Universidad Nacional de Colombia
Sede Manizales

Regularización y Métodos Kernel para Algoritmos de Clasificación

TESIS DE MAESTRÍA

Elaborada por
Juliana Ramirez Candamil

Carlos Daniel Acosta Medina
Director

Regularización y Métodos Kernel para Algoritmos de Clasificación

Tesis elaborada por

Juliana Ramirez Candamil

Presentada a la Facultad de Ciencias Exactas y Naturales
de la Universidad Nacional de Colombia - Sede Manizales
como Requisito Parcial para
obtener el Grado de
Magíster en Ciencias – Matemática Aplicada

Aprobada por el
Comité Evaluador:

_____
Carlos Daniel Acosta Medina, Director de Tesis

_____
Genaro Daza Santacoloma, Jurado de Tesis

_____
Mauricio Orozco Alzate, Jurado de Tesis

_____
, Jurado de Tesis

Departamento de Matemáticas y Estadística
Universidad Nacional de Colombia - Sede Manizales

16 de junio de 2010

Dedicatoria

A mis maravillosos padres, inspiración interminable.

Prefacio

El trabajo presentado a continuación es el resultado del estudio de la técnicas de Regularización y Métodos Kernel empleados para algoritmos de clasificación. Se eligió el método de Mínimos Cuadrados cómo clasificador, debido a que RLSC requiere la solución de un único problema de ecuaciones lineales, lo cual presenta ventajas en términos computacionales, eso hace que sea un método sencillo y existoso para realizar esta importante tarea.

La escencia del trabajo se enfoca básicamente en automatizar la selección de parámetros, tanto de regularización, como del kernel, empleando validación cruzada generalizada GCV.

Para regularizar el método, se empleó la técnica de Regularización de Tikhonov y el denominado "Kernel Trick" para llegar a la extensión no lineal del clasificador de Mínimos Cuadrados.

En ambos casos, es decir, versión lineal y no lineal de los mínimos cuadrados, se obtiene la función GCV correspondiente y se prueba la selección de parámetros evaluando la exactitud de la clasificación, y comparando ésta con otros métodos bastante empleados en la literatura para la tarea de clasificación.

Se realizan pruebas concluyentes, que determinan la eficacia del método elegido, especialmente en su versión no lineal. Se presenta el análisis por medio de tablas que muestran la medida de exactitud y se realizan gráficos de espacios ROC y de líneas de exactitud, que permiten visualizar el desempeño de los diferentes métodos.

Agradecimientos

A mis profesores, que con el tiempo se convierten en amigos. En especial a Carlos Daniel Acosta Medina y Jorge Eduardo Hurtado Gómez, por su apoyo, colaboración y confianza.

A mi familia, por su dedicación incansable, apoyo reiterado y fuente de inspiración.

A mis compañeros de estudio y amigos, quienes siempre tuvieron una voz de aliento.

A Jaime Enrique Arango Castro, quien fué fundamental a lo largo del proceso, por su apoyo, confianza y entrega.

A la Universidad Nacional de Colombia Sede Manizales, por permitirme continuar desarrollando esta parte tan importante de mi vida, en sus espacios y con su gente excepcional.

Índice general

Índice de tablas

Índice de figuras

1 Preliminares

1.1. Métodos Kernel

Uno de los principales problemas en la teoría de aprendizaje es el siguiente: suponga que se tiene dos clases de objetos y se encuentra un nuevo objeto que debe ser asignado a alguna de las dos clases, i.e. suponga que se tiene un conjunto de datos $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \in X \times \{\pm 1\}$, donde X es un conjunto de datos llamados *patrones* o entradas, y las y_i son llamadas *etiquetas* o salidas. Esta es un situación particular que se denomina *clasificación binaria* [25].

El problema de la clasificación es uno de los primeros que aparece en la actividad científica y constituye un proceso consustancial con casi cualquier actividad humana, de tal manera que en la resolución de problemas y en la toma de decisiones, la primera parte de la tarea consiste precisamente en clasificar el problema o la situación, para después aplicar la metodología correspondiente y que en buena medida dependerá de esa clasificación.

Los procedimientos de clasificación permiten establecer las relaciones subyacentes en los datos obtenidos durante la experimentación y mediciones en distintas áreas del conocimiento. Esto gracias a que consideran tanto las relaciones estructurales como la naturaleza estocástica de las variables y las imprecisiones en las mediciones de las cuales los datos son resultado. Esto ha permitido su gran impacto en áreas como: procesamiento de señales, química, bioquímica, matemáticas, ingeniería de sistemas, física, estadística, etc [14].

Para iniciar el estudio de los problemas de clasificación, se hace necesario emplear un tipo adicional de estructura, que permita generalizar la clasificación para datos desconocidos, en el caso de reconocimiento de patrones esto significa que, dado un nuevo punto $x \in X$, se pretende predecir el $y \in \{\pm 1\}$ correspondiente [25]. Para lograr este objetivo, se debe formular o escoger una medida de similitud en los datos de entrada. Considere una medida de similitud de la forma:

$$\begin{aligned} k : X \times X &\longrightarrow \mathbb{R} \\ : (x, x') &\longmapsto k(x, x') \end{aligned} \tag{1.1}$$

Esto es, una función que, dados dos patrones x y x', retorna un número real que caracteriza su similitud. Además, se asume que es una función simétrica. Esta función se denomina **kernel.** Un tipo simple de medida de similitud es el denominado *producto punto:*

Definición 1. Dados dos vectores $(x, x') \in \mathbb{R}^n$, el producto punto canónico [25], viene definido por

$$\langle x, x' \rangle = k(x, x') = \sum_{i=1}^{n} (x)_i (x')_i \tag{1.2}$$

donde $(x)_i$ denota la i-ésima entrada de x.

Para asegurar el uso del producto punto como una medida de similitud, es necesario representar los patrones como vectores en algún espacio H con producto punto (el cual no necesariamente coincide con $\mathbb{R}^n$). Para esto se emplea la siguiente aplicación:

$$\begin{aligned} \Phi : X &\longrightarrow H \\ : x &\longmapsto \Phi(x) \end{aligned} \tag{1.3}$$

Φ es generalmente un mapeo no lineal. El espacio H es llamado *espacio de características* [25] . El mapeo de los datos en H a través de Φ tiene tres beneficios, ver [23]:

1. Se puede definir una medida de similitud del producto punto en H

$$k(x, x') = \langle x, x' \rangle = \langle \Phi(x), \Phi(x') \rangle \tag{1.4}$$

2. Permite hacer frente a los patrones geométricos, por lo tanto, es posible estudiar los algoritmos de aprendizaje utilizando álgebra lineal y geometría analítica.

3. La libertad para la escogencia del mapeo Φ permite diseñar una gran variedad de medidas de similitud y algoritmos de aprendizaje.

Se puede calcular el valor del producto punto en H sin tener explícitamente calculado el mapeo Φ. Por ejemplo, cuando k es un caso de características polinomiales. Suponga $n = d = 2$.

$$\begin{array}{cccc} \Phi & \mathbb{R}^2 & \longrightarrow & \mathbb{R}^3 \\ (x_1, x_2) & & \longmapsto & (z_1, z_2, z_3) := \left(x_1^2,\ \sqrt{2}x_1x_2, x_2^2\right) \end{array}$$

De manera general, este kernel se representa como $k(x, x') = \langle x, x' \rangle^d$ donde d representa el grado del polinomio.

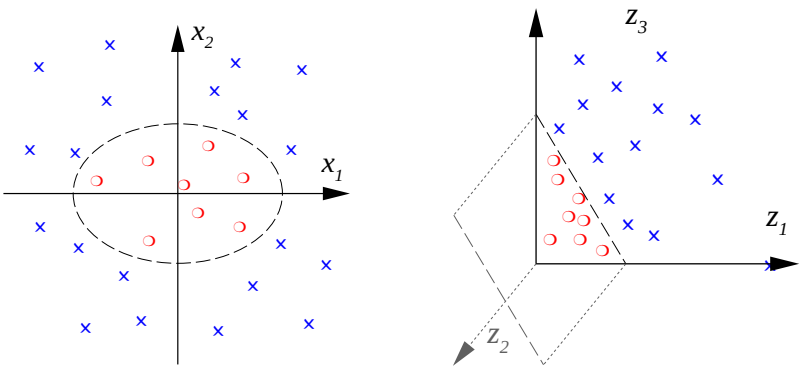


Figura 1.1. Kernel polinomial [25].

