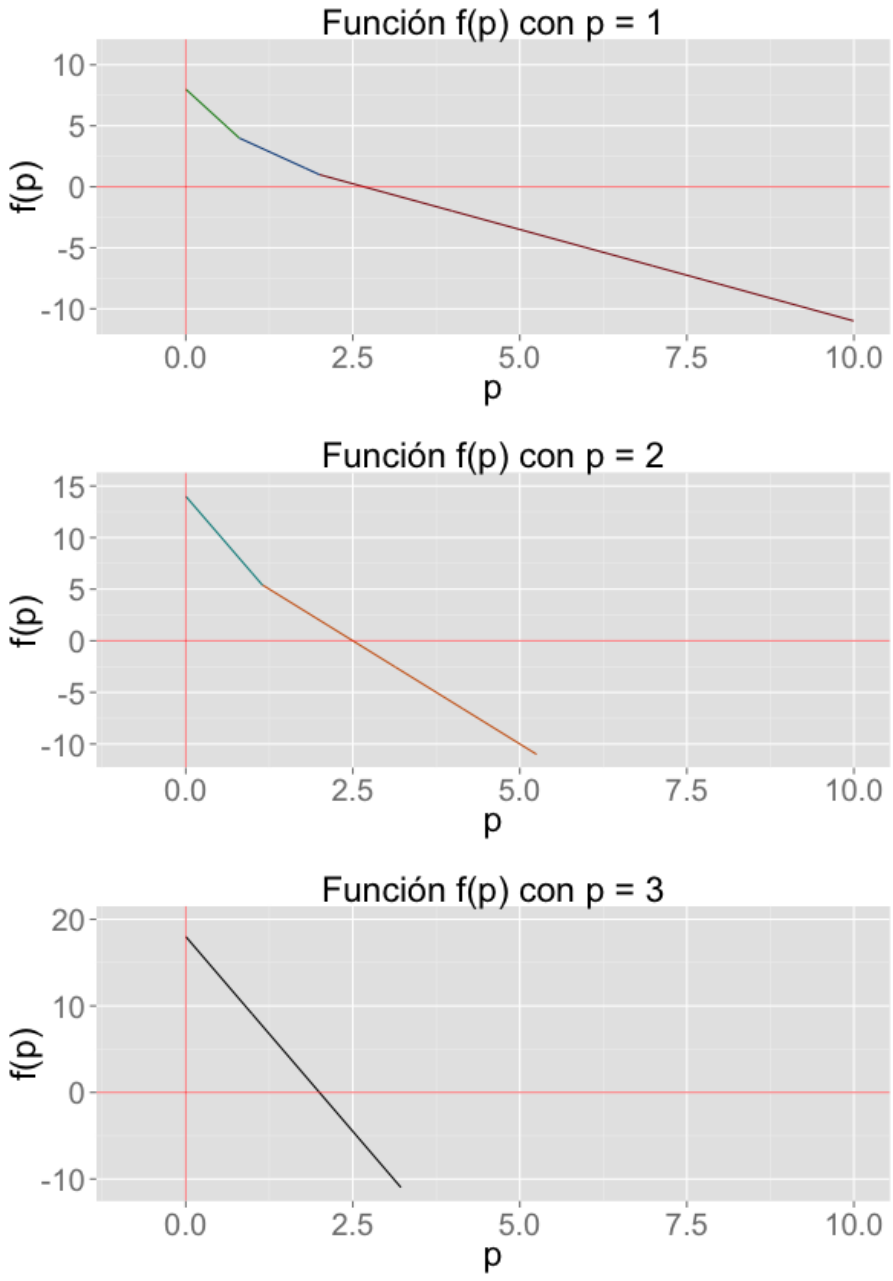


Figura 1.6: La figura superior representa a $f(\rho) = \lambda_{G(\rho)_1}$, la de enmedio $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2}$ y la de abajo $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2} + \lambda_{G(\rho)_3}$.

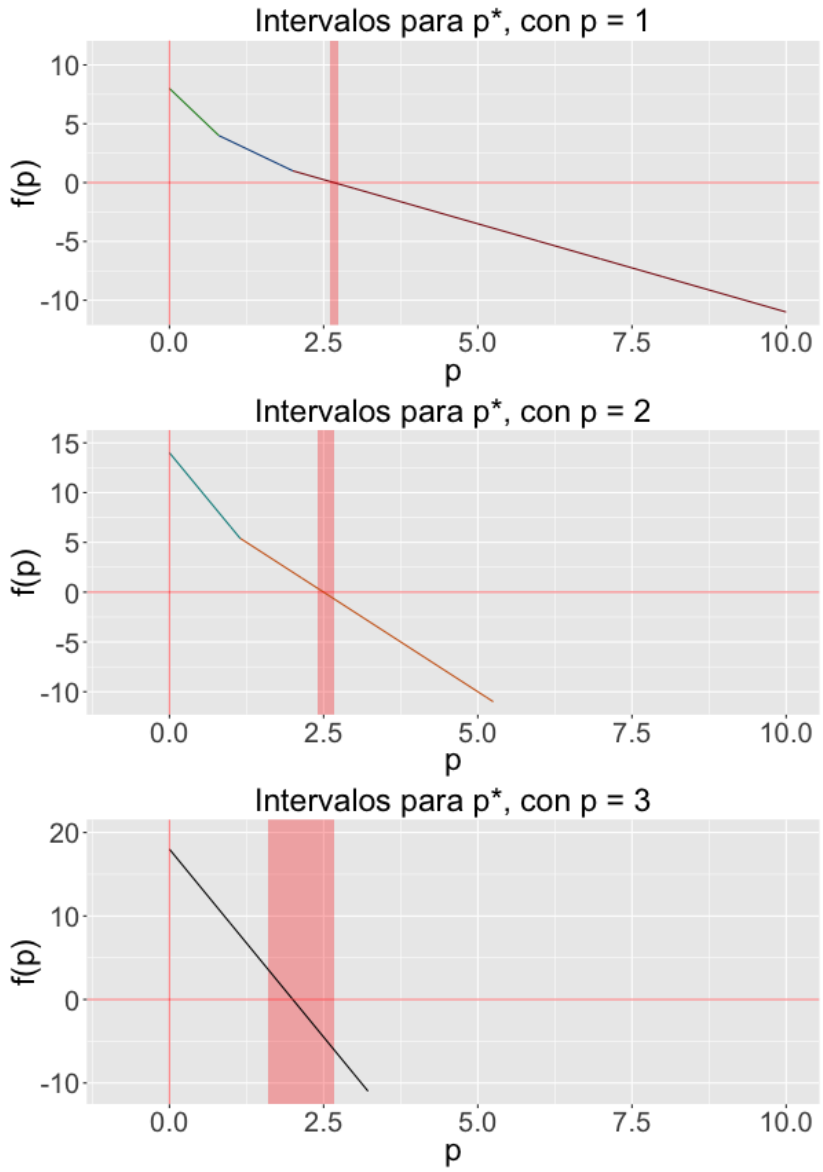


Figura 1.7: La figura superior representa intervalos para ρ^* cuando $f(\rho) = \lambda_{G(\rho)_1}$, la de enmedio cuando $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2}$ y la de abajo cuando $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2} + \lambda_{G(\rho)_3}$.

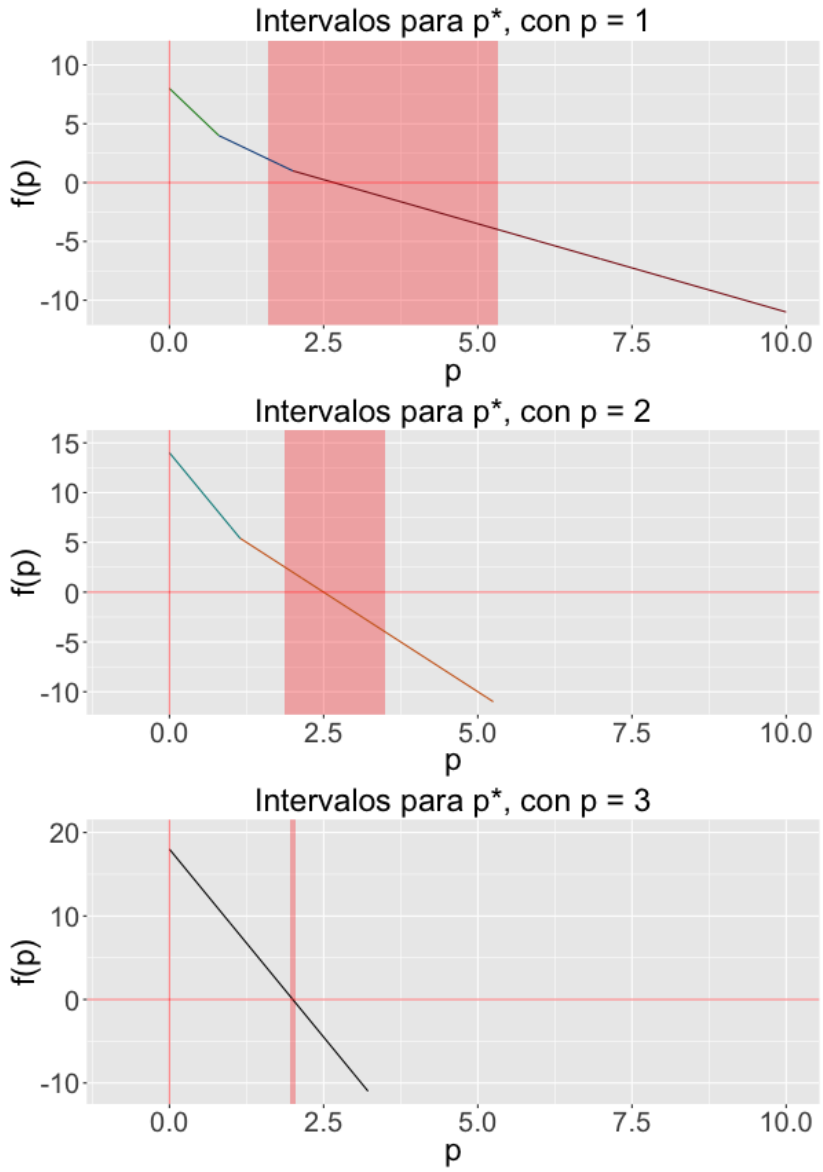


Figura 1.8: La figura superior representa intervalos para ρ^* cuando $f(\rho) = \lambda_{G(\rho)_1}$, la de enmedio cuando $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2}$ y la de abajo cuando $f(\rho) = \lambda_{G(\rho)_1} + \lambda_{G(\rho)_2} + \lambda_{G(\rho)_3}$.

Capítulo 2

El Método Newton-Lanczos

En el capítulo anterior se propuso una función no creciente $f(\rho)$ cuya raíz resulta ser la solución óptima para el problema del Discriminante Lineal de Fisher:

$$f(\rho) = \max_{V^T V = I} Tr(V^T (S_E - \rho S_I) V) \tag{2.1}$$

El algoritmo propuesto para encontrar la solución recibe el nombre de Newton-Lanczos [9]. Para entenderlo a profundidad, se explicarán brevemente los métodos de Lanczos que tridiagonalizan una matriz simétrica, para después calcular eficientemente los primeros (últimos) eigenvalores. Después, será implementado junto al método iterativo de Newton, que calcula el nuevo valor de ρ_n para cada paso. Para esto, se requiere el cómputo de la derivada de $f(\rho)$, por lo que se desarrollará su forma analítica. Finalizando, se proporcionarán las condiciones necesarias de optimalidad.

La primera división de los métodos para calcular eigenvectores y eigenvalores que propone J. Demmel [2] depende si la matriz es simétrica o no lo es. Después hace una sub-clasificación dependiendo si el método es iterativo o directo. Para este texto solo se calcularán los eigenvalores de matrices simétricas, por lo que el procedimiento a seguir sera el siguiente:

- Tridiagonalizar la matriz simétrica por un método iterativo
- Encontrar los eigenvalores por medio de la iteración tridiagonal QR (LA-

PACK DSTEVD)

2.1. Métodos de Lanczos

Antes de presentar los métodos de Lanczos, se dará una breve introducción acerca del costo computacional del algoritmo QR y la importancia de tridiagonalizar la matriz [2]. Sea $A \in \mathbb{R}^{n \times n}$, entonces su descomposición QR toma $O(n^3)$ flops. Suponiendo el mejor escenario en el que cada eigenvalor se encuentra con una iteración, tomaría $O(n^4)$ flops para calcular todos los eigenvalores de una matriz. Por otra parte, al tridiagonalizar la matriz A se reduce el costo computacional de la descomposición QR a $O(n^2)$ flops. De esta manera, tomaría $O(n^3)$ flops encontrar todos los eigenvalores. Una descripción más detallada del costo computacional de los algoritmos puede encontrarse en [2].

El primer paso, la tridiagonalización, ha sido muy estudiado y existen algoritmos especializados para distintos tipos de matrices simétricas. Si la matriz es de gran dimensión y rala, entonces se recomienda usar el método de Lanczos [6]. En otro caso, existen las transformaciones Householder y las rotaciones de Givens [6]. El método de Lanczos realiza tridiagonalizaciones parciales de la matriz original A, donde cada una es de tamaño $p \times p$ con $p \leq n$. Un aspecto interesante es que las matrices parciales van aproximando los eigenvectores extremos antes que la tridiagonalización esté completa. Por este motivo, el algoritmo es usado cuando se requieren solo algunos de los eigenvalores. Con respecto al costo computacional, es del orden $O(n^3)$, por lo que el algoritmo completo se mantiene en el mismo orden.

Lanczos en aritmética exacta tiene muchas ventajas computacionales y converge rápidamente a los eigenvalores reales, pero con aritmética inexacta es difícil usarlo en la práctica [6]. El problema que se presenta es que los eigenvectores van perdiendo la ortogonalidad entre ellos conforme la dimensionalidad y las iteraciones incrementan. Los textos [2] [6] incluyen algoritmos para solucionar este problema.

En la siguiente subsección se presentará el algoritmo original de Lanczos; se ejemplificará como los eigenvalores de la matriz tridiagonal convergen a los originales y se presenta el problema de *ghost eigenvalues* [2], que son los eigenvalores

que surgen al perder la ortogonalidad en aritmética inexacta.