1.1.1. Kernel definido positivo

Definición 2. (*Matriz de Gram*) Dada una función $k : X^2 \longrightarrow \mathbb{k}$ (donde $\mathbb{k} = \mathbb{C}$ o $\mathbb{k} = \mathbb{R}$) y patrones $x_1, x_2, ..., x_m \in X$, entonces la matriz K de $m \times m$ con elementos $K_{ij} := k(x_i, x_j)$ se llama la Matriz de Gram (o Matriz Kernel) de k con respecto a $x_1, x_2, ..., x_m$

Definición 3. (*Matriz definida positiva*) Una matriz compleja K de $m \times m$ que satisface:

$$\sum_{i,j} c_i \bar{c}_j K_{ij} \geq 0, \tag{1.5}$$

para todo $c_i \in \mathbb{C}$ se llama definida positiva [25]. Similarmente, una matriz real K simétrica de $m \times m$ que satisface la ecuación anterior para todo $c_i \in \mathbb{R}$ se llama definida positiva.

Cabe anotar que una matriz simétrica es definida positiva si y sólo si todos sus valores propios son no negativos [18].

1.1.2. La Función reproductora de Kernel

Asumiendo que k es un kernel definido positivo de valores reales y X es un subconjunto no vacío. Se define una función de X entre el espacio de funciones mapeadas de X en $\mathbb{R}$, denotadas como $\mathbb{R}^X : \{f : X \longrightarrow \mathbb{R}\}$, via

$$\begin{aligned} \Phi : X &\longrightarrow \mathbb{R} \\ : x &\longmapsto k(\cdot, x) \end{aligned} \tag{1.6}$$

Aquí Φ denota la función que asigna los valores de $k(x', x)$ a $x' \in X$ i.e. $\Phi(x)(\cdot) = k(\cdot, x)$, como se muestra en la figura 1.2.

Por lo tanto, cada patrón es transformado en una función sobre el dominio X. En este sentido, un patrón ahora es representado por su similitud con todos los otro puntos en el dominio de entrada X,

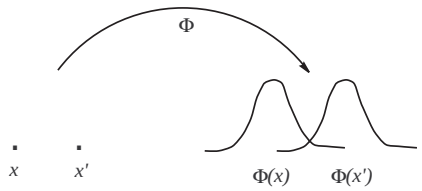


Figura 1.2. Función Φ [25].

Esto parece ser una representación muy rica, no obstante, a su vez el kernel permite el cálculo de un producto interior en esta representación. Para construir un espacio asociado a Φ, se siguen los siguientes pasos, ver [25]:

1. Convertir la imagen de Φ en un espacio vectorial.
2. Definir un producto punto, lo cual es una forma bilineal definida positiva estricta.
3. Mostrar que el producto punto satisface $k(x', x) = \langle \Phi(x), \Phi(x') \rangle$

1.1.3. Truco Kernel

Dado un algoritmo, el cual puede ser formulado en terminos de un kernel definido positivo k, se puede construir un algoritmo alternativo, reemplazando k por otro kernel definido positivo $\widetilde{k}$. ver [25] y [23].

$$\langle \Phi(x), \Phi(x') \rangle_H = k(x, x')_{\mathbb{R}^n}$$

En otras palabras, el *Truco Kernel* es un método para emplear un algoritmo de clasificación lineal para resolver un problema no lineal, por el mapeo de las observaciones no lineales originales en un espacio de alta dimensionalidad, donde el clasificador lineal es posteriormente empleado; esto hace una clasificación lineal en el nuevo espacio, equivalente a la clasificación no lineal en el espacio original (Tomado de Wikipedia http://en.wikipedia.org).

1.1.4. Espacio de Hilbert Reproductor del Kernel (RKHS)

Definición 4. (*Espacio de Hilbert Reproductor del Kernel*) Sea X un conjunto no vacío (frecuentemente llamado conjunto indexado) y H un espacio de Hilbert de funciones $f : X \longrightarrow \mathbb{R}$. Entonces H es llamado un Espacio de Hilbert Reproductor del Kernel inducido con el producto punto $\langle \cdot, \cdot \rangle$ y la norma $\|f\| = \sqrt{\langle f, f \rangle}$ si existe una función $k : X \times X \longrightarrow \mathbb{R}$ con las siguientes propiedades:

1. k tiene la propiedad de reproducción

$$\langle f, k(x, \cdot)\rangle = f(x) \quad \text{para todo } f \in H \tag{1.7}$$

En particular

$$\langle k(x, \cdot), k(x', \cdot)\rangle = k(x, x') \tag{1.8}$$

2. k genera H, i.e. $H = \overline{span\,\{k(x, \cdot)|x \in X\}}$ donde $\overline{X}$ denota la completitud del conjunto X.

En un nivel más abstracto, un RKHS puede ser definido como un espacio de Hilbert de funciones f de X tales que la evaluación de todos los funcionales (el mapeo $f \longmapsto f(x')$, donde $x' \in X$) son continuos [25]. En tal caso, por el teorema de Representación de Riesz [16], para cada $x' \in X$ existe una única función de x, llamada $k(x, x')$, tal que

$$f(x') = \langle f, k(\cdot, x')\rangle \tag{1.9}$$

1.2. Selección de parámetros

Los problemas mal condicionados se encuentran frecuentemente en la ciencia y la ingeniería, fueron introducidos por Hadamard quien los investigó en física matemática. Un problema se considera *bien definido* si cumple con los requerimientos de existencia, unicidad y estabilidad [12], por lo tanto, será *mal condicionado*, si no se cumple una o más de las condiciones anteriores.

Por ejemplo, un problema mal condicionado de la forma

$$min_x \|Ax - b\|_2 , \tag{1.10}$$

proviene, por ejemplo, de la discretización de una ecuación integral de Freldholm de primer tipo y ocurre en variedad de aplicaciones. En problemas discretos mal condicionados, A es mal condicionada, y con frecuencia no hay vacíos en el espectro de sus valores singulares. Frecuentemente b contiene ruido, debido a la medición y/o error de experimentación. Este ruido, en combinación con el mal condicionamiento de A, significa que la solución exacta de (1.10) tiene poca relación con la solución libre de ruido y no es útil [15].

Los métodos numéricos para resolver problemas discretos mal condicionados han sido presentados en mucho trabajos. Estos métodos se basan en los llamados **métodos de regularización**. El objetivo principal de la regularización es incorporar más información acerca de la solución deseada,

con el fin de estabilizar ese problema y encontrar una solución adecuada y estable [12].

Uno de los métodos de regularización más popular es la regularización de Tikhonov, la cual se introduce en la sección 1.3, donde (1.10), se reemplaza por

$$min_x \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \tag{1.11}$$

Para resolver ese problema se hace necesario entonces la adición de un nuevo parámetro, denotado por λ, al problema original, como ya se había mencionado para lograr una solución útil y estable por lo tanto, es allí donde se encuentra el siguiente inconveniente, la "Selección de Parámetro", es necesario elegir el mejor parámetro de regularización que permita cumplir con los objetivos de existencia, unicidad y estabilidad [26], [22].

Cabe anotar que el problema de selección de parámetro también se presenta cuando se resuelven problemas no lineales de clasificación y debe ser empleada la teoría de kernels para abordarlos, por ejemplo, la matriz A del problema (1.10) puede ser reemplazada por algún tipo de kernel, el cual podría ser el de base radial gaussiana:

$$K(x, x^{'}) = \exp\left(-\frac{\left\|x - x^{'}\right\|^2}{2\sigma^2}\right) \tag{1.12}$$

Como se puede observar en (1.12), se hace presente un nuevo parámetro, denotado por la letra σ, es allí donde también se hace necesario una nueva selección [6].

Para la estimación del mejor parámetro de regularización han sido empleadas diferentes técnicas [26],[4]. En este trabajo se emplearán basicamente dos de ellas, a saber, Validación Cruzada Generalizada GCV, por sus siglas en inglés (Generalized Cross Validation) [10],[11], [17] y Dejar Uno por Fuera Validación Cruzada (LOO) [21], [4]. A continuación se describen éstas para la selección de parámetro:

1.2.1. Dejar uno por Fuera (Leave-One-Out) LOO

En general es necesario algún mecanismo para encontrar un "buen"valor del parámetro de regularización λ. Es de usual interés encontrar una función que trabaje bien con ejemplos futuros. Ya que la distribución es generalmente desconocida. Si se tiene una cantidad pequeña de datos, una buena idea es emplear LOO [21].

La validación cruzada, algunas veces llamada estimación de rotación, es una técnica para la evaluar cómo el resultado de un análisis estadístico puede generalizar a un conjunto de datos inde-

pendientes. Una rutina de validación cruzada particiona una muestra de datos, en subconjuntos complemetarios, para realizar el análisis en un subconjunto, en este caso el conjunto de entrenamiento, y validando el análisis en el otro subconjunto llamado, conjunto de validación. Para reducir la variabilidad, se realizan múltiples rondas de validación cruzada empleando diferentes particiones y se promedian los errores de validación de todas las rondas [4], [22].

LOO, emplea una única observación de la muestra original como dato de validación y el resto lo utiliza como datos de entrenamiento. Esto se repite tantas veces hasta que cada observación de la muestra haya sido empleada como dato de validación, lo cual genera la mayoría de los casos alto costo computacional desde el punto de vista del número de veces que el proceso de entrenamiento se repite [4].

1.2.2. Validación Cruzada Generalizada GCV

GCV es un método muy popular y bastante éxitoso para la escogencia del parámetro de regularización. GCV es un método predictivo que busca minimizar el error cuadrático medio predictivo $\left\|Ax_\lambda - b^{exact}\right\|_2$. Como b^{exact} es deconocido, el método trabaja en lugar de ello con la función GCV

$$GCV(\lambda) = \frac{\|Ax_\lambda - b\|_2 .}{trace(I_m - AA^I)^2} \tag{1.13}$$

GCV se basa en la filosofía que si un elemento arbitrario b_i del lado derecho de b es dejado por fuera, entonces la correspondiente solución regularizada debería predecir bien esta observación, y la escogencia del parámetro de regularización debería ser independiente de una transformación ortogonal de b, así, esto conduce a la escogencia del parámetro de regularización, que minimiza la función GCV. Donde A^I es una matriz que produce la solución regularizada x_λ cuando es multiplicada con b, es decir, $x_\lambda = A^I b$. Note que $GCV(\lambda)$ está definida tanto para parámetros de regularización discretos como continuos. Existe un límite en los cálculos, puesto que GCV no puede ser empleada para calcular la función GCV para métodos de regularización iterativos [12],[13], [10].

Estabilidad

Cuando se tiene un problema de la forma $Ax = b$, realmente se cuenta con algo de la forma $Ax = b^\epsilon$, donde b^ϵ es una aproximación de b. Se puede calcular entonces, solamente $x_\lambda^\epsilon = A_\lambda^\# b^\epsilon$, donde $A_\lambda^\#$ ($A^\#$ es la matriz calculada con los datos disponibles), se emplea en vez de trabajar con la inversa generalizada A^+, la cual es en la mayoría de los casos mal condicionada. El parámetro λ permite construir una aproximación de esta inversa generalizada que sea fácil de calcular y sea bien condicionada. Se tendra $\left\|A_\lambda^\# - A^+\right\| \to 0$, cuando $\lambda \to 0$.

$$\left\|\tilde{x} - x_\lambda^\epsilon\right\| = \left\|\tilde{x} - x_\lambda + x_\lambda - x_\lambda^\epsilon\right\| \leq \|\tilde{x} - x_\lambda\| + \left\|x_\lambda - x_\lambda^\epsilon\right\|$$

donde $\tilde{x} = A^+ b$ y $\|\tilde{x} - x_\lambda\|$ se denomina "Error de Regularización" y $\left\|x_\lambda - x_\lambda^\epsilon\right\|$, es el "Error de Perturbación".

Cualquier procedimiento para la escogencia de parámetro, busca establecer un balance entre los errores de regularización y los de perturbación. Por tal razón, si se selecciona un λ muy grande, esto implicará errores de regularización muy grandes.

Recalculando, se tiene:

$$\|\tilde{x} - x_\lambda\| = \left\|A^+ b - A_\lambda^\# b\right\| = \left\|(A^+ - A_\lambda^\#) b\right\| \leq \left\|(A^+ - A_\lambda^\#)\right\| \|b\|$$

$$\left\|x_\lambda - x_\lambda^\epsilon\right\| = \left\|A_\lambda^\# b - A_\lambda^\# b^\epsilon\right\| = \left\|A_\lambda^\# (b - b^\epsilon)\right\| \leq \left\|A_\lambda^\#\right\| \left\|b - b^\epsilon\right\|$$

1.3. Regularización de Tikhonov

La principal idea en el método de Tikhonov es incorporar asunciones a priori acerca del tamaño y la suavidad de la solución deseada, en la forma de la suavidad de la función $\omega(f)$ en el caso continuo, o la seminorma $\|Lx\|_2$ en el caso discreto. Para problemas mal condicionados, es decir, problemas donde la solución no es única o no es una función continua de los datos, o sea si una perturbación arbitrariamente pequeña en los datos de entrada, puede causar una perturbación arbitrariamente grande en la solución encontrada; la regularización de Tikhonov en general se presenta como un problema de minimización de la forma [12], [3], [2]:

$$\text{mín} \left\{ \|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2 \right\}, \tag{1.14}$$

donde el parámetro de regularización λ controla el peso dado por la minimización del término de regularización con relación a la minimización de la norma residual.

1.3.1. Factores Filtrantes

Se puede reescribir la solución regularizada x_{reg} y el correspondiente vector residual $b - Ax_{reg}$ en términos de la descomposición en valores singulares de A de forma genérica [12]:

$$x_{reg} = \sum_{i=1}^{n} f_i \frac{u_i^T b}{\sigma_i} v_i \tag{1.15}$$

y

$$b - Ax_{reg} = \sum_{i=1}^{n} (1 - f_i) u_i^T b u_i + \sum_{i=n+1}^{m} u_i^T b u_i, \tag{1.16}$$

donde f_i son llamados *factores filtrantes*. Para los métodos particulares de regularización, estos caracterizan la amortiguación o el filtrado de los componentes SVD/GSVD (Generalized Singular Value Decomposition). Los factores filtrantes para la regularización de Tikhonov son

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \tag{1.17}$$

y los conjuntos efectivos de filtrado, son estos componentes de SVD/GSVD para los cuales $\sigma_i^2 < \lambda$.

2 Mínimos cuadrados regularizados para clasificación biclases (RLSC)

2.1. Fundamentos de RLSC

Uno de los problemas fundamentales en la teoría de aprendizaje es el siguiente. Suponga que se tienen dos clases de objetos, se encuentra con un nuevo objeto y se quiere asignar éste a una de las dos clases. Este problema puede formalizarse como se muestra a continuación, [25], [23]:

Teniendo los siguientes datos empíricos

$$\{(x_1, y_1), ..., (x_n, y_n)\} \in X \times \{\pm 1\} \tag{2.1}$$

Aquí, X es conjunto no vacío del cual se toman los patrones x_i (algunas veces llamados *casos, entradas* u *observaciones)* , usualmente refiriéndose al dominio; los y_i son llamados *etiquetas, objetivos,* o *salidas*. En aprendizaje se quiere ser capaz de generalizar los puntos desconocidos [21].

En el caso del reconocimiento de patrones, esto significa que dado un nuevo patrón $x \in X$, se quiere predecir el $y \in \{\pm 1\}$ correspondiente. El conjunto de entrenamiento satisface generalmente $x_i \in \mathbb{R}^n$, el objetivo es encontrar una función f que generalice bien sobre los nuevos ejemplos.