2.1.1. Algoritmo de Lanczos

El algoritmo de Lanczos busca calcular los elementos de esta matriz tridiagonal directamente. Definiendo $Q^T AQ = T$, con $Q = [q_{(1)} \mid q_{(2)} \mid ... \mid q_{(n)}]$ ortogonal, y T_n tridiagonal igual a:

$$T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{n-1} & \\ & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & & & & \beta_n & \alpha_n \end{pmatrix}, \tag{2.2}$$

Como $q_{(1)}$ es una columna de Q, entonces $q_1^T = [q_{11},\ q_{21},\ ...,\ q_{n1}]$. De esta manera, se tiene que $AQ = QT$. Descomponiendo la multiplicación $AQ = [Aq_{(1)} \mid Aq_{(2)} \mid ... \mid Aq_{(n)}]$ e igualándola a cada columna de QT, se tiene que $Aq_{(1)}$ [6]:

$$\begin{aligned} Aq_{(1)} = \ & q_{(1)(1)}\alpha_{(1)} + q_{(1)(2)}\beta_{(2)} + \\ & q_{(2)(1)}\alpha_{(1)} + q_{(2)(2)}\beta_{(2)} + \\ & \vdots \qquad\qquad \vdots \\ & q_{(n)(1)}\alpha_{(1)} + q_{(n)(2)}\beta_{(2)} \\ Aq_{(1)} = \ & \alpha_{(1)}q_{(1)} + \beta_{(2)}q_{(2)} \end{aligned}$$

Ahora, para $i = 2, ..., n-1$

$$\begin{aligned} Aq_{(i)} = \ & q_{(i-1)(i-1)}\beta_{(i)} + q_{(i-1)(i)}\alpha_{(i)} + q_{(i-1)(i+1)}\beta_{(i+1)} + \\ & q_{(i)(i-1)}\beta_{(i)} + q_{(i)(i)}\alpha_{(i)} + q_{(i)(i+1)}\beta_{(i+1)} + \\ & \vdots \qquad\qquad\qquad\qquad \vdots \\ & q_{(n)(i-1)}\beta_{(i)} + q_{(n)(i)}\alpha_{(i)} + q_{(n)(i+1)}\beta_{(i+1)} \\ Aq_{(i)} = \ & \beta_{(i)}q_{(i-1)} + \alpha_{(i)}q_{(i)} + \beta_{(i+1)}q_{(i+1)} \end{aligned}$$

Por último, para Aq_n:

$$\begin{aligned} Aq_{(n)} = \quad & q_{(1)(n)}\alpha_{(n)} \quad + \quad q_{(1)(n-1)}\beta_{(n)} \quad + \\ & q_{(2)(n)}\alpha_{(n)} \quad + \quad q_{(2)(n-1)}\beta_{(n)} \quad + \\ & \vdots \qquad\qquad\qquad \vdots \\ & q_{(n)(n)}\alpha_{(n)} \quad + \quad q_{(n)(n-1)}\beta_{(n)} \\ Aq_{(n)} = \quad & \alpha_{(n)}q_{(n)} \qquad + \quad \beta_{(n)}q_{(n-1)} \end{aligned}$$

Si se define $q_{(0)} = 0$, entonces se puede resumir el paso como:

$$Aq_{(i)} = \beta_{(i)}q_{(i-1)} + \alpha_{(i)}q_{(i)} + \beta_{(i+1)}q_{(i+1)} \tag{2.3}$$

para $i = 1, ...n-1$. Multiplicando esta expresión por $q_{(i)}^T$, y usando el supuesto de ortogonalidad, entonces $q_{(i)}^T q_{(j)} = 0$ con $i \neq j$. De esta manera resulta la siguiente expresión:

$$q_{(i)}^T Aq_{(i)} = q_{(i)}^T \alpha_{(i)} q_{(i)} \quad = \alpha_{(i)} \tag{2.4}$$

Por otra parte, despejando $\beta_{(i+1)}q_{(i+1)}$ de 2.3, se tiene que

$$\begin{aligned} \beta_{(i+1)}q_{(i+1)} =& Aq_{(i)} - \beta_{(i)}q_{(i-1)} - \alpha_{(i)}q_{(i)} \\ =& (A - \alpha_{(i)}I)q_{(i)} - \beta_{(i)}q_{(i-1)} = r_{(i)} \end{aligned} \tag{2.5}$$

Con la ecuación 2.5, $q_{(i+1)} = \frac{r_{(i)}}{\beta_{(i+1)}}$. Calculando la norma de $r_{(i)}$, se tiene que

$$\begin{aligned} ||r_{(i)}||_2 =& |\beta_{(i+1)}|||q_{(i+1)}||_2 \\ =& |\beta_{(i+1)}| \end{aligned} \tag{2.6}$$

Cuando $r_k = 0$ entonces la iteración se detiene. [6]

Implementación en aritmética exacta

Sea $A \in \mathbb{R}^{n\times n}$ una matriz simétrica y $q_i \in \mathbb{R}^n$. Entonces el algoritmo que se presenta a continuación produce una matriz $T_k \in \mathbb{R}^{k\times k}$ tridiagonal tal que los eigenvalores de T_k convergen a los de la matriz original A [6].

$q_0 \leftarrow 0$;
$r_0 \leftarrow$ vector aleatorio;
$\beta_1 \leftarrow ||r_0||_2$;
$q_1 \leftarrow r_0/\beta_1$;
$\alpha_1 \leftarrow q_1^T A q_1$;
$eps = 0.0000001$;
$k = 1$;
while *($\beta_k > eps$)* **do**
    $r_k \leftarrow Aq_k - \alpha_k q_k - \beta_k q_{k-1}$;
    $\beta_{k+1} \leftarrow ||r_{k-1}||_2$;
    $q_{k+1} \leftarrow r_{k+1}/\beta k + 1$;
    $\alpha_{k+1} \leftarrow q_{k+1}^T A q_{k+1}$;
    $k \leftarrow k + 1$;
end

Algorithm 1: Algoritmo de Lanczos

Para hacer frente a la pérdida de ortogonalidad entre los vectores, se han creado distintos métodos, los cuales se basan en la reortogonalización de la base de vectores. [2] Esta puede realizarse en $O(n^2)$ operaciones, por lo que no afecta el orden de $O(n^3)$ flops.

2.2. Derivada de $f(\rho)$

La fórmula analítica de la derivada de $f(\rho)$ se puede obtener con cálculo multivariado. Para encontrarla, sea $V(\rho) \in \mathbb{R}^{n\times p}$ una función diferenciable con respecto a ρ. Esta cumple la característica de ser una matriz ortogonal con columnas:

$$V(\rho) = (v_1(\rho) \mid v_2(\rho) \mid ... \mid v_p(\rho))$$

Antes de calcular la derivada de $f(\rho)$, es conveniente examinar la derivada de $V(\rho)^T V(\rho)$ con respecto a ρ. Sea $V(\rho)$ una matriz ortogonal; es decir, que cumpla $V^T(\rho)V(\rho) = I$, entonces [9]:

$$\frac{d}{d\rho}V(\rho)^T V(\rho) = \left(\frac{d}{d\rho}V(\rho)^T\right) V(\rho) + V(\rho)^T \left(\frac{d}{d\rho}V(\rho)\right)$$

La derivada de $V(\rho)$ no se conoce explícitamente, pero se puede derivar componente a componente. De esta manera:

$$\begin{aligned} V(\rho)^T V(\rho) &= \begin{pmatrix} v_{11}(\rho) & v_{21}(\rho) & \dots & v_{n1}(\rho) \\ v_{12}(\rho) & v_{22}(\rho) & \dots & v_{n2}(\rho) \\ \vdots & \vdots & \vdots & \vdots \\ v_{1p}(\rho) & v_{2p}(\rho) & \dots & v_{np}(\rho) \end{pmatrix} \begin{pmatrix} v_{11}(\rho) & v_{12}(\rho) & \dots & v_{1p}(\rho) \\ v_{21}(\rho) & v_{22}(\rho) & \dots & v_{2p}(\rho) \\ \vdots & \vdots & \vdots & \vdots \\ v_{n1}(\rho) & v_{n2}(\rho) & \dots & v_{np}(\rho) \end{pmatrix} \\ &= \begin{pmatrix} v_1(\rho)^T \\ v_2(\rho)^T \\ \vdots \\ v_p(\rho)^T \end{pmatrix} \begin{pmatrix} v_1(\rho) & v_2(\rho) & \dots & v_p(\rho) \end{pmatrix} \end{aligned}$$

Entonces la entrada (i, j) de $V(\rho)^T V(\rho)$ es:

$$[V(\rho)^T V(\rho)]_{ij} = v_i(\rho)^T v_j(\rho)$$

Calculando la derivada:

$$\frac{d}{d\rho}[V(\rho)^T V(\rho)]_{ij} = \left(\frac{d}{d\rho}v_i(\rho)^T\right) v_j(\rho) + v_i(\rho)^T \left(\frac{d}{d\rho}v_j(\rho)\right)$$

En específico para el caso $i = j$:

$$\frac{d}{d\rho}[V(\rho)^T V(\rho)]_{ii} = 2\left(\frac{d}{d\rho}v_i(\rho)^T\right) v_i(\rho) \tag{2.7}$$

Lema 2.1. *Sea $V(\rho) \in \mathbb{R}^{n\times p}$ una matriz ortogonal y ρ su parámetro. Entonces [9]:*

$$(i) \quad \frac{d}{d\rho}V(\rho)^T V(\rho) = \left(\frac{d}{d\rho}V(\rho)^T\right) V(\rho) + V(\rho)^T \left(\frac{d}{d\rho}V(\rho)\right) = 0$$
$$(ii) \quad Diag\left(\left(\frac{d}{d\rho}V^T\right) V(\rho)\right) = 0$$

Demostración. (i) Para demostrar la primer propiedad se hace uso de que $V(\rho)^T V(\rho) = I_p$, entonces:

$$\frac{d}{d\rho}[V(\rho)^T V(\rho)] = 0$$

(ii) Se parte de la ecuación (2.7), y se usa el punto (i):

$$\frac{d}{d\rho}[V(\rho)^T V(\rho)]_{ii} = 2\left(\frac{d}{d\rho}v_i(\rho)^T\right) v_i(\rho) = 0$$

Como cada elemento i es cero, entonces en particular $Diag\left(\left(\frac{d}{d\rho}V(\rho)^T\right) V(\rho)\right) = 0$. □

Del lema 2.1 se tiene que $\left(\frac{d}{d\rho}V(\rho)^T\right) V(\rho)$ tiene diagonal igual a 0. Ahora, para derivar $f(\rho)$, primero se deriva la expresión $\frac{d}{d\rho}\left[V^T(A-\rho B)V\right]$:

$$\begin{aligned}\frac{d}{d\rho}\left[V^T(A-\rho B)V\right] &= \frac{d}{d\rho}\left[V^T AV\right] - \frac{d}{d\rho}\left[V^T \rho BV\right] \\ &= \frac{dV^T}{d\rho}AV + V^T A\frac{dV}{d\rho} - \frac{dV^T}{d\rho}\rho BV - V^T\left[BV + \rho B\left(\frac{dV}{d\rho}\right)\right] \\ &= \frac{dV^T}{d\rho}\left[A-\rho B\right]V + V^T\left[A-\rho B\right]\frac{dV}{d\rho} - V^T BV\end{aligned} \tag{2.8}$$

Sea V la matriz que diagonaliza $(A-\rho B)$, de manera que $V^T(A-\rho B)V = D$. Entonces 2.8:

$$\frac{d}{d\rho}\left[V^T(A-\rho B)V\right] = \frac{dV^T}{d\rho}VD + DV^T\frac{dV}{d\rho} - V^T BV \tag{2.9}$$

Al calcular la traza de (2.9) y usando del lema 2.1, se tiene que:

$$\begin{aligned} Tr\left[\frac{d}{d\rho}\left[V^T(A-\rho B)V\right]\right] &= Tr\left[\frac{dV^T}{d\rho}VD + DV^T\frac{dV}{d\rho} - V^TBV\right] \\ &= 2Tr\left[DV^T\frac{dV}{d\rho}\right] - Tr\left[V^TBV\right] \\ &= -Tr\left[V^TBV\right] \end{aligned} \tag{2.10}$$

2.3. Método Newton-Lanczos

El método de Newton establece que la iteración está dada por:

$$x_{(n+1)} = x_{(n)} - \frac{f(x_n)}{f'(x_n)}$$

con $f'(x_n)$ la derivada de $f(x_n)$.

Con esta fórmula y con (2.10), se puede calcular explícitamente ρ_{n+1} para la iteración de Lanczos [9]:

$$\begin{aligned} \rho_{n+1} =& \rho_n - \frac{Tr\left[V^T(A-\rho_nB)V\right]}{-Tr\left[V^TBV\right]} \\ =& \frac{\rho_nTr\left[V^TBV\right] + Tr\left[V^T(A-\rho_nB)V\right]}{Tr\left[V^TBV\right]} \\ =& \frac{\rho_nTr\left[V^TBV\right] + Tr\left[V^TAV\right] - \rho_nTr\left[V^TBV\right]}{Tr\left[V^TBV\right]} \\ =& \frac{Tr\left[V^TAV\right]}{Tr\left[V^TBV\right]} \end{aligned} \tag{2.11}$$

Una vez calculado el paso de la iteración, se enuncia el método de Newton-Lanczos para maximizar el cociente de trazas. El método recibe como entrada la matriz A, B y la dimensión a la cual se desea proyectar p [9]:

i = 1;
ρ_1 un número aleatorio;
ρ_0;
$f(\rho_1) = \sum_{j=1}^{p} \lambda_{(A-\rho_1 B)_j}$;
V = primeros p eigenvectores de $A - \rho_1 B$;
$tol = 1e-10$;
while *(*$i < 50$, $abs(\rho_i - \rho_{i-1}) > tol$*)* **do**
    $i = i + 1$;
    V = primeros p eigenvectores de $(A - \rho_i B)$;
    $\rho_i = Tr(V^T AV)/Tr(V^T BV)$;
    $f(\rho_i) = \sum_{j=1}^{p} \lambda_{(A-\rho_i B)_j}$;
end

Algorithm 2: Algoritmo de Newton-Lanczos

2.3.1. Condiciones necesarias de optimalidad

Considerando el problema de maximizar las trazas:

$$\max_{\substack{V \in \mathbb{R}^{n\times p} \\ V^T V = I}} \frac{Tr(V^T S_E V)}{Tr(V^T S_I V)} \tag{2.12}$$

La función lagrangiana asociada a este problema es la siguiente [9]:

$$L(V, \Gamma) = \frac{Tr(V^T S_E V)}{Tr(V^T S_I V)} - Tr[\Gamma(V^T V - I)]$$

Con $\Gamma \in \mathbb{R}^{p\times p}$, la matriz de multiplicadores de Lagrange de la forma:

$$\Gamma = \begin{pmatrix} \gamma_{(1)(1)} & \gamma_{(1)(2)} & \cdots & \gamma_{(1)(p-1)} & \gamma_{(1)(p)} \\ \gamma_{(2)(1)} & \gamma_{(2)(2)} & \cdots & \gamma_{(2)(p-1)} & \gamma_{(2)(p)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{(p-1)(1)} & \gamma_{(p-1)(2)} & \cdots & \gamma_{(p-1)(p-1)} & \gamma_{(p-1)(p)} \\ \gamma_{(p)(1)} & \gamma_{(p)(2)} & \cdots & \gamma_{(p)(p-1)} & \gamma_{(p)(p)} \end{pmatrix},$$