El problema de mínimos cuadrados busca el clasificador óptimo que minimice la siguiente función de costo

$$\frac{1}{2}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{2.2}$$

Este problema es mal condicionado porque no se tiene específicamente el conjunto de funciones f que se considera cuando se minimiza la función anterior [20]. Se requiere que la función f esté en un subconjunto convexo acotado de un RKHS H definido por una función kernel k definida positiva, (la norma de una función en este espacio es denotada por $\|f\|_K$) entonces RLS se plantea como un problema de regularización de Tikhonov con una pérdida cuadrática [5], [29], [20]:

$$\min_{f\in H}\frac{1}{2}\sum_{i=1}^{n}(f(x_i)-y_i)^2+\frac{\lambda}{2}\|f\|_K^2 \tag{2.3}$$

La forma de Tikhonov suaviza el equilibrio entre $\|f\|_K$ y el riesgo empírico, este equilibrio es controlado por el parámetro λ [12], [2].

Teorema 2.1. ***(Teorema de Representación)*** *Sea H un RKHS con un kernel, $K : X \times X \longrightarrow \mathbb{R}$ una función simétrica semidefinida positiva sobre un dominio compacto. Para cualquier función $L : \mathbb{R}^n \longrightarrow \mathbb{R}$, y cualquier función no decreciente $\Omega : \mathbb{R} \longrightarrow \mathbb{R}$ si*

$$J^* = \min_{f\in H} J(f) = \min_{f\in H} L(f(x_1), ..., f(x_n) + \Omega(\|f\|_K^2)) \tag{2.4}$$

está bien definida, entonces para cualquier $\alpha_1, \alpha_2, ..., \alpha_n \in \mathbb{R}$, tal que

$$f(\cdot) = \sum_{i=1}^{n}\alpha_i k(x_i, \cdot) \tag{2.5}$$

logra $J(f) = J^*$, Además, si Ω, es decreciente, entonces cada minimizador de $J(f)$ puede ser expresado en la forma de $f(\cdot)$ [24].

Lema 2.2. *El problema* (2.3) *es equivalente a*

$$\min_{c\in\mathbb{R}^n}\frac{1}{2}\|Y-Kc\|_2^2+\frac{\lambda}{2}c^tKc \tag{2.6}$$

Demostración. El teorema de representación [24] garantiza que la solución de (2.3) puede ser escrita como

$$f(\cdot) = \sum_{i=1}^{n} c_i k(x_i, \cdot) \tag{2.7}$$

para algún $c \in \mathbb{R}^n$, empleando las propiedades básicas del RKHS, [23], [25], reemplazando (2.7) en (2.3), ésta puede reescribirse como:

$$\min_{c\in\mathbb{R}^n}\frac{1}{2}\|Y-Kc\|_2^2+\frac{\lambda}{2}\|f\|_K^2 \tag{2.8}$$

Ahora

$$
\begin{aligned}
\|f\|^2 &= \langle f, f \rangle_K \\
&= \left\langle \sum_{i=1}^{n} c_i k(x_i, \cdot), \sum_{j=1}^{n} c_j k(x_j, \cdot) \right\rangle_K \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \left\langle k(x_i, \cdot), k(x_j, \cdot) \right\rangle_K \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \\
&= c^T K c
\end{aligned}
$$

□

Tomando la derivada con respecto a c en (2.6) e igualando a cero, c debe satisfacer

$$(K + \lambda I)c = Y \tag{2.9}$$

c existe y es único, además la matriz K es semidefinida positiva, por lo tanto $(K + \lambda I)$ es definida positiva (para $\lambda > 0$) [21].

La predicción de los puntos de entrenamiento, viene dado por

$$f(X) = Kc = K(K + \lambda I)^{-1} Y = KG^{-1} Y \tag{2.10}$$

Y la predicción de un nuevo punto, puede definirse por

$$f(X_*) = \sum c_i K(X_i, X_*) = K(X, X_*)^t c = Y^t G^{-1} k(X, X_*) \tag{2.11}$$

2.2. RLSC Caso Lineal

Aquí se asume el caso especial en el cual el kernel es lineal, $k(X_i, X_j) = X_i^T X_j$. En este caso, en la matriz X de $m \times n$, se debe tener ($m \geq n$). Es fácil ver que con el kernel lineal, la función de aprendizaje también lo es. Sea X de $m \times n$ la matriz de los datos de entrenamiento, entonces

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} c_i k(X_i, x) \\ &= c^T X x \\ &= w^T x \end{aligned}$$

donde se define el hiperplano w como $X^T c$ [30], [21].

2.2.1. Aproximación SVD para el caso lineal

La Descomposición en Valores Singulares SVD [18] puede resultar bastante útil para representar el problema que se quiere resolver de manera más eficiente en términos de costo computacional, por esta razón aquí se presenta una aproximación al problema (2.8) a partir del SVD de la matriz K, únicamente cuando se está empleando el kernel lineal y además teniendo presente que la cantidad de datos que aquí se maneje sea moderada (n, el número de variables no es muy grande) [21].

En síntesis, se puede emplear SVD, para expresar RLS. En esta descomposición, SVD puede ser escrita como $X = USV^t$, con $U \in \mathbb{R}^{m\times n}$, $S \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{n\times n}$, $U^tU = V^tV = VV^t = I_n$, y S diagonal y definida positiva. Entonces

$$K = XX^t = (USV^t)(VSU^t) = US^2U^t \tag{2.12}$$

$$\begin{aligned} K + \lambda I &= US^2U^t + \lambda I_n \\ &= \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} S^2 + \lambda I_n & \\ & \lambda I_{m-n} \end{bmatrix} \begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix} \\ &= U(S^2 + \lambda I_n)U^t + \lambda U_\perp U_\perp^t \\ &= U(S^2 + \lambda I_n)U^t + \lambda(I_n - UU^t) \end{aligned} \tag{2.13}$$

Ahora, hallando su inversa se tiene

$$
\begin{aligned}
(K+\lambda I)^{-1} &= \left(\begin{bmatrix} U & U_\perp \end{bmatrix}\begin{bmatrix} S^2+\lambda I_n & \\ & \lambda I_{m-n} \end{bmatrix}\begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix}\right)^{-1} \\
&= \begin{bmatrix} U & U_\perp \end{bmatrix}\begin{bmatrix} S^2+\lambda I_n & \\ & \lambda I_{m-n} \end{bmatrix}^{-1}\begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix} \\
&= U(S^2+\lambda I)^{-1}U^t + \lambda^{-1}U_\perp U_\perp^t \\
&= U(S^2+\lambda I)^{-1}U^t + \lambda^{-1}(I-UU^t) \\
&= U[(S^2+\lambda I)^{-1} - \lambda^{-1}I]U^t + \lambda^{-1}I
\end{aligned}
\tag{2.14}
$$

Por lo tanto, se puede definir c en términos de la descomposición en valores singulares de K, de la siguiente manera:

$$
\begin{aligned}
c &= (K+\lambda I)^{-1}Y \\
&= U[(S^2+\lambda I)^{-1} - \lambda^{-1}I]U^tY + \lambda^{-1}Y
\end{aligned}
\tag{2.15}
$$

La aproximación para resolver el problema es sencilla, pero se debe tener en cuenta el parámetro de regularización desconocido λ. Por lo tanto, a continuación se presentan los métodos empleados para su selección de manera autómatica, con el fin de evitar la introducción manual de parámetros que propicien que el método no proporcione resultados satisfactorios en términos de clasificación.

2.2.2. Escogencia del parámetro de regularización para RLSC lineal

Como puede apreciarse en el caso lineal, el problema (2.6) sólo requiere la selección de un único parámetro, puesto que al emplear el kernel lineal, la matriz K no necesita la escogencia de ningún parámetro adicional, por lo tanto, se presentan los métodos empleados para tal fin:

Función LOO para RLSC

Se denota $S = \{(X_1, Y_1), ..., (X_n, Y_n)\}$, se define S^i como el conjunto de datos con el i-ésimo punto removido. El i-ésimo valor LOO es $f_{S^i}(X_i)$, que corresponde con el valor de la función RLS entrenado sobre S^i aplicada a X_i. El $i-ésimo$ error LOO es $Y_i - f_{S^i}(X_i)$. Se define LOOE , como el vector de los valores leave-one-out, donde $\|LOOE\|_2^2$ se considera como una buena aproximación empírica para los errores en valores futuros, y generalmente, lo que se pretende es encontrar un parámetro minimizando esta cantidad [22], [21], [1].