Ya que al multiplicar esta matriz por $[V^TV-I]$ (con $V=(v_1 \mid v_2 \mid ... \mid v_{p-1} \mid v_p)$) se tiene que:

$$V^TV-I=\begin{pmatrix} v_1^Tv_1-1 & v_1^Tv_2 & \dots & v_1^Tv_{p-1} & v_1^Tv_p \\ v_2^Tv_1 & v_2^Tv_2-1 & \dots & v_2^Tv_{p-1} & v_2^Tv_p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{p-1}^Tv_1 & v_{p-1}^Tv_2 & \dots & v_{p-1}^Tv_{p-1}-1 & v_{p-1}^Tv_p \\ v_p^Tv_1 & v_p^Tv_2 & \dots & v_p^Tv_{p-1} & v_p^Tv_p-1 \end{pmatrix},$$

entonces, la diagonal de $(\Gamma(V^TV-I))$ contiene p elementos que son los siguientes:

$$\begin{pmatrix} \gamma_{(1)(1)}(v_1^Tv_1-1) & +\gamma_{(1)(2)}(v_2^Tv_1) & +\dots & +\gamma_{(1)(p)}(v_p^Tv_1) \\ \gamma_{(2)(1)}(v_1^Tv_2) & +\gamma_{(2)(2)}(v_2^Tv_2-1) & +\dots & +\gamma_{(2)(p)}(v_p^Tv_2) \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_{(p-1)(1)}(v_1^Tv_{p-1}) & +\gamma_{(p-1)(2)}(v_2^Tv_{p-1}) & +\dots & +\gamma_{(p-1)(p)}(v_p^Tv_{p-1}) \\ \gamma_{(p)(1)}(v_1^Tv_p) & +\gamma_{(p)(2)}(v_2^Tv_p) & +\dots & +\gamma_{(p)(p)}(v_p^Tv_p-1) \end{pmatrix},$$

Para calcular la traza se suman estos elementos de la diagonal. De esta forma, se tienen p restricciones de la forma $v_i^Tv_i=1 \quad i=1,...,p$ y $p(p-1)$ restricciones de la forma $v_i^Tv_j=0 \quad i\neq j, \quad i,j=1,...,p$, cada una con su multiplicador lagrangiano.

Como la ecuación a maximizar tiene un maximizador global V^*, entonces existe un multiplicador matricial lagrangiano tal que en el óptimo:

$$\frac{\partial \mathcal{L}(V^*,\Gamma^*)}{\partial V}=0 \qquad con \qquad V^{T*}V^*=I \tag{2.13}$$

Para encontrar (V^*,Γ^*) de (2.13), primero se debe conocer la derivada de $\frac{\partial Tr(V^TMV)}{\partial V}$, con M cualquier matriz:

$$\frac{\partial Tr(V^TMV)}{\partial V}=(M^*+M)V \tag{2.14}$$

Derivando el lagrangiano (2.13) y utilizando (2.14) se tiene que:

$$\begin{aligned}\frac{\partial \mathcal{L}(V,\Gamma)}{\partial V} = \quad & \left[\frac{\partial Tr(V^T AV)}{\partial V}\right]\left[Tr(V^T BV)^{-1}\right] \quad + \\ & \left[Tr(V^T AV)\right]\left[\frac{\partial (Tr(V^T BV))^{-1}}{\partial V}\right] \quad - \\ & \left[\frac{\partial Tr[\Gamma(V^T V - I)]}{\partial V}\right]\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}(V,\Gamma)}{\partial V} = \quad & [2AV]\left[Tr(V^T BV)^{-1}\right] \quad + \\ & \left[Tr(V^T AV)\right]\left[\frac{2BV}{Tr(V^T BV)^2}\right] \quad - \\ & [V(\Gamma^* + \Gamma)]\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}(V,\Gamma)}{\partial V} = \quad & \left[\frac{2AVTr(V^T BV) - 2BVTr(V^T AV)}{(Tr(V^T BV))^2}\right] \quad + \\ & [V(\Gamma^* + \Gamma)]\end{aligned} \tag{2.15}$$

Igualando (2.15) a cero y acomodando la expresión, resulta:

$$\left[A - \frac{Tr(V^{T*} AV^*)}{Tr(V^{T*} BV^*)}B\right]V^* = \left[\frac{Tr(V^{T*} BV^*)}{2}V^*(\Gamma^{T*} + \Gamma^*)\right]$$

$$\left[A - \rho^* B\right]V^* = \left[\frac{Tr(V^{T*} BV^*)}{2}V^*(\Gamma^{T*} + \Gamma^*)\right] \tag{2.16}$$

Con $\rho^* = \frac{Tr(V^{T*} AV^*)}{Tr(V^{T*} BV^*)}$. Sea Q la matriz ortogonal que diagonaliza $\Gamma^{T*} + \Gamma^*$, entonces se puede escribir la expresión $(\Gamma^{T*} + \Gamma^*)$ como:

$$\left(\Gamma^{T*} + \Gamma^*\right) = Q\Sigma^* Q^T \qquad con \qquad Q^T Q = I$$

Con Σ^* una matriz diagonal. Multiplicando (2.16) por V^{T*} y calculando la traza de ambos lados de la ecuación y se tiene que:

$$Tr\left[V^{T*}(A-\rho^*B)V^*\right] = \left[\frac{Tr(V^{T*}BV^*)}{2}\right] Tr(\Gamma^{T*}+\Gamma^*)$$

$$Tr(\Gamma^{T*}+\Gamma^*) = 2\frac{Tr(V^{T*}(A-\rho^*B)V^*)}{Tr(V^{T*}BV^*)} = 0$$

$$Tr(Q\Sigma^*Q^T) = 2\frac{Tr(V^{T*}(A-\rho^*B)V^*)}{Tr(V^{T*}BV^*)} = 0 \tag{2.17}$$

Como $Tr(V^{T*}(A-\rho^*B)V^*) = 0$ entonces $Tr(Q\Sigma^*Q^T) = 0$, por lo que $Tr(\Sigma^*) = 0$. Definiendo $U^* = V^*Q$, con $U^*U = I$, la expresión (2.16) se puede escribir como:

$$\begin{aligned}
\left[A-\rho^*B\right]V^* &= \left[\frac{Tr(V^{T*}BV^*)}{2}V^*(Q\Sigma^*Q^T)\right] \\
&= \left[\frac{Tr(V^{T*}BV^*)}{2}U^*Q^{T*}(Q\Sigma^*Q^T)Q\right] \\
&= \left[\frac{Tr(V^{T*}BV^*)}{2}U^*\Sigma^*\right]
\end{aligned}$$

Definiendo $\Lambda_* = \frac{Tr(V^{T*}BV^*)}{2}\Sigma^*$, entonces:

$$\begin{aligned}
\left[A-\rho^*B\right]U^* &= U^*\Lambda_* \\
U^{T*}\left(A-\rho^*B\right)U^* &= \Lambda_* \\
Tr(U^{T*}\left(A-\rho^*B\right)U^*) &= Tr(\Lambda_*) = 0
\end{aligned} \tag{2.18}$$

De esta manera se tiene que la ecuación (2.18) es la condición necesaria para que U^*, ρ^* sean las óptimas

Capítulo 3

Experimentos numéricos

Para medir el desempeño del algoritmo Newton-Lanczos se compara con otros dos métodos, el Análisis Discriminante Lineal (ADL) y la Regresión Logística Multinomial (RLM). Los criterios para elegir el mejor serán la tasa de reconocimiento y el tiempo de cómputo sobre los conjuntos de datos JAFFE y MNIST. Como el algoritmo de Newton-Lanczos ocupa los eigenvectores de una matriz, en la primer parte de este capítulo se medirá el tiempo de Lanczos y el costo de la rutina SVD de R para una matriz en particular [1]. En la segunda parte del capítulo, se explicará el preprocesamiento de las bases de datos y como se crearon los conjuntos de entrenamiento y prueba. En la tercer y cuarta parte, se realizan los experimentos con la bases de MNIST y JAFFE respectivamente.

Para todos los experimentos de este capítulo, se utilizó el lenguaje de R en su versión 3.2.2 *Fire Safety*. Los cálculos fueron realizados en una iMac 3.2 Ghz Intel Core i3 con 12 GB de RAM.

3.1. Desempeño del método de Lanczos

Antes de analizar el tiempo de cómputo de Newton-Lanczos, se evaluará el desempeño del algoritmo de Lanczos comparándolo con la factorización *svd*.

[1] La rutina *svd* de *R* ocupa la función *gesdd* de la librería $LAPACK$. Para más información puede consultarse la página $http://www.netlib.org/lapack/$.

Esto experimento es para saber en que casos es conveniente utilizar Lanczos y en cuales conviene calcular todos eigenpares. Para la prueba se usó $A - \rho B$ con A y B las matrices de dispersión intra clase y entre clase de la base de datos MNIST y con $\rho = 3$. La figura 3.1 resume el desempeño de ambos métodos.

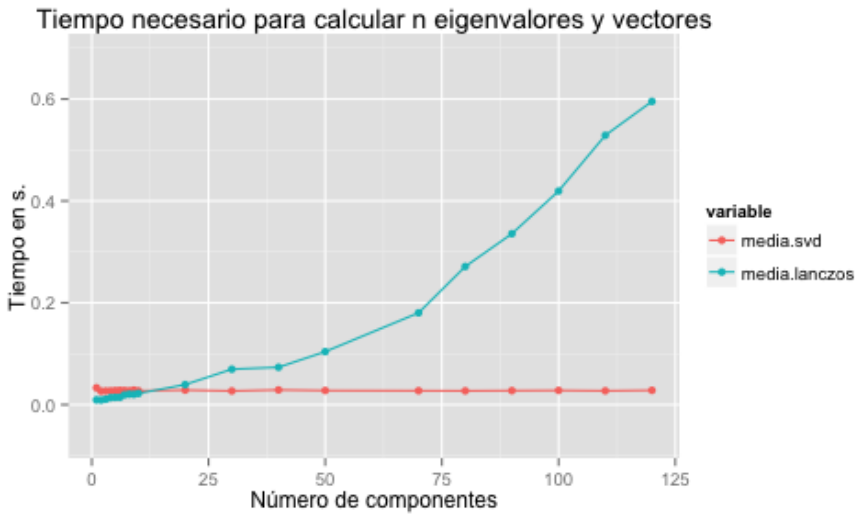


Figura 3.1: Desempeño de Lanczos. En el eje x se representa k, el número de eigenpares a calcular de la matriz $A - \rho B$. La matriz original es de 193 x 193. El tiempo reportado para cada k, es el promedio de 30 repeticiones para cada método.

Se observa que, para k menor a 10, el algoritmo de Lanczos es más rápido que la factorización SVD de la matriz completa. Para valores más grandes, conviene factorizar toda la matriz. Esto puede deberse a que en aritmética inexacta Lanczos presenta el problema de *Ghost eigenvalues* (eigenvalores que se repiten, pero que son espurios) [6]. Estos se originan porque la ortogonalidad entre los eigenvectores se pierde conforme la dimensionalidad aumenta. Para solucionar este problema, se tuvo que realizar una reortogonalización completa en cada iteración, lo que pudo aumentar el tiempo de cómputo.

3.2. Modelos comparados y preprocesamiento de las bases

En este capítulo, el método de Newton-Lanczos se comparó otros dos métodos de clasificación: la RLM y con el ADL. Para el primero, se usó la función *multinom* del paquete *nnet*, mientras que para el segundo se utilizó una modificación de la función *lda* [2] del paquete *MASS*. Ambas funciones fueron desarrolladas por Brian Ripley, profesor de estadística aplicada de la Universidad de Oxford.

Como primer paso, se definieron los tamaños del conjunto de entrenamiento y prueba. Para el entrenamiento de MNIST se ocuparon 4000 datos (400 por clase); es decir, el 6 % del total. En el caso de JAFFE se utilizaron 50 datos (5 por clase); es decir, el 23 % del total. En ambos, el resto de los datos se ocupó en el conjunto de prueba.

Para el preprocesamiento de las bases, cada imagen se convirtió en un vector con tamaño asociado al número de píxeles. Esto genera un problema de dimensionalidad, por lo que se realizó un paso de reducción dimensional. Se tomaron las primeras componentes principales (PCA) del conjunto de entrenamiento de manera que se explicara el 95 % de la varianza total, después se proyectó el conjunto de prueba. (De esta manera, los individuos en lugar de pertenecer a un espacio de $\mathbb{R}^{m\times n}$, pertenecen a uno menor). Para MNIST se tomaron 193 de las 784 componentes; en cambio, para JAFFE solo 50 de las 65536.

Una vez hecho el preprocesamiento, se realizaron 2 experimentos para cada base de datos. En el primero se utilizó el método de Newton-Lanczos con $V \in \mathbb{R}^{m\times 20}$. Después de esto se proyectaron a los individuos originales a un espacio 20-dimensional. Los objetivos fueron los siguientes:

- Analizar como cambia el valor de $f(\rho)$ con el método de Newton-Lanczos
- Ver como se ven proyectados los datos en dimensiones menores

En el segundo experimento se entrenaron los 3 métodos con distintas p. Después se midió la tasa de reconocimiento alcanzada (usando el conjunto de prueba) y su tiempo de ejecución.

[2] La modificación está presentada en los códigos del apéndice B

3.3. Base de datos MNIST

Las siglas MNIST provienen de (*Mixed National Institute of Standards and Technology*), la cual contiene datos de 70,000 dígitos escritos a mano que se usan comúnmente para probar códigos de clasificación de imágenes. De esta manera, en el experimento se tendrán 10 clases: los dígitos del 0-10. La base de datos surge de la colaboración de Yann LeCun del Instituto Courant (NYU), de Corinna Cortes de Google Labs (NY) y de Christopher J.C. Burges de Microsoft Research en Redmond. Puede descargarse de la página Web *yann.lecun.com/exdb/mnist*, cuyo link sigue vigente al 13 de marzo del 2016.

Las imágenes están en un formato especial llamado IDX. Para poderlas convertir en un archivo leíble para R, se utilizó la función *readMNIST* del paquete *darch*. Esta paquetería convierte los archivos a objetos *.RData*. La figura 3.2 contiene una extracto de tamaño 225 de todo el conjunto.

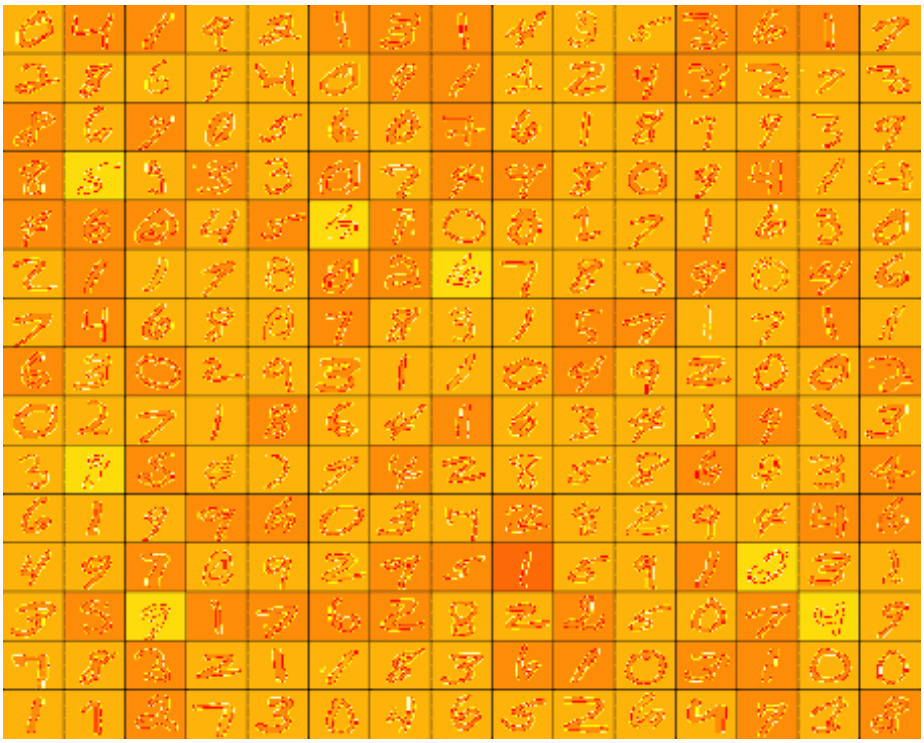


Figura 3.2: Ejemplo de números de la base de datos (MNIST).

3.3.1. Proyección sobre un espacio de dimensión 20

1) Cotas para ρ^*

En el capítulo 2 se presentaron cotas para ρ^* y en esta sección se ejemplificará la segunda de ellas para esta base de datos. Se calcularon los eigenvalores de la matriz intraclase (B) y de la matriz entre clases (A). Los 20 eigenvalores más grandes y más pequeños de la primera son [3]:

12694	10367	7822	6793	6013	5887	5269	4704	4474	3969
3833	3428	3124	3015	2978	2627	2535	2353	2232	2150

73	72	72	71	71	70	70	69	68	68
67	67	66	66	65	64	64	63	63	62

Mientras que los 9 eigenvalores más grandes de la matriz entre clases (A) son (los demás son 0):

12865	9821	7408	4736	3558	2312	1879	885	631

Sustituyendo estos valores en la fórmula de las cotas de la raíz, se tiene que:

$$\frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_i}} \leq \rho^* \leq \frac{\sum_{i=1}^{p} \lambda_{(A)_i}}{\sum_{i=1}^{p} \lambda_{(B)_{n-i+1}}}$$

$$\frac{44099.28}{96277.31} \leq \rho^* \leq \frac{44099.28}{1364.54}$$

$$0.4580443 \leq \rho^* \leq 32.31806$$

A continuación se presentan como se ven $(\rho^*, f(\rho))$ con cada iteración para el método de Newton-Lanczos.

[3] los eigenvalores menores a $1e^{-10}$ se consideraron numéricamente como 0

2) Valores de $(\rho^*, f(\rho))$

Para el punto inicial de este experimento se utilizará el punto medio del intervalo. Los criterios de paro se fijaron con una tolerancia de $1e^{-10}$ y que las iteraciones sean menor a 50. Con este ejemplo, se obtienen los siguientes resultados para ρ y $f(\rho)$:

iter	ρ	$f(\rho)$
1	16.38805180	$-22,326.51$
2	0.03121373	$42,841.16$
3	1.11003833	$15,197.50$
4	2.05957127	$3,926.463$
5	2.50843464	444.8128
6	2.57388828	9.007894
7	2.57526971	.004008613
8	2.57527033	$7.891856e^{-10}$
9	2.57527033	$5.371703e^{-12}$

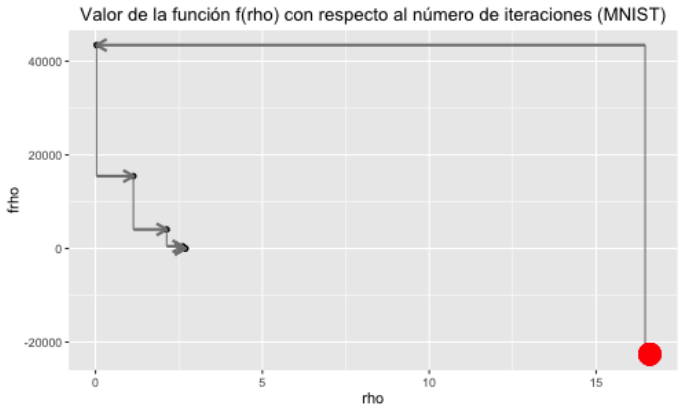


Figura 3.3: Valores de ρ y $f(\rho)$ para distintas iteraciones (MNIST). El punto rojo es el punto inicial. Datos del conjunto de entrenamiento.

En la figura 3.3 se muestran $(\rho, f(\rho))$ para cada una de las 9 iteraciones. El punto inicial es el rojo, en este caso $\rho = 16.39$, mientras que $f(\rho)$ toma el valor de $-22,326$. Tras la primer iteración, ρ toma el valor de 0.03, y $f(\rho)$ es igual $42,841$. Conforme hay más iteraciones, ρ y $f(\rho)$ convergen a los óptimos.

3) Datos proyectados en 4 dimensiones

Se maximizó la traza del cociente para un espacio de 20 dimensiones. En la figura 3.4 se muestra como se agrupan los individuos del conjunto de entrenamiento conforme las iteraciones avanzan.

Figura 3.4: Ejemplo de proyección en 20 dimensiones (MNIST). Las dos imágenes superiores son los datos de entrenamiento proyectadas sobre las 4 primeras componentes tras la primer iteración. Las dos intermedias son en la iteración 3 y las últimas 2 son la última iteración. Datos del conjunto de entrenamiento.

3.3.2. Comparación con otros métodos

Para la comparación de Newton-Lanczos con la RLM y el ADL se tomaron las siguientes consideraciones:

- Las dimensiones a considerar (k) son: 2, 5, 10, 15, 20, 40, 60, 80, 100, 120, 140, 160, 180 y 193
- El número de variables que entran en cada modelo es el mismo (dada la dimensión k)
- Los modelos se ajustan con el conjunto de entrenamiento y se calcula el error con el de prueba
- Para Newton-Lanczos y ADL se proyectarán a un espacio k-dimensional y se usará 3-vecinos más cercanos para clasificarlos
- Para el caso de regresión logística multinomial se elegirá la clase que tenga una mayor probabilidad posterior de selección

1) Tasa de reconocimiento

La tasa de reconocimiento se mide como el número de individuos clasificados correctamente entre el número total del conjunto de test. Como el objetivo de esta sección es observar como se modifica la tasa con respecto aumenta k, se fijó el número de vecinos más cercanos a 3 para el caso de ADL y de Newton-Lanczos y también el tamaño del conjunto de entrenamiento. En la figura 3.5 se muestra que el modelo RLM y el ADL se comportan mejor para dimensiones pequeñas. Para casos de k mayor a 120, el método de Newton-Lanczos alcanza una mejor tasa de reconocimiento.

Es interesante observar que si se busca la mejor proyección con para k = 1 y k = 2, Newton-Lanczos resulta ser muy buen método. En la 3.4 se observó como los individuos se agrupan con este método.

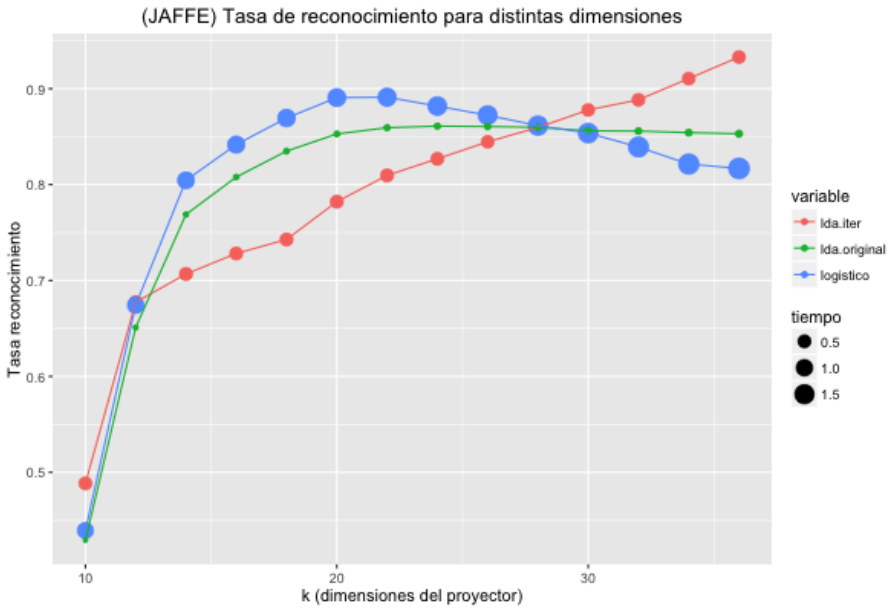


Figura 3.5: Tasa de reconocimiento base (MNIST). En azul se representa el RLM, en verde el ADL y en rojo el método de Newton-Lanczos. En el eje y se muestra la tasa de reconocimiento y en el eje x la dimensión a considerar (k). El tamaño representa el tiempo. Datos del conjunto de prueba.

2) Tiempo de ejecución

A continuación, se comparan los tiempos de ejecución de cada método. Para comparar los tiempos de ejecución justamente, se tomaron 3 distintos:

- 1) (PCA) Tiempo para calcular los componentes principales del conjunto de entrenamiento
- 2) (Proyectar) Tiempo para calcular la proyección del conjunto de test
- 3) (Modelo) Tiempo necesario para hacer las operaciones de cada método

PCA(s)	Proyectar(s)
11.728	21.680

La siguiente tabla muestra los tiempos de cómputo para los 3 métodos y con distinto número de componentes.

Comp	lda.iter	logístico	lda.orig	Comp	lda.iter	logístico	lda.orig
2	0.9092	0.2712	0.01	80	0.9222	4.3186	0.2684
5	0.8576	0.5188	0.0466	100	0.8764	5.6474	0.3572
10	0.864	1.0646	0.025	120	0.8952	8.1026	0.4942
15	0.8768	1.103	0.0372	140	0.9016	9.2944	0.7902
20	0.8886	1.2888	0.0524	160	0.8782	10.1534	0.908
40	0.886	2.5528	0.1004	180	0.898	12.0426	1.0596
60	0.8728	3.2034	0.1744	193	0643	14.2512	1.2186

El tiempo está en segundos (s).

Para cada iteración del método de Newton-Lanczos, se calcularon todos los eigenvalores ya que al no calcularlos, solo se tendrá una aproximación a sus valores y sus eigenvectores. Por este motivo, se decidió sacrificar tiempo de cómputo por precisión.

3.4. Base de datos JAFFE

Ahora se analizará la base JAFFE *Japanese Female Facial Expression*, que consta de 213 fotos provenientes de 10 personas distintas. Este conjunto de datos fue creado inicialmente para clasificar emociones, pero se usa comúnmente para probar otros tipos de algoritmos de clasificación. En nuestro caso será usado para clasificar el rostro de la persona, de esta manera se tendrán 10 clases. La base fue creada por Michael Lyons, Miyuki Kamachi y Jiro Gyoba en el departamento de psicología de la universidad de Kyushu. Puede descargarse de la página Web *www.kasrl.org/jaffe.html*, cuyo link sigue vigente al 13 de marzo del 2016.

Las imágenes se encuentran en un formato *.tiff*, por lo que se requirió la función *readTIFF* del paquete $tiff$ el cual permite leerlas y convertirlas en un formato *.RData*. La figura 3.6 contiene un extracto de tamaño 35 de la base original.

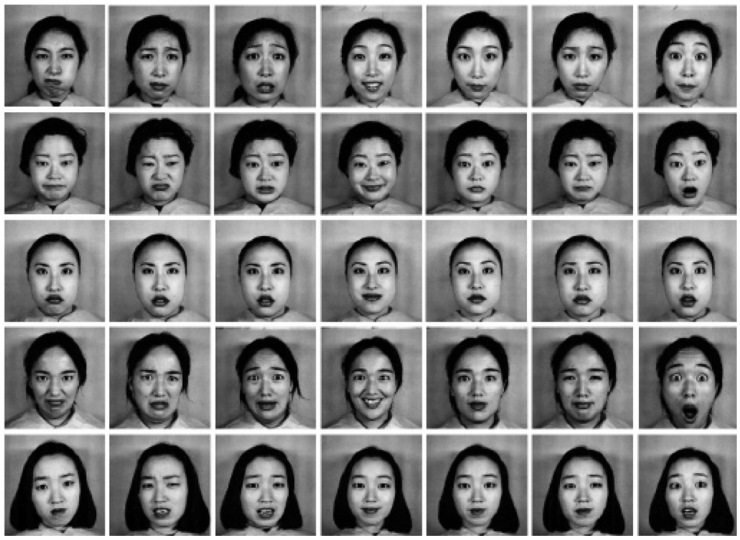


Figura 3.6: Ejemplo de la base de datos (JAFFE).