Empleando las ecuaciones (2.14) y (2.15), puede presentarse una aproximación a la función LOO, para el caso especial en el que se emplea RLS como método de clasificación, definida como:

$$LOOE = \frac{c}{diag((K + \lambda I)^{-1})} = \frac{U[(S^2 + \lambda I)^{-1} - \lambda^{-1} I]U^t Y + \lambda^{-1} Y}{diag\,(U[(S^2 + \lambda I)^{-1} - \lambda^{-1} I]U^t + \lambda^{-1} I)} \tag{2.16}$$

En [21], se menciona un abuso de notación en la escritura de la ecuación (2.16), puesto que ellos consideran el cociente entre vectores, cómo otro vector. LOOE se denomina vector de errores LOO, el cual se considera como una buena aproximación para el error sobre los datos futuros. Por lo tanto, se emplea el minimizador de $||LOOE||_2^2$ como aproximación del "mejor" parámetro de regularización λ, según [22] y [21], posteriormente se calcula el error de clasificación.

Función GCV para RLSC

En esta sección se presenta cómo obtener la función GCV para el problema específico en el cual se emplea RLS para el problema de clasificación. En las secciones anteriores se mostró que RLSC se aproxima como un problema de regularización de Tiknonov. Por tal razón, el paramétro de regularización λ se hace presente en la solución del problema.

La obtención de la función GCV para la escogencia de λ en RLSC se hace partiendo de la descomposición en valores singulares de la matriz de entrada. Tal como se mostró en la sección anterior, la función GCV genérica para abordar la regularización de Tikhonov es la siguiente:

$$GCV(\lambda) = \frac{||Kc_\lambda - Y||}{trace(I - K(K + \lambda I)^{-1})} \tag{2.17}$$

Por tratarse del caso lineal, la matriz K, en este caso se toma como el kernel lineal, i.e $K = X^T X$, donde X, es la matriz de los datos de entrada.

Aplicando la descomposición en valores singulares de la matriz K y empleando los factores filtrantes, se obtiene la siguiente función GCV.

$$GCV(\lambda) = \frac{\sum_{i=1}^{n} \left(\frac{u_i^T Y}{\sigma_i^2 + \lambda} \right)}{\sum_{i=1}^{n} \frac{1}{\sigma_i + \lambda}} \tag{2.18}$$

Esta función representará la alternativa adecuada para la selección del parámetro de regularización, que permita trabajar con problemas mal condicionados, y será la función empleada para hallar en primera instancia el parámetro λ en el caso RLSC lineal.

2.3. RLSC caso no lineal

Aquí se presenta la extensión al caso no lineal del método de Mínimos Cuadrados Regularizados empleando el "Kernel Trick", en el cual se asume que la matriz K del sistema (2.6), es alguno de los kernel que se encuentran resumidos en la tabla 2.1.

Tipos de kernel	
Kernel polinomial	$K(x, x') = \langle x, x' \rangle^d$ $K(x, x') = \left(\langle x, x' \rangle + 1\right)^d$
Funciones de base radial Gaussiana	$K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
Función de base radial exponencial	$K(x, x') = \exp\left(-\frac{\|x-x'\|}{2\sigma^2}\right)$
Perceptrón Multicapa	$K(x, x') = \tanh\left(\rho \langle x, x' \rangle + \varrho\right)$
Series de Fourier	$K(x, x') = \frac{\sin\left(N+\frac{1}{2}\right)(x-x')}{\sin\left(\frac{1}{2}(x-x')\right)}$

Tabla 2.1. Tipos de kernel

Los tipos de kernel más empleados en la literatura son el de base radial gaussiana y el exponencial [29], [31], [30]. Cuando se emplean algunos de estos, se encuentra otro reto importante para realizar la tarea de clasificación, que consiste en la aparición de un nuevo parámetro libre, como en el caso de los kernel de base radial, que es el parámetro σ. En la literatura suele encontrarse que los investigadores realizan la prueba de sus métodos, variando manualmente su valor o realizando pruebas con intervalos de valores como en [29], y mostrando los diferentes resultados obtenidos al realizar esta escogencia.

En este trabajo, se plantea el uso del método de validación cruzada generalizada GCV para hallarlo, teniendo presente que al trabajar con el método regularizado, como ya se había mencionado en

la sección 2.2, también es necesario la escogencia del parámetro de regularización. Por tal razón, el desafío consiste ahora, en sintonizar ambos parámetros, con el fin de seleccionar automáticamente la mejor pareja (σ, λ) que ayude al buen desempeño del clasificador de Mínimos Cuadrados.

2.3.1. Escogencia del parámetro de regularización y el parámetro del Kernel para RLSC no Lineal

Aquí se presentará cómo realizar la escogencia de los parámetros tanto de regularización, como del kernel a partir de GCV. El primer paso consiste en encontrar la función GCV adecuada para la selección de ambos parámetros.

El sistema que se debe resolver es:

$$Kc = Y \tag{2.19}$$

donde K es la matriz kernel, Y es el vector que contiene las etiquetas y c es la solución buscada. Suponga en este caso que el kernel seleccionado es el de base radial, por lo tanto, la matriz K, entra a depender de la selección de este parámetro σ. El problema (2.19) se puede reescribir como:

$$K_\sigma c = Y \tag{2.20}$$

Por lo tanto, la solución deseada, cuando se tiene la versión regularizada de (2.19), dependerá de los parámetros σ y λ

$$c_{\lambda,\sigma} = (K_\sigma + \lambda I)^{-1} Y \tag{2.21}$$

Empleando la ecuación general GCV (1.13), es necesario conocer A^I, en este caso se tiene

$$K_\sigma^I = (K_\sigma + \lambda I)^{-1} \tag{2.22}$$

Así, se obtiene la ecuación GCV, para este problema particular

$$GCV(\sigma, \lambda) = \frac{\left\| K_\sigma c_{\lambda,\sigma} - Y \right\|}{trace(I - K_\sigma K_\sigma^I)^2} \tag{2.23}$$

reemplazando (2.21) y (2.22) en (2.23), se obtiene la función

$$GCV(\sigma, \lambda) = \frac{\left\| K_\sigma (K_\sigma + \lambda I)^{-1} Y - Y \right\|}{trace(I - K_\sigma (K_\sigma + \lambda I)^{-1})^2} \tag{2.24}$$

La función (2.24) será entonces la función objetivo al momento de realizar la clasificación, cuan-

do se encuentre con el problema de selección de parámetros. La minimización de esta función permitirá conocer el mejor parámetro de regularización λ y parámetro del kernel σ para RLSC [10].

Se implementan dos versiones de RLSC-GCV, las cuales se presentan a continuación:

2.3.2. RLSC-GCV iterativo

Para este caso, se empleará la versión no lineal del método de RLSC, empleando como tipo de kernel el de Base Radial Gaussiana, que se encuentra especificado en la tabla 2.1. Se denomina iterativo, puesto que deben seguirse los pasos mostrados a continuación, para obtener los parámetros de regularización λ y el parámetro σ del kernel gaussiano. Cabe anotar que el esquema para obtener este método fue adoptado de un método similar que se emplea para la escogencia de múltiples parámetros a partir de GCV, implementada por Peiling Xu en [28]:

1. Tomar como parámetro inicial de regularización λ_0, el que se obtiene a partir del método RLSC-GCV que emplea el kernel lineal K, a partir de la función (2.18). Es importante anotar que esta función usa la contenida en el toolbox **optimization** de Matlab, llamada *fminbnd*, donde se eligen como valores de búsqueda para el mínimo, los contenidos en el intervalo $(min(abs(s)), max(abs(s)))$, donde s, corresponde al vector que contiene los valores singulares de la matriz K, obtenidos a partir de la descomposición en valores singulares empleando la función *svd* de Matlab, como lo hace Hansen en [13].