3.4.1. Proyección sobre un espacio de dimensión 20

1) Cotas para ρ^*

Al igual que el ejemplo de MNIST, se ejemplificará la segunda cota presentada en el capítulo 2. Para esto se calcularon los eigenvalores de la matriz intraclase (B) y de la matriz entre clases (A). Los 20 eigenvalores más grandes y más pequeños de la primera son: [4]

5303	4627	2083	1556	1370	1177	851	781	740	705
629	596	562	520	498	466	453	403	397	382

250	245	231	223	219	203	198	192	176	147
141	0	0	0	0	0	0	0	0	0

De la misma manera, se calculan los 9 eigenvalores más grandes de la matriz entre clases (Los demás son 0):

14051	12922	7775	3247	3138	2849	1918	1336	1317

Usando las formulas del capítulo 2, se tienen las siguientes cotas para ρ^*:

$$\frac{\sum_{i=1}^{p} \lambda_{A_i}}{\sum_{i=1}^{p} \lambda_{B_i}} \leq \rho^* \leq \frac{\sum_{i=1}^{p} \lambda_{(A)_i}}{\sum_{i=1}^{p} \lambda_{(B)_{n-i+1}}}$$

$$\frac{48553}{24099} \leq \rho^* \leq \frac{48553}{2225}$$

$$2.0147 \leq \rho^* \leq 21.8216$$

[4] los eigenvalores menores a $1e^{-10}$ se consideraron numéricamente como 0

2) Valores de $(\rho^*, f(\rho^*))$

En la figura 3.7 se observa fácilmente rapidez de convergencia. En esta gráfica se muestra en el eje y la raíz cuadrada de $f(\rho)$. Los criterios de paro se fijaron iguales que el ejemplo pasado (tol: $1e^{-10}$ y menos de 50 iteraciones). Para la base de JAFFE, se encontraron los siguientes resultados:

$iter$	ρ	$f(\rho)$
1	11.88493	6093.742
2	14.33840	80.44169
3	14.37160	$.01093739e$
4	14.37161	$1.873559e^{-10}$

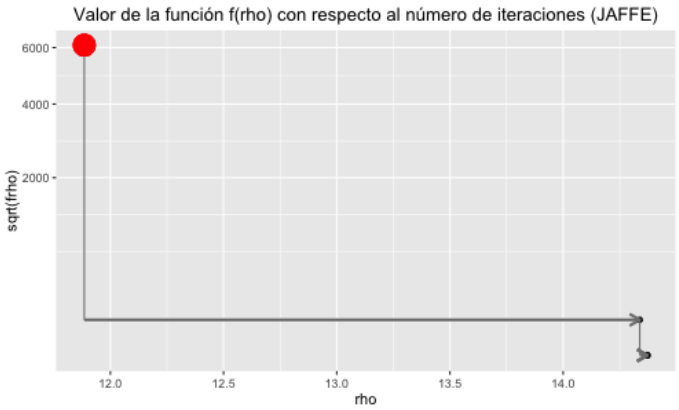


Figura 3.7: Valores de ρ y $f(\rho)$ para las iteraciones (JAFFE). El punto rojo es el punto inicial. Datos del conjunto de entrenamiento.

Se ve que la convergencia cuadrática del método de Newton ayuda a que se alcance el óptimo en pocas iteraciones. En la cuarta iteración $f(\rho)$ ya toma el valor de $1.87e^{-10}$, muy cercano a 0.

3) Datos proyectados en 4 dimensiones

De nuevo se maximizó la traza del cociente para un espacio de 20 dimensiones. En la figura 3.8 se muestra como se agrupan los individuos del conjunto de entrenamiento conforme las iteraciones avanzan.

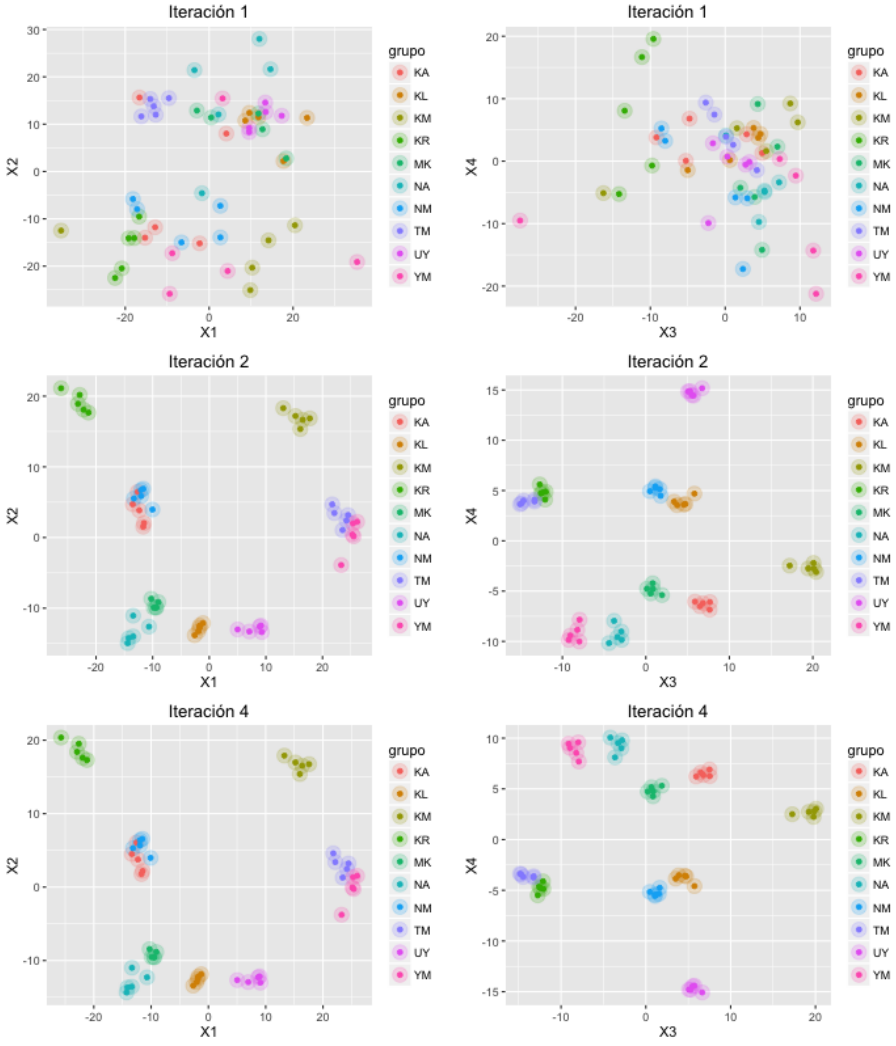


Figura 3.8: Ejemplo de proyección en 20 dimensiones (JAFFE). Las dos imágenes superiores son los datos de entrenamiento proyectadas sobre las 4 primeras componentes tras la primer iteración. Las dos intermedias son en la iteración 3 y las últimas 2 son la última iteración.

3.4.2. Comparación con otros métodos

De la misma forma que el ejemplo pasado se compararon los métodos en este caso. Lo único distinto serán las dimensiones a considerar k = 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46 y 49.

1) Tasa de reconocimiento

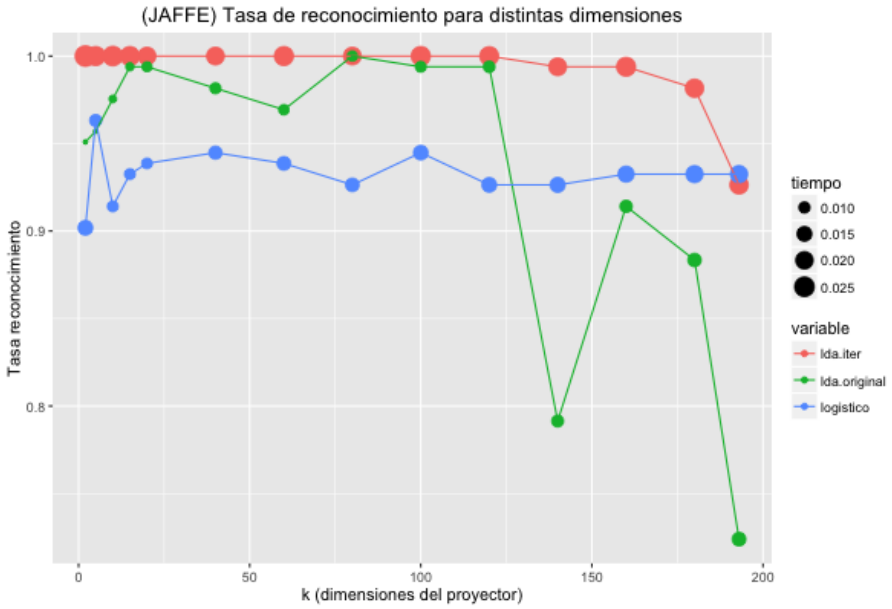


Figura 3.9: Tasa de reconocimiento base (JAFFE). En azul se representa el RLM, en verde el ADL y en rojo el método de Newton-Lanczos. En el eje y se muestra la tasa de reconocimiento y en el eje x la dimensión a considerar (k). El tamaño representa el tiempo. Se observa que en el ADL los últimos 4 puntos bajan la predicción. Esto es porque las variables son colineales y al calcular la inversa, causa inestabilidad numérica. Datos del conjunto de entrenamiento.

En la figura 3.9 se observa que el método de lda iterativo no tiene ningún error de clasificación para espacios de proyección menor a 40 dimensiones, reduciéndose en espacios de proyección con dimensión más alta. Esto puede deberse a que las últimas componentes son las menos discriminadoras, pero el método de 3-vecinos más cercanos las pondera igual. Una posible modificación que se puede realizar, es ponderar cada componente de acuerdo al poder discriminativo del

correspondiente eigenvector.

En la figura 3.10 se muestran las últimas 4 componentes. Se observa que los individuos ya no están tan agrupados como en las primeras 4, por lo que el clasificador de 3-vecinos más cercanos podría no estar funcionando bien para $k > 40$.

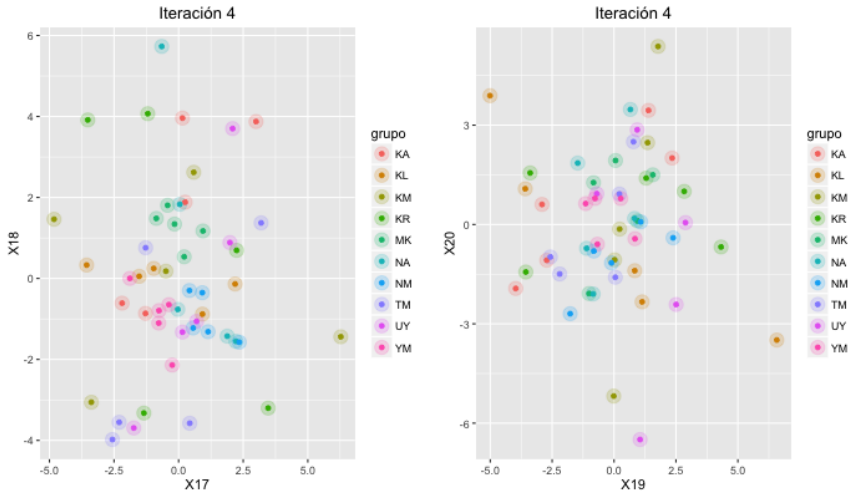


Figura 3.10: Individuos proyectados (JAFFE). Conjunto de entrenamiento proyectados sobre las ultimas 4 componentes (17,18,19,20). Se observa que los grupos siguen estando relativamente cerca, pero si se toma 3-vecinos más cercanos no necesariamente se escogerá la persona correcta.

2) Tiempo de ejecución A continuación, se comparan los tiempos de ejecución de cada método. Para compararlos justamente, se tomaron 3 fases distintas y se comparó solamente el tiempo del modelo.

- 1) (PCA) Tiempo para calcular los componentes principales del conjunto de entrenamiento
- 2) (Proyectar) Tiempo para calcular la proyección del conjunto de test
- 3) (Modelo) Tiempo necesario para hacer las operaciones de cada método

PCA(s)	Proyectar(s)
4.666	4.801

Comp	lda.iter	logístico	lda.orig	Comp	lda.iter	logístico	lda.orig
10	0.0352	0.0098	0.0068	31	0.0312	0.0436	0.0136
13	0.0324	0.0146	0.0074	34	0.0314	0.0176	0.0134
16	0.0378	0.018	0.0082	37	0.0314	0.0224	0.0144
19	0.031	0.0114	0.0104	40	0.0312	0.0266	0.0152
22	0.0308	0.0128	0.0112	43	0.0294	0.0248	0.0164
25	0.0326	0.0136	0.0102	46	0.03	0.0228	0.0188
28	0.0318	0.0158	0.0112	49	0.0204	0.0594	0.0176

El tiempo está en segundos (s).

Para cada iteración del método de Newton-Lanczos, se calcularon todos los eigenvalores (Excepto si el número de eigenpares a calcular es menor al 5 % del total) ya que al no calcular todos, solo se tendrá una aproximación a sus valores y sus eigenvectores. Por este motivo, se decidió sacrificar tiempo de cómputo por precisión.

Capítulo 4

Conclusiones

En este trabajo se expone la técnica de optimización de Newton-Lanczos aplicada al problema de aprendizaje supervisado del Análisis Discriminante Lineal de Fisher (ADLF). El ADLF busca encontrar la mejor matriz de proyección que maximice el cociente de trazas, logrando que los individuos de una misma clase sean proyectados lo más cercano posible entre ellos y lo más separado posible de otra clase. Anteriormente, la solución era considera computacionalmente costosa de resolver; sin embargo, en la actualidad se puede aprovechar la velocidad de convergencia del método de Newton y la rapidez del algoritmo de Lanczos para calcular eigenpares.

El método de Lanczos es muy efectivo para calcular solamente algunos de los eigenpares de las matrices. Bajo el contexto de aritmética inexacta, Lanczos implementado con reortogonalización completa resulta ser más eficiente que la factorización SVD, cuando se desea calcular menos del 5 % de los eigenpares. Aunque este número parece ser demasiado pequeño resulta de gran utilidad; por ejemplo, si la matriz tiene dimensión de 1000×1000, tener 50 eigenpares podría ser adecuado para el problema.

En este trabajo de tesis se comparó el método ADLF vía Newton-Lanczos con el Análisis Discriminante Lineal (ADL) y la técnica de Regresión Lineal Múltiple (RLM). La comparación se realizó empleando las bases JAFFE y MNIST. Los resultados obtenidos con ADLF vía Newton-Lanczos tuvieron un desempeño similar en comparación con los otros métodos en términos de tasa de reconocimiento y tiempo de cómputo.

Las conclusiones más relevantes de esta tesis son las siguientes:

- Se implementó computacionalmente una técnica de optimización que anteriormente resultaba difícil de resolver.
- Una de las principales ventajas de esta metodología es que no requiere ningún supuesto sobre la distribución de los datos.
- Se evaluó el desempeño de esta metodología con respecto a técnicas conocidas y los resultados fueron satisfactorios.
- Se realizaron dos pruebas diferentes y se obtuvo que en algunos casos el ADLF vía Newton-Lanczos tuvo una precisión mayor con respecto a los otros dos métodos.

Una de las complicaciones del algoritmo de Lanczos es la reortogonalización de la base. En este estudio se utilizó el método de reortogonalización completa; sin embargo, existen modificaciones al algoritmo que pueden ser exploradas con el objetivo de lograr mayor eficiencia en términos computacionales. Por ejemplo, J.W. Demmel (1997) [2] propone algunas alternativas que pueden ser utilizadas para mejorar el proceso de reortogonalización de la base en el algoritmo de Lanczos.

Apéndice A

Apéndice A: Optimización de $Tr(V^T AV)$

Sea $A \in \mathbb{R}^{nxn}$ una matriz simétrica cuya factorización espectral es:

$$\begin{aligned} A =& U^T \Lambda U \qquad U^T U = I_n \\ \Lambda =& diag(\lambda_{A_1}, ..., \lambda_{A_n}) \end{aligned}$$

donde $\lambda_{A_1} \geq \lambda_{A_2} \geq ... \geq \lambda_{A_n}$ son los valores propios de A y U una matriz ortogonal. En este apéndice se demostrará que:

$$\max_{\substack{V^T V = I \\ V \in \mathbb{R}^{nxp}}} Tr(V^T AV) = \lambda_{A_1} + \lambda_{A_2} + ... + \lambda_{A_p} \tag{A.1}$$

A.1. Problema relajado

Sean $\mathcal{U}_p$ y $\mathcal{V}_p$ dos conjuntos de matrices en $\mathbb{R}^{nxp}$ tales que:

$$\begin{aligned} \mathcal{U}_p =& \{V \in \mathbb{R}^{nxp} | V^T V = I_p\} \\ \mathcal{V}_p =& \{V \in \mathbb{R}^{nxp} | diag(V^T V) = I_p\} \end{aligned}$$

El conjunto $\mathcal{V}_p$ contiene a todas las matrices donde que cada columna tiene norma euclidiana igual a 1.

Lema A.1. *El conjunto $\mathcal{V}_p$ es compacto en $\mathbb{R}^{nxp}$*

Demostración. Por definición, si $V \in \mathcal{V}_p$ entonces $||V||_F = \sqrt(p)$. Más aún, $\mathcal{V}_p$ contiene todos sus puntos límite. □

Si se relaja el problema original como:

$$\max_{V \in \mathcal{V}_p} Tr(V^TAV) \tag{A.2}$$

Como $\mathcal{V}_p$ es un conjunto compacto, la función continua $Tr(V^TAV)$ alcanza su valor máximo (o mínimo) en este conjunto. Ahora, como $\mathcal{U}_p \subset \mathcal{V}_p$, se tiene inmediatamente la desigualdad:

$$\max_{V \in \mathcal{U}_p} Tr(V^TAV) \leq \max_{V \in \mathcal{V}_p} Tr(V^TAV) \tag{A.3}$$

Por lo tanto, se se procede a establecer el siguiente resultado:

Teorema A.1.

$$\max_{V \in \mathcal{V}_p} Tr(V^TAV) = \lambda_{A_1} + \lambda_{A_2} + ... + \lambda_{A_p}$$

Demostración. Para $V \in \mathcal{V}_p$ con $V = (v_1 \ v_2 \ ... \ v_p)$ y v_j la j-ésima columna de V. Se define el vector:

$$\mathbf{v} = (v_1^T \ v_2^T \ ... \ v_p^T)^T \ \in \ \mathbb{R}^{np}$$

o lo que es equivalente:

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{pmatrix}$$

Entonces la $Tr(V^TAV) = v_1^TAv_1 + v_2^TAv_2 + ... + v_p^TAv_p$ y el problema A.2 puede ser formulado como sigue:

$$\max_{\substack{\mathbf{v}^T\mathbb{B}_j\mathbf{v}=1 \\ j=1,...,p}} \mathbf{v}^T\mathbb{A}\mathbf{v} \tag{A.4}$$

con las matrices $\mathbb{A}$ y $\mathbb{B}_j$:

$$\mathbb{A} = \begin{pmatrix} A & 0 & \dots & 0 & \dots & 0 \\ 0 & A & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & A & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & A \end{pmatrix}$$

$$\mathbb{B}_j = \begin{pmatrix} 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & I_n & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}$$

En ambas matrices hay p bloques en los renglones y p bloques en las columnas. Para $\mathbb{B}_j$ el bloque (j, j) es I_n y todos los demás son matrices cero. De esta manera, la función lagrangiana asociada al problema es:

$$\mathcal{L}_j(\mathbf{v}, \eta) = \mathbf{v}^T \mathbb{A}\mathbf{v} - \sum_{j=1}^{p} (\eta_j(\mathbf{v}^T \mathbb{B}_j \mathbf{v} - 1))$$

Las condiciones de primer orden para la solución óptima son (considerando que A es simétrica:

$$\begin{aligned} \nabla_{\mathbf{v}} \mathcal{L}(\mathbf{v}, \eta) = &\quad 2\mathbb{A}\mathbf{v} - 2\sum_{j=1}^{p} (\eta_j(\mathbf{v}^T \mathbb{B}_j \mathbf{v})) = \quad 0 \\ \nabla_{\eta_j} \mathcal{L}(\mathbf{v}, \eta) = &\quad \mathbf{v}^T \mathbb{B}_j \mathbf{v} = 1 \quad con \quad j = 1, ..., p \end{aligned} \tag{A.5}$$

De la primera ecuación de A.5, se tiene que:

$$\left(\mathbb{A} - \sum_{j=1}^{p} \eta_j \mathbb{B}_j \right) v = 0$$

Donde la matriz $(\mathbb{A} - \sum_{j=1}^{p} \eta_j \mathbb{B}_j)$ es diagonal a bloques. De hecho, el bloque (j, j) es:

$$(A - \eta_j I_n)v_j = 0 \quad \Rightarrow \quad Av_j = \eta_j v_j$$

Entonces, η_j debe ser un eigenvalor de A y v_j su correspondiente eigenvector. Tomando de nuevo la ecuación de arriba, y multiplicándola por v^T se obtiene:

$$v^T \mathbb{A} v = \sum_{j=1}^{p} (\eta_j)(v^T \mathbb{B}_j v)$$

Ahora, usando la segunda ecuación de A.5, se tiene que:

$$v^T \mathbb{A} v = \sum_{j=1}^{p} \eta_j$$

Por lo tanto, con el fin de maximizar la función $v^T \mathbb{A} v$ en $\mathcal{V}_p$, se deben escoger los primeros p eigenvalores de la matriz A. Esto es:

$$\eta_j = \lambda_{A_j}, \qquad j = 1, ..., p$$

□

Ahora, se tienen todas las piezas para demostrar el resultado principal.

Teorema A.2.

$$\max_{\substack{V^T V = I_p \\ V \in \mathbb{R}^{nxp}}} Tr(V^T AV) = \lambda_{A_1} + \lambda_{A_2} + ... + \lambda_{A_p}$$

Demostración. Se ha mostrado que:

$$\max_{V \in \mathcal{U}_p} Tr(V^T AV) \leq \lambda_{A_1} + \lambda_{A_2} + ... + \lambda_{A_p}$$

□

Considerando la matrix $U \in \mathbb{R}^{nxn}$ de la factorización espectral de A, tal que $U = (u_1 \ u_2 \ ... \ u_n)$. Tomando las primeras p columnas de U se define U^*:

$$\begin{aligned} U^* &= (u_1 \ u_2 \ ... \ u_p) \quad con \quad U^* \in \mathcal{U}_p \quad y \\ Tr(U^* AU) &= \lambda_{A_1} + \lambda_{A_2} + ... + \lambda_{A_p} \end{aligned}$$

A.2. Observaciones finales

Se toman los resultados de la sección anterior para hacer las siguientes observaciones:

Observación A.1. *Se puede extender el último teorema, usando el mismo conjunto $\mathcal{V}_p$ y cambiando la maximización a minimización para obtener que:*

$$\min_{V \in \mathcal{U}_p} Tr(V^T AV) = \lambda_{A_n} + \lambda_{A_{n-1}} + ... + \lambda_{A_{n-(p-1)}} \tag{A.6}$$

Observación A.2. *Cuando $p = 1$, se pueden obtener las bien conocidas propiedades del cociente de Raleigh:*

$$\begin{aligned} \left(\min_{v^v v=1} v^T Av\right) &= \lambda_{A_n} \\ \left(\max_{v^v v=1} v^T Av\right) &= \lambda_{A_1} \end{aligned} \tag{A.7}$$

Apéndice B

Apéndice B: Códigos en R

B.1. Experimentos numéricos

B.1.1. 01_*Prueba*20*comp*

```
library(ProjectTemplate)
reload.project()

# Jaffe.20 & Mnist.20  are 5 element lists:
# 1) iterative model
# 2) data frame with \rho, f\rho values
# 3) projection matrix V
# 4) test error
# 5) knn model

Jaffe.20 <- prueba.20comp(JAFFE.train, JAFFE.test, "JAFFE")
Mnist.20 <- prueba.20comp(MNIST.train, MNIST.test, "MNIST")

Jaffe.20$modelo.iterativo
Mnist.20$modelo.iterativo
```

B.1.2. 02_*Models_comparisons*

```
# 1) Data y proyeccion size ----------------------------------
```

```
library(ProjectTemplate)
reload.project()

numComp.Jaffe <- c(10,13,16,19,22,25,28,31,
                   34,37,40,43,46,49)
numComp.Mnist <- c(2,5,10,15,20,40,60,80,
                   100,120,140,160,180,193)

comp.error.MNIST<-comparacion.modelos(train.p = MNIST.train,
                                      test.p = MNIST.test,
                                      dataset = "MNIST",
                                      numComp = numComp.Mnist,
                                      fortran = FALSE)

comp.error.JAFFE<-comparacion.modelos(train.p = JAFFE.train,
                                      test.p = JAFFE.test,
                                      dataset = "JAFFE",
                                      numComp = numComp.Jaffe,
                                      fortran = FALSE)
# 2) Graphs ------------------------------------------------------

graficar.comparacion(datos = comp.error.MNIST,
                     numComp = numComp.Mnist,
                     label = "MNIST")

graficar.comparacion(datos = comp.error.JAFFE,
                     numComp = numComp.Jaffe,
                     label = "JAFFE")
```

B.1.3. 03_*Profiling*

```
# 1) Data y proyeccion size --------------------------------------
library(ProjectTemplate)
reload.project()

# 1) Componentes -------------------------------------------------
numComp.Jaffe <- c(10,13,16,19,22,25,28,31,
                   34,37,40,43,46,49)
```

```
numComp.Mnist <- c(2,5,10,15,20,40,60,80,
                   100,120,140,160,180,193)
# 2) Profiling ------------------------------------------
prof.Mnist <- prolifiling.model(train.p = MNIST.train,
                                test.p = MNIST.test,
                                dataset = "MNIST",
                                numComp = numComp.Mnist)
prof.Jaffe <- prolifiling.model(train.p = JAFFE.train,
                                test.p = JAFFE.test,
                                dataset = "JAFFE",
                                numComp = numComp.Jaffe)

# 3) Graphs ---------------------------------------------
graficar.profiling(times.profiling = prof.Mnist,
                   numComp = numComp.Mnist,
                   dataset = "MNIST")
graficar.profiling(times.profiling = prof.Jaffe,
                   numComp = numComp.Jaffe,
                   dataset = "JAFFE")
```