2. A partir de λ_0, se obtiene el sigma inicial σ_0, empleando la función presentada en la ecuación (2.24), dejando estático el valor de λ y minimizando la función que depende en tal caso únicamente del valor de σ. Para minimizar la función (2.24), se emplea la función *fminbnd* del toolbox *optimization* de Matlab, en el cual el intervalo para buscar el valor σ que minimiza la función, está dado por $(0.1std(X), 10std(X))$, donde la notación **std** representa la desviación estándar de la matriz que contiene los datos de entrada.

3. Ya calculados los valores (λ_0, σ_0), se procede nuevamente a calcular el valor de λ_1. Notese que en este caso al tener explícito el valor de σ_0, por el paso anterior, es posible realizar la descomposición en valores singulares SVD, a la matriz K. Por esta razón, para ahorrar recursos, se puede emplear nuevamente la función (2.18), que encontrará el valor óptimo de λ_1, basándose en las mismas características propuestas en el paso 1.

4. Con el valor de λ_1 obtenido en el paso 3, se invoca nuevamente la función (2.24), para que calcule de nuevo el valor de σ_1, con las mismas características empleadas en el paso 2.

5. Se repite nuevamente el proceso dado en los pasos anteriores hasta obtener la convergencia del método, lo cual arrojará la pareja (σ_i, λ_i) adecuada.

Después de aplicar los 5 pasos anteriores y de iterar cuantas veces sea necesario, se elige la pareja (σ_i, λ_i), como parámetro para calcular el kernel y parámetro de regularización, y se procede a aplicar el algoritmo base para realizar la clasificación. Para RLSC-GCV iterativo, con las bases de datos propuestas, sólo fué necesario realizar 2 iteraciones, tal cómo se muestra en [28], donde se prueba la convergencia del método en dos pasos, en los cuales se logra sintonizar la pareja (σ, λ), que funciona bien como parámetro del kernel y de regularización en el método RLSC.

En la figura 2.1, se muestra el diagrama de flujo para RLSC-GCV iterativo.

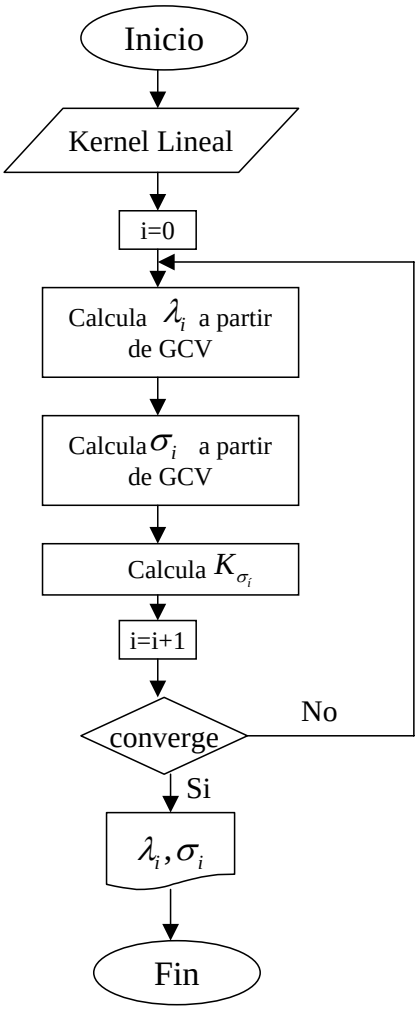


Figura 2.1. Diagrama de flujo para RLSC-GCV iterativo

2.3.3. RLSC-2DGCV

En este caso el método será llamado RLSC-2DGCV (Regularized Least Square Clasification - 2Dimensional Generalized Cross Validation) porque, a diferencia del algoritmo anterior, la función

(2.24), calcula al mismo tiempo los parámetros σ y λ. Para lograr este objetivo debe emplearse una función que resuelva en este caso, el problema de optimización multidimensional, teniendo en cuenta dos restricciones importantes. La primera de ellas es que el valor de λ debe ser mayor a cero, y la segunda es que el kernel Gaussiano es una función simétrica y presenta discontinuidad en el punto $\sigma = 0$, por tal razón debe restringirse el intevalo de búsqueda del mínimo a valores de σ y λ mayores a cero.

Para lograr este objetivo, se emplea la función de Matlab, también disponible en el toolbox *optimization*, llamada *fmincon*, la cual permite minimizar la función (2.24) multidimensional, imponiendo las restricciones respectivas, tanto lineales, como no lineales. Una de las entradas de esta función, son los valores iniciales de búsqueda para el mínimo, para ello, se emplean como aproximación, los valores iniciales σ_0 y λ_0 del algoritmo anterior.

3 Marco Experimental

3.1. Medición de Resultados

Para calcular el rendimiento de los clasificadores, se prueba con el 70 % de los datos como datos de entrenamiento y el 30 % restante como datos de validación. Se emplean matrices de confusión, las cuales son utilizadas en problemas de clasificación binaria, para extraer algunas medidas que permitan medir el desempeño de los clasificadores, ver [9]. A continuación se realiza una descripción detallada para la construcción de las matrices y toma de medidas a partir de las mismas.

En las matrices de confusión, se puede distinguir entre la clase verdadera y la clase predicha por el clasificador, empleando las etiquetas producidas por el modelo. Se tendrán entonces 4 casos posibles: Si una instancia es positiva y es clasificada como positiva, se llamará *verdadero positivo* (TP), si es clasificada como negativa, se contará como *falso positivo* (FP). Si la instancia en negativa y es clasificada como negativa, se contará como *verdadero negativo* (TN) y si es clasificada como positiva, se contará como *falso negativo* (TN), ver [9].

La matriz de confusión, será por lo tanto de tamaño 2×2 (también llamada tabla de contingencia), y se construye de la siguiente manera:

	Verdaderos	Falsos	
Positivos	TP	FP	Valores predichos positivos
Negativos	FN	TN	Valores predichos negativos
	Sensibilidad	Especificidad	EXACTITUD

Tabla 3.1. Construcción de una matriz de confusión para clasificación binaria

De allí se pueden extraer las siguientes medidades que permitirán medir el desempeño de los clasificadores:

$$\text{Sensibilidad} = TPtasa = \frac{TP}{TP + FN} \tag{3.1}$$

$$\text{Especificidad} = TNtasa = \frac{TN}{TN + FP} \tag{3.2}$$

$$FPtasa = 1 - \text{Especificidad} = \frac{FP}{FP + TN} \tag{3.3}$$

$$FNtasa = 1 - \text{Sensibilidad} = \frac{FN}{FN + TP} \tag{3.4}$$

Para medir la exactitud del algoritmo, se empleará:

$$\text{Exactitud} = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.5}$$

La exactitud será la medida estándar empleada para evaluar el desempeño de los clasificadores, pues permite conocer que tan cerca del valor real se encuentra el valor medido (sesgo de la estimación), tal como muestra la figura 3.1. En algunos casos será necesario medir para algunas bases de datos, la especificidad y sensibilidad del clasificador para realizar un análisis un poco más detallado.

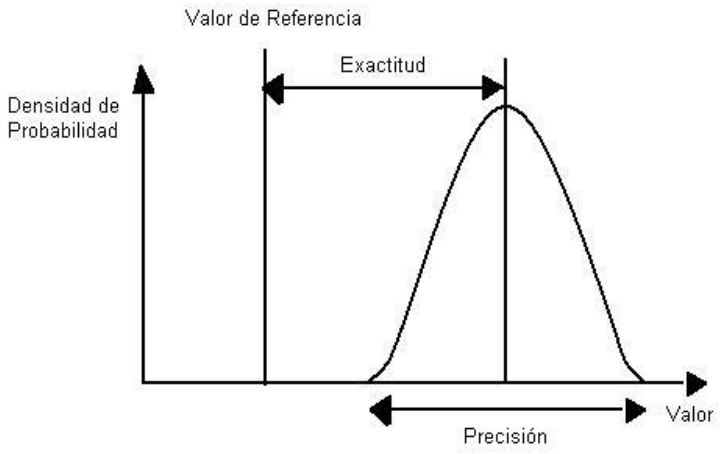


Figura 3.1. La exactitud indica la proximidad del resultado medido al valor verdadero, la precisión indica la reproducibilidad de la medida

En términos más prácticos, tener una exactitud del 100 % significa que las etiquetas originales son exactamente iguales a las obtenidas por el algoritmo de clasificación. Al emplear el sistema de tomar 70 % de los datos para entrenamiento y 30 % para validación para medir el desempeño de los clasificadores, se pueden rotar los grupos cuantas veces sea requerido y se pueda, ya que la permutación debe ser finita y completamente aleatoria. Para tomar medidas en este experimento, los grupos de entrenamiento se rotan 15 veces para cada clasificador, se evalúa la exactitud media de cada clasificador y su respectiva desviación estándar.