B.2. Funciones

B.2.1. *load_libraries*

```
# 1) Load libraries -------------------------------------
set.seed(12345)
library(plyr) # Tools for split, applying and combine data
library(dplyr) # package of grammar of data manipulation
library(tidyr) # package to efficiently "tidy" data
library(FactoMineR) # Package for multivariate analysis
library(ggplot2) # Graphing package "Grammar of Graphics"
library(gridExtra) # Package to work with grid graphics
library(class) # Functions for classification, including knn
library(nnet) # Software for multinomial linear models
library(MASS) # "Applied Statistics with S"
```

```
# 1) Fisher eficiente ----------------------------------
lda.iter.ef <- function (test.p,
                         train.p,
                         espacio.proy,
                         prior = proportions,
                         tol = 1e-10) {

  # Prepare matrix of data
  x <- train.p %>% dplyr::select(-grupo)
  x <- as.matrix(x)

  # Get dimensions
  n <- nrow(x)
  p <- ncol(x)

  # Get number of groups and the levels
  g <- as.factor(train.p$grupo)
  lev <- lev1 <- levels(g)

  # Get the number of images per class
  counts <- as.vector(table(g))
  proportions <- counts/n
  ng <- length(proportions)
  names(prior) <- names(counts) <- lev1

  # Group means and total mean
  group.means <- tapply(c(x), list(rep(g, p), col(x)), mean)
  all.means <- as.matrix(colMeans(x)) %>% t()

  # Scatter Matrices
  #Intra class
  B <- t(x - group.means[g, ]) %*%(x - group.means[g, ])
  # Between class
  A <- (t(group.means)-matrix(rep(all.means,10),ncol=10))%*%
    diag(counts) %*%
    t(t(group.means)-matrix(rep(all.means,10),ncol=10))
```

```
# Initialization
rhoAnt = -1000000
iter=1
dimension = espacio.proy
# rho = (cota.inf + cota.sup)/2
rho = 0
frho = sum(eigen(A-rho*B)$values[1:dimension])
V = eigen(A-rhoAnt*B)$vectors[,(1:dimension)]

# list of results
a=list()
a[[1]] <- list(iter = 1, rho = rho, frho = frho, V = V)

# Iterative method (use of efficient irlba method)
while(iter < 50 & abs(rho-rhoAnt)>tol){
  print(paste("iteracion: ",as.character(iter)))
  rhoAnt = rho
  iter=iter+1
  V = eigen(A-rhoAnt*B)$vectors[,(1:dimension)] # Lanczos
  rho = sum(diag(t(V) %*%A %*%V)) /
  sum(diag(t(V) %*%B %*%V))
  frho = sum(eigen(A-rho*B)$values[1:dimension])
  a[[iter]] <- list(iter = iter, rho = rho,
                    frho = frho, V = V)
}

# Getting the projection matrix
scaling <- V
dimnames(group.means)[[2L]] <- colnames(x)
cl <- match.call()
cl[[1L]] <- as.name("lda.iter")
structure(list(iteraciones = a,
               prior = prior,
               counts = counts,
               means = group.means,
               scaling = scaling,
```

```
                    lev = lev,
                    N = n,
                    call = cl,
                    class = "lda.iter"))
}

# 2) Fisher para pruebas ----------------------------------
lda.iter <- function (test.p,
                      train.p,
                      espacio.proy,
                      prior = proportions,
                      tol = 1e-10) {

  # Prepare matrix of data
  x <- train.p %>% dplyr::select(-grupo) %>% as.matrix()

  # Get dimensions
  n <- nrow(x)
  p <- ncol(x)

  # Get number of groups and the levels
  g <- as.factor(train.p$grupo)
  lev <- lev1 <- levels(g)

  # Get the number of images per class
  counts <- as.vector(table(g))
  proportions <- counts/n
  ng <- length(proportions)
  names(prior) <- names(counts) <- lev1

  # Group means and total mean
  group.means <- tapply(c(x), list(rep(g, p), col(x)), mean)
  all.means <- as.matrix(colMeans(x)) %>% t()

  # Scatter Matrices
  #Intra class
  B <- t(x - group.means[g, ]) %*%(x - group.means[g, ])
```

```
  # Between class
  A <- (t(group.means)-matrix(rep(all.means,10),ncol=10))%*%
    diag(counts) %*%
    t(t(group.means)-matrix(rep(all.means,10),ncol=10))

  # [not run] Intra class + Between class = Total variance
  diag(A+B)
  diag(49*(as.matrix(x) %%cov))

  # [not run] EigenValues Time
  (eigen(B))$values
  (eigen(A))$values

  # [not run] Boundaries
  cota.inf <- sum((eigen(A))$values[1:espacio.proy])/
    sum((eigen(B))$values[1:espacio.proy])
  cota.sup <- sum((eigen(A))$values[1:espacio.proy])/
sum((eigen(B))$values[(dim(A)[1]+1-espacio.proy):dim(A)[1]])

  # Initialization
  rhoAnt = -1000000
  iter=1
  dimension = espacio.proy
  rho = (cota.inf + cota.sup)/2

  frho = sum(eigen(A-rho*B)$values[1:dimension])
  V = eigen(A-rhoAnt*B)$vectors[,(1:dimension)]

  # list of results
  a=list()
  a[[1]] <- list(iter = 1, rho = rho, frho = frho, V = V)

  # Iterative method (use of efficient irlba method)
  while(iter < 50 & abs(rho-rhoAnt)>tol){
    print(paste("iteracion: ",as.character(iter)))
    rhoAnt = rho
    iter=iter+1
```

```
    V = eigen(A-rhoAnt*B)$vectors[,(1:dimension)]
rho = sum(diag(t(V) %*% A %*% V)) /sum(diag(t(V) %*% B %*% V))
    frho = sum(eigen(A-rho*B)$values[1:dimension])
a[[iter]] <-list(iter = iter, rho = rho, frho = frho, V = V)
  }

  # Getting the projection matrix
  scaling <- V
  dimnames(group.means)[[2L]] <- colnames(x)
  cl <- match.call()
  cl[[1L]] <- as.name("lda.iter")
  structure(list(iteraciones = a,
                 prior = prior,
                 counts = counts,
                 means = group.means,
                 scaling = scaling,
                 lev = lev,
                 N = n,
                 call = cl,
                 cota.inf = cota.inf,
                 cota.sup = cota.sup,
                 class = "lda.iter"))
}

# 3) LDA original ------------------------------------------
lda.original <- function (x,
                          grouping,
                          prior = proportions,
                          tol = 1e-04,
                          nu = 5) {

  # Prepare matrix of data
  x <- as.matrix(x)
  n <- nrow(x)

  # Get dimensions
  p <- ncol(x)
```

```
g <- as.factor(grouping)

# Get number of groups and the levels
lev <- lev1 <- levels(g)
counts <- as.vector(table(g))

# Get the number of images per class
proportions <- counts/n
ng <- length(proportions)
names(prior) <- names(counts) <- lev1

# Group means
group.means <- tapply(c(x), list(rep(g, p), col(x)), mean)
f1 <- sqrt(diag(var(x - group.means[g, ])))
scaling <- diag(1/f1, , p)

# Calculate and project
fac <- 1/(n - ng)
X <- sqrt(fac) * (x - group.means[g, ]) %*% scaling
X.s <- svd(X, nu = 0L)
rank <- sum(X.s$d > tol)
scaling <- scaling %*% X.s$v[, 1L:rank] %*%
  diag(1/X.s$d[1L:rank], , rank)
xbar <- colSums(prior %*% group.means)
fac <-1/(ng - 1)
X <- sqrt((n * prior) * fac) * scale(group.means,
      center = xbar, scale = FALSE) %*% scaling
X.s <- svd(X, nu = 0L)
rank <- sum(X.s$d > tol * X.s$d[1L])
scaling <- scaling %*% X.s$v[, 1L:rank]

# Export
dimnames(scaling) <- list(colnames(x), paste("LD",
        1L:rank, sep = ""))
dimnames(group.means)[[2L]] <- colnames(x)
cl <- match.call()
cl[[1L]] <- as.name("lda")
```

```
  structure(list(prior = prior, counts = counts,
                 means = group.means,
                 scaling = scaling, lev = lev,
                 svd = X.s$d[1L:rank], N = n,
                 call = cl), class = "lda")
}

# 4) Lanczos Fortran ----------------------------------
# Give r0, A
lanczos.iter <- function(A){
  Dim = dim(A)[2]
  r0 = matrix(rep(0,Dim), ncol = 1)
  for(i in 1:Dim){
    r0[i] = runif(1, 0, 1)
  }
r0 = r0 / sqrt(sum(r0^2))
r = list()
b = list()
a = list()
q = list()

q[[1]] = 0 # q0
r[[1]] = r0 #r0
b[[1]] = sqrt(sum(r[[1]]^2)) # b1
q[[2]] = as.matrix(r[[1]]/b[[1]], ncol =D) #q1
a[[1]] = t(q[[2]]) %*%A %*%q[[2]] #a1
q[[1]]
j <- 2
q = cbind(q[[1]], q[[2]])
while(b[[j-1]] > .0000001 | j < Dim+1){
  z = A %*%q[,j]
  z = z - q %*%(t(q) %*%z)
  z = z - q %*%(t(q) %*%z)
  b[[j]] = sqrt(sum(z^2))
  q.temp = z/b[[j]]
  q <- cbind(q, q.temp)
  a[[j]] = t(q[,j+1]) %*%A %*%q[,j+1]
```

```
  j = j+1
  print(j)
}

dyn.load("/usr/lib/liblapack.dylib")

Jobz <- as.character('V')#Get The eigenval and Eigenvector
N <- as.integer(Dim)#Matrix Dimension
D <- unlist(a)[1:Dim]#Diagonal
E <- unlist(b)[2:(Dim)]#Upper (and lower) diagonal
Z <- array(0.0,N*N)#storage for eigenvectors
Ldz <- N#Leading dimension of Z (in our case number of rows)
Work <- array(0.0,2*N)#Storage for processing
info <- as.integer(0)#Info flag

res<-.Fortran("dstev", Jobz, N, D, E, Z, Ldz, Work, info)

evalues <- res[[3]]
evectors <- matrix(res[[5]],N,N)
list(evalues, evectors)
}

# 5) Fisher eficiente ----------------------------------
lda.iter.fortran<- function (test.p = test.p,
                             train.p = train.p,
                             espacio.proy,
                             prior = proportions,
                             tol = 1e-10) {

  # Prepare matrix of data
  x <- train.p %>% dplyr::select(-grupo)
  x <- as.matrix(x)

  # Get dimensions
  n <- nrow(x)
  p <- ncol(x)
```

```
# Get number of groups and the levels
g <- as.factor(train.p$grupo)
lev <- lev1 <- levels(g)

# Get the number of images per class
counts <- as.vector(table(g))
proportions <- counts/n
ng <- length(proportions)
names(prior) <- names(counts) <- lev1

# Group means and total mean
group.means <- tapply(c(x), list(rep(g, p), col(x)), mean)
all.means <- as.matrix(colMeans(x)) %*% t()

# Scatter Matrices
#Intra class
B <- t(x - group.means[g, ]) %*%(x - group.means[g, ])
# Between class
A <- (t(group.means)-matrix(rep(all.means,10),ncol=10))%*%
  diag(counts) %*%
  t(t(group.means)-matrix(rep(all.means,10),ncol=10))

# Initialization
rhoAnt = -1000000
iter=1
dimension = espacio.proy
# rho = (cota.inf + cota.sup)/2
rho = 0
frho = sum(eigen(A-rho*B)$values[1:dimension])
V = eigen(A-rhoAnt*B)$vectors[,(1:dimension)]

# list of results
a=list()
a[[1]] <- list(iter = 1, rho = rho, frho = frho, V = V)

# Iterative method (use of efficient irlba method)
while(iter < 50 & abs(rho-rhoAnt)>tol){
```

```
    print(paste("iteracion: ",as.character(iter)))
    rhoAnt = rho
    iter=iter+1

  V = lanczos.iter(A-rhoAnt*B)[[2]][,(1:dimension)]#Lanczos
    rho = sum(diag(t(V) %*%A %*%V)) /
      sum(diag(t(V) %*%B %*%V))
    frho = sum(lanczos.iter(A-rhoAnt*B)[[1]][1:dimension])
    a[[iter]] <- list(iter = iter, rho = rho,
                      frho = frho, V = V)
  }

  # Getting the projection matrix
  scaling <- V
  dimnames(group.means)[[2L]] <- colnames(x)
  cl <- match.call()
  cl[[1L]] <- as.name("lda.iter")
  structure(list(iteraciones = a,
                 prior = prior,
                 counts = counts,
                 means = group.means,
                 scaling = scaling,
                 lev = lev,
                 N = n,
                 call = cl,
                 class = "lda.iter"))
}
```

B.2.2. *functions_comparison*

```
comparacion.modelos <- function(train.p, test.p,
                                dataset,numComp,
                                fortran = FALSE){
  # lda iterativo
  print(paste("==Corriendo el modelo con la base",dataset))

  lda.iter <- sapply(1:14, function(x){
    print(paste("Iteracion con: ",
```

```
                numComp[x],
                "componentes."))

  # Run model with different number of components
  if(fortran == FALSE){
  modelo.iterativo <- lda.iter.ef(train.p = train.p,
                                  test.p = test.p,
                                  espacio.proy = numComp[x])
  }else{
    modelo.iterativo <- lda.iter.fortran(train.p = train.p,
                                  test.p = test.p,
                                  espacio.proy = numComp[x])
  }
  # Data.frame with information of  (\rho and f(\rho))
  iter <- sapply(1:length(modelo.iterativo$iteraciones),
                 function(x){
        data.frame(modelo.iterativo$iteraciones[[x]]$rho,
                   modelo.iterativo$iteraciones[[x]]$frho)%>%
                setNames(c("rho", "frho")) %>% data.frame()
                  }) %>% t() %>% data.frame() %>%
    mutate(iter = 1:nrow(.),
           rho = unlist(rho),
           frho = unlist(frho))
  iter <- iter %>% data.frame() %>%
    dplyr::mutate(frho2 = c(iter$frho[2:nrow(.)],NA ),
                  rho2 = c(iter$rho[2:nrow(.)], NA))

  # Projected data into a 20-dimensional space
  V <- modelo.iterativo$iteraciones[[
    length(modelo.iterativo$iteraciones)]]$V

  train.proyectados <- data.frame(as.matrix(train.p %>%
                          dplyr::select(-grupo)) %*%V) %>%
    mutate(grupo = train.p$grupo)

  test.proyectados <- data.frame(as.matrix(test.p %>%
                          dplyr::select(-grupo)) %*%V) %>%
```

```
    mutate(grupo = test.p$grupo)

  # K-nn model

  print("Iniciando modelo knn")
modelo <- knn(train = train.proyectados[,c(1:numComp[x])],
              test = test.proyectados[, c(1:numComp[x])],
                cl = train.proyectados$grupo, k = 3)
  pred <- data.frame(original = test.proyectados$grupo,
                     predicho = modelo)

  pred <- pred %>% mutate(verdad = original != predicho)
  sum(pred$verdad)/nrow(pred)
})

modelo.logistico <- lapply(1:14, function(x){
  modelo <- multinom(grupo~.,
        data = train.p[,c(1:numComp[x], dim(train.p)[2])],
                     MaxNWts = 10000)
  predichos <- predict(modelo,
                       newdata = test.p[,c(1:numComp[x])])
  pred <- data.frame(original = test.p$grupo,
                     predicho = predichos)
  pred <- pred %>% mutate(verdad = original != predicho)
  sum(pred$verdad)/nrow(pred)
})

lda.original <- lapply(1:14, function(x){
  modelo<- lda(grupo ~ .,
               train.p[,c(1:numComp[x],
                          dim(train.p)[2])],
               prior = c(1,1,1,1,1,1,1,1,1,1)/10)
predichos <- predict(modelo, test.p[,1:numComp[x]])$class
  pred <- data.frame(original = test.p$grupo,
                     predicho = predichos)
  pred <- pred %>% mutate(verdad = original != predicho)
  sum(pred$verdad)/nrow(pred)
```

```
  })

  data.frame(lda.iter = unlist(lda.iter),
             logistico = unlist(modelo.logistico),
             lda.original = unlist(lda.original))
}

graficar.comparacion <- function(datos, numComp, label){
  plot1 <- datos %>%
    #write.table(sep = ';', pipe('pbcopy'), row.names= F)
    mutate(numComp = numComp) %>%
    gather(variable, value, lda.iter:lda.original) %>%
    ggplot(aes(x = numComp , y = 1-value,
      group = factor(variable), color = factor(variable))) +
    geom_point() + geom_line()+
    labs(title = "Tasa de reconocimiento conforme
          numero el espacio de proyeccion",
         x = "numero de entrenamiento por grupo",
         y="tasa reconocimiento")

  png(filename =paste(getwd(), "/graphs/Chapter4_comp_",
        label, ".png", sep =""), width = 600,  height = 400)
  grid.arrange(plot1, ncol =1)
  dev.off()
}
```