Se calculan las matrices de confusión en cada ejecución del clasificador y se promedian sus entradas, con lo cual se pueden realizar gráficos para comparar desempeño.

3.2. Descripción de las Bases de Datos

Los clasificadores serán probados empleando 7 bases de datos del repositorio de la **UCI Machine Learning Repository**, (http://archive.ics.uci.edu/ml/). Estas bases de datos contienen variables escalares de tipo real, especiales para clasificación supervisada. A continuación se presenta una breve descripción de las bases de datos:

Base de datos de Vinos (wine): Estos datos son los resultados de un análisis químico de vinos cultivados en la misma región en Italia, pero que provienen de tres diferentes cultivos, de los cuales se tomarán sólo dos, debido a que se esta trabajando con clasificación binaria. Esta base de datos posee 13 variables con 130 mediciones cada una, de las cuales 59 pertenecen a la primera clase y 71 a la segunda.

Base de datos para identificación de vidrios (glasses): Contiene 9 variables, con un total 214 mediciones divididas en 6 clases de vidrios, de las cuales se tomarán las dos primeras clases. La primera contiene 70 mediciones (ventanas de construcción procesadas), y la segunda 76 (ventanas de construcción no procesadas), para un total de 146 mediciones.

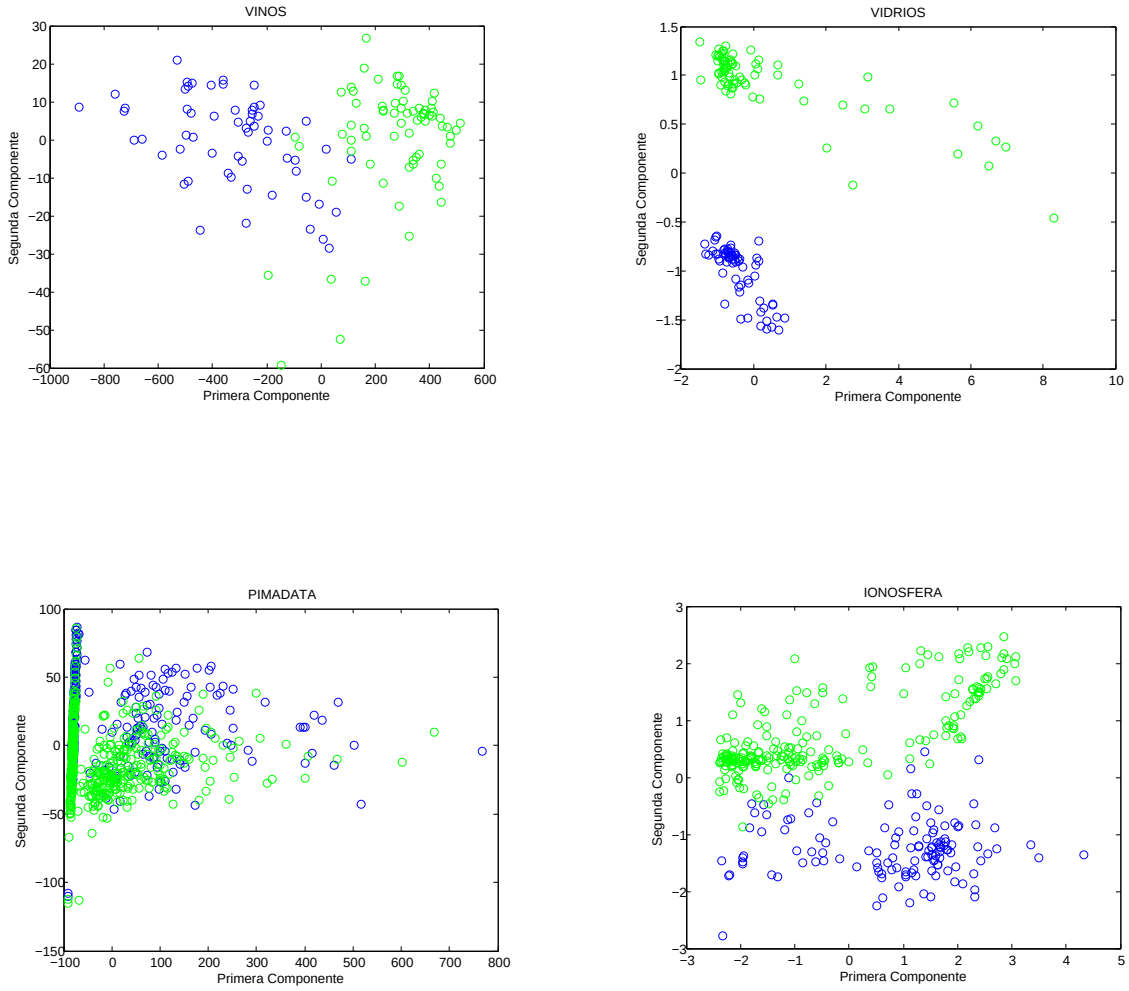
Base de datos de diabetes de los indígenas Pima (pimadata): Diagnóstico que determina si la paciente (mujeres de herencia índigena Pima) muestra signos de diabetes de acuerdo con los criterios de la Organización Mundial de la Salud. La población vive cerca de Phoenix, Arizona, EE.UU. La base contiene mediciones de 8 variables. Esta base posee 768 mediciones, repartidas en dos clases, la primera 268 (diagnostico de diabetes positivo), y la segunda (500 diagnostico de diabetes negativo).

Base de datos de la Ionosfera (ionosfer): Esta base de datos contiene mediciones de electrones libres en la ionosfera a partir de un radar que procesa las señales empleando funciones de autocorrelación cuyos argumentos son el tiempo de un pulso y el número de pulsos. El radar retorna *bueno* si estos muestran evidencia de algún tipo de estructura en la ionosfera y *malo* si no lo hace. Esta base contiene 33 variables con 351 mediciones, de las cuales 126 pertenecen a la primera clase y 225 a la segunda.

Base de datos Heart-c y Heart-h: El objetivo de estas bases de datos es identificar pacientes con afecciones cardiacas de las que no las poseen, son datos tomados en Cleveland y Hungarian. contienen 13 variables y 294 mediciones cada una. Para Heart-c se tienen 160 mediciones pertenecientes al primer grupo y 134 del segundo grupo, mientras que Heart-h 188 del primer grupo y 106 del segundo.

Base de datos Cartas (letters): El objetivo de esta base es identificar de una muestra rectangular de pixeles blancos y negros, las 26 letras mayúsculas del alfabeto inglés. Esta base contiene 17 variables con 20.000 mediciones cada una y un total de 26 clases, de las cuales se tomaron las correspondientes a la letras U con 813 mediciones y la letra W con 752, para un total de 1565 mediciones.

Se aplica Análisis de Componentes Pricipales PCA, para tener una idea global de la naturaleza de los datos. Para esto, se proyectan las bases de datos a las dos primeras componentes, lo cual se realiza empleando la función *princomp* disponible en Matlab. Se pinta de azul el primer grupo y de verde el segundo, con el fin de analizar una posible separabilidad de los datos y determinar que tanto influye esto en la tarea de clasificación. La figura 3.2, muestra los resultados obtenidos.