B.2.3. *functions_prueba20comp*

```
# funciones prueba 20 componentes

plot.comp <- function(V1, label, train.p, test.p){
  train.proyectados <- data.frame(as.matrix(train.p %>%
                            dplyr::select(-grupo)) %*%V1) %>%
    mutate(grupo = train.p$grupo)

  test.proyectados <- data.frame(as.matrix(test.p %>%
                            dplyr::select(-grupo)) %*%V1) %>%
    mutate(grupo = test.p$grupo)
```

```
  a1 = ggplot(train.proyectados) +
    geom_point(aes(x=X1, y=X2,
                   group = grupo, color = grupo),
               size=5, alpha = .2) +
    geom_point(aes(x=X1, y=X2,
                   group = grupo, color = grupo))+
    labs(title = label)
  a2 = ggplot(train.proyectados) +
    geom_point(aes(x=X3, y=X4,
                   group = grupo, color = grupo),
               size=5, alpha = .2) +
    geom_point(aes(x=X3, y=X4,
                   group = grupo, color = grupo))+
    labs(title = label)
  list(a1,a2)
}
plot.ult <- function(V1, label, train.p, test.p){
  train.proyectados <- data.frame(as.matrix(train.p %>%
                          dplyr::select(-grupo)) %*%V1) %>%
    mutate(grupo = train.p$grupo)

  test.proyectados <- data.frame(as.matrix(test.p %>%
                          dplyr::select(-grupo)) %*%V1) %>%
    mutate(grupo = test.p$grupo)
  a7 =  ggplot(train.proyectados) +
    geom_point(aes(x=X17, y=X18,
                   group = grupo, color = grupo),
               size=5, alpha = .2) +
    geom_point(aes(x=X17, y=X18,
                   group = grupo, color = grupo))+
    labs(title = label)

  a8 =  ggplot(train.proyectados) +
    geom_point(aes(x=X19, y=X20,
                   group = grupo, color = grupo),
               size=5, alpha = .2) +
```

```
    geom_point(aes(x=X19, y=X20,
                   group = grupo, color = grupo))+
    labs(title = label)
  list(a7,a8)
}
prueba.20comp <- function(train.p, test.p, dataset){
  # Iterative LDA, projected to a 20-dimentional space
  modelo.iterativo <- lda.iter(train.p = train.p,
                               test.p = test.p,
                               espacio.proy = 20)
  # Data.frame with information of (\rho and f(\rho))
  iter <- sapply(1:length(modelo.iterativo$iteraciones),
                 function(x){
          data.frame(modelo.iterativo$iteraciones[[x]]$rho,
                modelo.iterativo$iteraciones[[x]]$frho)  %>%
                     setNames(c("rho",
                                "frho")) %>% data.frame()}) %>%
    t() %>% data.frame() %>%
    mutate(iter = 1:nrow(.),
           rho = unlist(rho),
           frho = unlist(frho))
  iter <- iter %>%
    dplyr::mutate(frho2 = c(iter$frho[2:nrow(.)],NA),
                  rho2 = c(iter$rho[2:nrow(.)], NA))

  # f(\rho) graph. It contains the iterations of the algorit
  plot <- iter %>% ggplot() +
    geom_point(aes(x = rho, y = frho)) +
    geom_segment(aes(x = rho, y = frho,
                     xend = rho, yend = frho2),
                 color = "gray50") +
    geom_segment(aes(x = rho, y = frho2,
                     xend = rho2, yend = frho2),
                 arrow = arrow(length = unit(.3, "cm")),
                 size = 1,
                 color = "gray50") +
    labs(title = paste("Valor de la funcion f(rho) con
```

```
      respecto al numero de iteraciones ", dataset, sep =""),
          ylab = "f(rho)",
          xlab = "rho")
  # Save the image into a *.png file
  png(paste(getwd(), "/graphs/Chapter4_iteraciones_",
      dataset, ".png", sep =""), width = 500, height = 500)
  grid.arrange(plot, ncol = 1)
  dev.off()

  # Projected data into a 20-dimensional space
  V <- modelo.iterativo$iteraciones[[
    length(modelo.iterativo$iteraciones)]]$V

  r1 <- 1
  r2 <- round(length(modelo.iterativo$iteraciones)/3*1)
  r3 <- round(length(modelo.iterativo$iteraciones))

  # iteration1
  V1 <- modelo.iterativo$iteraciones[[r1]]$V
  plot1 <- plot.comp(V1, paste("Iteracion ",r1, sep=""),
                     train.p, test.p)

  # iteracion3
  V2 <- modelo.iterativo$iteraciones[[r2]]$V
  plot2 <- plot.comp(V2, paste("Iteracion ",r2, sep=""),
                     train.p, test.p)

  # iteracion9
  V3 <- modelo.iterativo$iteraciones[[r3]]$V
  plot3 <- plot.comp(V3, paste("Iteracion ",r3, sep=""),
                     train.p, test.p)

  png(paste(getwd(),"/graphs/Chapter4_ejemplo20componentes_",
        dataset, ".png", sep =""), width = 700, height = 800)
  grid.arrange(plot1[[1]], plot1[[2]], plot2[[1]],
               plot2[[2]], plot3[[1]], plot3[[2]],
               ncol = 2)
```

```
  dev.off()

  # last components
  V4 <- modelo.iterativo$iteraciones[[r3]]$V
  plot4 <- plot.ult(V4, "Ultimas componentes",
                    train.p, test.p)
  png(paste(getwd(),"/graphs/Chapter4_ultimasComponentes_",
        dataset, ".png", sep =""),height = 400, width = 700)
  grid.arrange(plot4[[1]],plot4[[2]], ncol = 2)
  dev.off()
  # knn model ------------------------------------------------
  V <- modelo.iterativo$iteraciones[[
    length(modelo.iterativo$iteraciones)]]$V

  train.proyectados <- data.frame(as.matrix(train.p %%
                            dplyr::select(-grupo)) %*%V) %%
    mutate(grupo = train.p$grupo)

  test.proyectados <- data.frame(as.matrix(test.p %%
                            dplyr::select(-grupo)) %*%V) %%
    mutate(grupo = test.p$grupo)

  train.proyectados %% head
  knn.model <- knn(train = train.proyectados[,c(1:20)],
                   test = test.proyectados[, c(1:20)],
                   cl = train.proyectados$grupo, k = 3)

  pp <- data.frame(original = test.proyectados$grupo,
                   predicho = knn.model)

  pp <- pp %% mutate(verdad = original != predicho)

  error <- sum(pp$verdad)/nrow(pp)
  list(modelo.iterativo=modelo.iterativo,iter = iter, V = V,
       error = error, knn.model = knn.model)
}
prolifiling.model <- function(train.p, test.p,
```

```
                                        dataset,numComp){

print(paste("====Realizando profiling de la base",dataset))
lda.time <- lapply(1:14, function(x){
  print(paste("Iteracion con:",
              numComp[x], "componentes."))

  lda.iter <- sapply(1:5, function(y){
    system.time(modelo.iterativo <-
                  lda.iter.ef(train.p = train.p,
                              test.p = test.p,
                              espacio.proy = numComp[x]))
  }) %>% data.frame() %>% t() %>% data.frame() %>%
    dplyr::select(elapsed) %>%
    dplyr::summarise(mean = mean(elapsed))

  logistico <- sapply(1:5, function(y){
    system.time(modelo <- multinom(grupo~.,
          data = train.p[,c(1:numComp[x], dim(train.p)[2])],
                                   MaxNWts = 10000))
  }) %>% data.frame() %>% t() %>% data.frame() %>%
    dplyr::select(elapsed) %>%
    dplyr::summarise(mean = mean(elapsed))

  lda.original <- sapply(1:5, function(y){
    system.time(modelo <- lda(grupo ~ .,
                train.p[,c(1:numComp[x], dim(train.p)[2])],
                          prior = c(1,1,1,1,1,1,1,1,1,1)/10))
  }) %>% data.frame() %>% t() %>% data.frame() %>%
    dplyr::select(elapsed) %>%
    dplyr::summarise(mean = mean(elapsed))

  list(lda.iter, logistico, lda.original)
})

data.frame(unlist(lda.time)) %>%
  mutate(var = rep(c("lda.iter", "logis", "lda.orig"), 14),
```

```
        ID = rep(1:(nrow(.)/3), each = 3)) %>%
  setNames(c("time", "var", "ID")) %>%
  spread(var, time)
}


graficar.profiling <- function(times.profiling, numComp,
                               dataset){
plot1 <- times.profiling %>%
  mutate(numComp = numComp) %>%
  gather(variable, value, lda.iter:logis) %>%
  ggplot(aes(x = numComp , y = value,
             group = variable, color = variable)) +
  geom_point() + geom_line()+
  labs(title = "Tiempo de los modelos",
       x = "Espacio de proyeccion",
       y="Tiempo en (s)")
png(filename = paste(getwd(), "/graphs/Chapter4_profiling",
      dataset, ".png", sep =""), width = 600, height = 400)
grid.arrange(plot1, ncol = 1)
dev.off()
}
```

B.3. Preprocesamiento

B.3.1. *MNIST_munge*

```
# 1) Load libraries ------------------------------------------
set.seed(12345)
library(plyr) # Tools for split, applying and combine data
library(dplyr) # package of grammar of data manipulation
library(tidyr) # package to efficiently "tidy" data
library(FactoMineR) # Package for multivariate analysis
library(ggplot2) # Graphing package "Grammar of Graphics"
library(gridExtra) # Package to work with grid graphics

# 1) Generate train and test from MNIST (run once) -----------
```

```
# darch::readMNIST("data/MNIST/")

# 2) Read MNIST data ------------------------------------------
# Original datasets
load("data/MNIST/train.RData")
load("data/MNIST/test.RData")

# Matrix with images in the rows and the variable in columns
numeros.ind.pix = rbind(data.frame(trainData,
                    grupo = factor(apply(trainLabels[,], 1,
                        function(x){ which(x == 1)-1 }))),
                    data.frame(testData,
                        grupo = factor(apply(testLabels[,], 1,
                        function(x){ which(x == 1)-1 })))) %>%
  mutate(ID = 1:nrow(.))
base.mnist <- numeros.ind.pix
cache("base.mnist")
rm(testData, testLabels, trainData, trainLabels)

# 3) Train and test sets --------------------------------------
# Size of images per class
n.train <- 400

# Sample $n.train$ imager per class
train.id <- numeros.ind.pix %>%
  group_by(grupo) %>%
  sample_n(n.train) %>%
  data.frame()

# Train and Test data.frames
train <- numeros.ind.pix[train.id$ID, ]
test <- numeros.ind.pix[-train.id$ID, ]

# Percentaje of images in train dataset
nrow(train)/(nrow(train)+nrow(test))

# 4) PCA ------------------------------------------------------
```

```
# PCA of MNIST dataset (each row is an image)
# 193 colums contains ~95% of the total variance
system.time(pca.train <- PCA(train %>%
                               dplyr::select(-grupo, -ID),
                             graph = F,
                             scale.unit = F,
                             ncp = 193))

# Matrix of loadings: 784 rows, 193 columns
loadings <- sweep(pca.train$var$coord,2,  # 784 x 193
        sqrt(pca.train$eig[1:ncol(pca.train$var$coord),1]),
        FUN="/")

# Principal components
train.p <- pca.train$ind$coord %>% data.frame() %>%
  dplyr::mutate(grupo = train$grupo)

# [Revision] reconstuyendo primer componente
as.matrix(scale(train %>% dplyr::select(-grupo, -ID),
                center = apply(train %>%
                      dplyr::select(-grupo, -ID), 2,mean),
                scale = F)) %*%loadings[,1] %>% head

pca.train$ind$coord[,1] %>% head

# Projection of the test dataset with the train loadings
# (centered with mean of train dataset)
system.time(test.p <- as.matrix(scale(test %>%
            dplyr::select(-grupo, -ID), # 69,000 x 193

            center = apply(train %>%# centers of train
                    dplyr::select(-grupo, -ID), 2,mean),
                         scale = F)) %*%loadings %>%
              data.frame() %>%
              mutate(grupo = test$grupo))
dim(test.p)
dim(train.p)
```

```
MNIST.test <- test.p
MNIST.train <- train.p
cache("MNIST.test")
cache("MNIST.train")
```

B.3.2. *JAFFE_munge*

```
# 1) Load libraries ------------------------------------------------
set.seed(12345)
library(tiff) # package that reads *.tiff images
library(tidyr) # package to efficiently "tidy" data
library(dplyr) # package of grammar of data manipulation
library(ProjectTemplate) # Template data analysis project
library(FactoMineR) # Package for multivariate analysis

# 2) Read databases ------------------------------------------------
# Get the names of the files
files <- (Sys.glob("data/Jaffe/*.tiff"))

# Create an array that contains all the images paths
nombres.personas <- apply(as.matrix(files),1,function(x){
  gsub("data/Jaffe/","",x)
})

# Data.frame with info about each image (ID, name, class)
guia.personas.df <- data.frame(nombres.personas,
                               nombres.personas) %>%
  separate(nombres.personas.1, c("persona",
                                 "postura",
                                 "ID",
                                 "extension")) %>%
  dplyr::select(-extension,
                -ID) %>%
  mutate(ID = 1:213,
         nombres.personas = as.character(nombres.personas),
         persona = factor(persona),
         postura = factor(postura))
```

```
# List of length 213 that contains an image in each element
lista.completa.imagenes <- lapply(files, function(x){
  imagen <- readTIFF(x)
  if(length(dim(imagen)) == 3){
    matrix <- as.vector(imagen[,,1])
  }else{
    matrix <- as.vector(imagen)
  }
})

# Data.frame, each column is an image
base.imagenes <- data.frame(lista.completa.imagenes)
names(base.imagenes) <- guia.personas.df$nombres.personas

# 2 matrices: In base.pix.ind.1, the columns are the images
#             In base.ind.pix.1, the rows are the images
base.pix.ind.1 <- data.frame(base.imagenes)
base.ind.pix.1 <- data.frame(t(base.imagenes)) %>%
  mutate(grupo = names(base.imagenes)) %>%
  separate(grupo, c("grupo",
                    "postura",
                    "ID",
                    "extension")) %>%
  dplyr::select(-postura,
                -ID,
                -extension) %>%
  mutate(ID = 1:nrow(.))

# 3) Explorar bases de datos ------------------------------
# sizes of the matrices
dim(base.pix.ind.1) # 65,536 rows and 213 columns
dim(base.ind.pix.1) # 213 rows and 65,536 columns

# save in cache the data.base
base.jaffe <- base.ind.pix.1
cache("base.jaffe")
```

```
# 4) Train and test ------------------------------------------
# Size of images per class
n.train <- 5

# Sample $n.train$ imager per class
train.id <- base.jaffe %>%
  group_by(grupo) %>%
  sample_n(n.train) %>%
  data.frame()

# Train and Test data.frames
train <- base.jaffe[train.id$ID, ]
test <- base.jaffe[-train.id$ID, ]

# Percentaje of images in train dataset
nrow(train)/(nrow(train)+nrow(test))

# 5) PCA -----------------------------------------------------

# PCA of JAFFE dataset (each row is an image)
# 52 colums contains ~95% of the total variance
system.time(pca.train <- PCA(train %>%
                               dplyr::select(-grupo, -ID),
                             graph = F,
                             scale.unit = F,
                             ncp = 49))

# Matrix of loadings: 65,536 rows, 49 columns
loadings <- sweep(pca.train$var$coord,2,
        sqrt(pca.train$eig[1:ncol(pca.train$var$coord),1]),
        FUN="/")

# Principal components
train.p <- pca.train$ind$coord %>% data.frame() %>%
  dplyr::mutate(grupo = train$grupo)

# [Revision] Reconstuyendo primer componente
```

```
as.matrix(scale(train %>% dplyr::select(-grupo, -ID),
                center = apply(train %>%
                          dplyr::select(-grupo, -ID), 2,mean),
                scale = F)) %*% loadings[,1] %>% head

pca.train$ind$coord[,1] %>% head

# Projection of the test dataset with the train loadings
# (centered with mean of train dataset)
system.time(test.p <- as.matrix(scale(test %>%
                     dplyr::select(-grupo, -ID),
                     center = apply(train %>%
                        dplyr::select(-grupo, -ID), 2,mean),
                     scale = F)) %*% loadings %>%
                data.frame() %>%
                mutate(grupo = test$grupo))

JAFFE.test <- test.p
JAFFE.train <- train.p
cache("JAFFE.test")
cache("JAFFE.train")

rm(list = ls())
```

Bibliografía

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] J.W. Demmel. *Applied Numerical Linear Algebra*. Miscellaneous Bks. Society for Industrial and Applied Mathematics, 1997.

[3] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[4] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[5] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic press, 2013.

[6] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[7] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[8] Tom Michael Mitchell. *The discipline of machine learning*. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

[9] Thanh T Ngo, Mohammed Bellalij, and Yousef Saad. The trace ratio optimization problem. *SIAM review*, 54(3):545–569, 2012.

[10] Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.