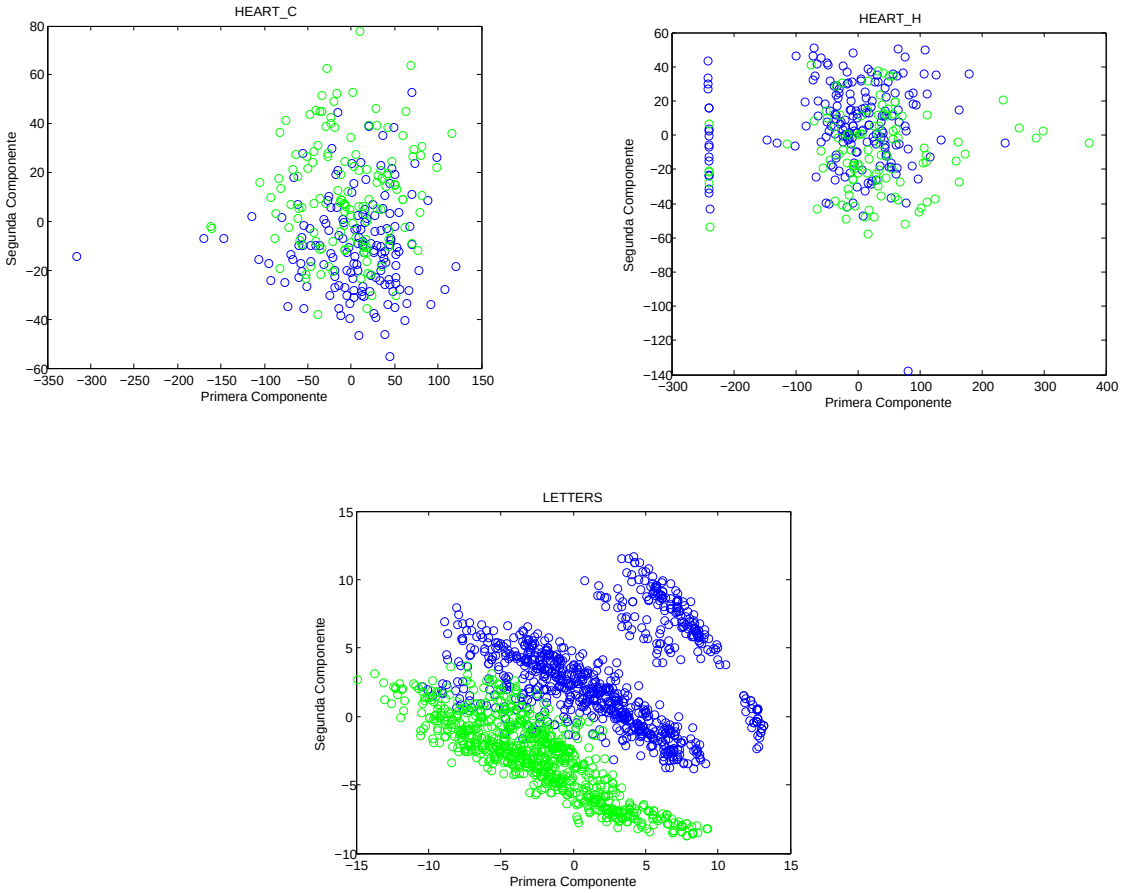


Figura 3.2. Proyección de las bases de datos en 2 dimensiones aplicando PCA

3.3. Marco experimental RLSC lineal

Para probar y comparar el desempeño de los métodos de clasificación por mínimos cuadrados regularizados, en su versión lineal, con otros métodos de clasificación, se emplean las 7 bases de datos de la UCI. La técnica RLSC-GCV será entonces el método de Mínimos Cuadrado Regularizados donde se emplea GCV para la escogencia del parámetro de Regularización, RLSC-LOO, será la versión automatizada de mínimos cuadrados regularizados, pero empleando LOO como método para la escogencia del parámetro de regularización.

3.3.1. Experimento

El desempeño de los clasificadores lineales implementados en este trabajo, tales como RLSC-GCV y RLSC-LOO, se comparan con otros métodos de clasificación, tales como **LDA**, que es un clasificador lineal discriminante, basado en densidades normales (Regla de Bayes): este clasificador calcula la clasificación entre las clases de la base de datos asumiendo densidades normales con matrices de covarianza iguales. La matrices de covarianza conjunta son ponderadas por el promedio

(las probabilidades a priori) de las clases de las matrices de covarianza. Este clasificador es tomado de un toolbox para reconocimiento de patrones de Matlab, llamado **PRtools**, [7].

El clasificador **BAYES**, es el clasificador bayesiano disponible en el toolbox *bioinformatic* de Matlab, el cual emplea como función discrimante *DiagLinear* que utiliza la diagonal de las matrices de covarianza calculadas para cada grupo y se basa en la regla de Bayes como criterio de clasificación.

Otro clasificador empleado es el de máquinas de vectores de soporte **SVM**. En este caso se utilizó también el clasificador **SVC** disponible en el toolbox PRtools. Para SVM es necesario la selección del parámetro C que controla la compensación entre los errores de entrenamiento y los márgenes rígidos, con el fin de evitar sobreajuste, creando así un margen blando, que permita algunos errores en la clasificación a la vez que los penaliza. Aquí el toolbox realiza la optimización automática de este parámetro de regularización empleando validación cruzada, con la función *regoptc*. Además, por defecto emplea el kernel lineal, lo cual es fundamental para esta prueba, puesto que se trata de clasificadores lineales.

Base de datos	LDA AV-STD	BAYES AV-STD	SVM AV-STD	RLSC-LOO AV-STD	RLSC-GCV AV-STD
Wine	100 ± 0.0	94.1 ± 4.0	99.9 ± 0.0	96.1 ± 1.5	100 ± 0.0
Glasses	77.8 ± 2.1	59.5 ± 7.0	74.6 ± 2.7	67.1 ± 4.1	71.4 ± 7.7
Pimadata	77.6 ± 0.7	73.2 ± 1.2	78.1 ± 1.0	68.6 ± 1.0	73.9 ± 1.2
Ionosfer	91.2 ± 1.2	80.6 ± 5.3	93.1 ± 0.7	85.9 ± 1.6	89.3 ± 1.4
Heart_C	85.4 ± 1.9	82.8 ± 1.5	83.6 ± 0.9	80.9 ± 1.6	85.5 ± 1.4
Heart_H	N/C	N/C	84.9 ± 1.1	80.8 ± 1.2	83.4 ± 1.3
Letters	98 ± 0.2	96.8 ± 0.4	99.3 ± 0.1	95.6 ± 0.3	98 ± 0.2

Tabla 3.2. Exactitud en la clasificación, medida en porcentaje, donde AV (Promedio) y STD (desviación estándar)

	LDA	BAYES	SVM	RLSC-LOO	RLSC-GCV
Wine	0.53	0.22	1.20	9.09	0.68
Glasses	0.54	0.22	1.16	10.89	0.50
Pimadata	0.55	0.74	18.64	987.65	18.57
Ionosfer	0.59	0.36	3.74	105.66	1.75
Heart_C	0.54	0.27	2.24	64.74	1.31
Heart_H	N/C	N/C	2.13	63.07	1.25
Letters	1.89	2.13	431.17	7802.63	193.87

Tabla 3.3. Tiempo de ejecución de los clasificadores lineales, medido en segundos

En la tabla 3.2 se presentan los desempeños de los 5 clasificadores con su respectiva exactitud

media, y desviación estándar y en la tabla 3.3 se muestran los tiempos de ejecución de cada cada clasificador, medido en segundos. En ambas tablas se puede observar que **RLSC-GCV** aunque no supera en general el desempeño de los demás clasificadores, si es uno de los más competitivos. Es necesario resaltar que la versión regularizada con escogencia de parámetro automática a partir de la función GCV optimiza el proceso y presenta buenos resultados en la clasificación sin demandar un alto costo computacional, comparado con los otros métodos. Por lo tanto, sus mayores ventajas recaen sobre la exactitud, bajo costo computacional y sobre todo automatización del algoritmo.

RLSC con LOO como escogencia de parámetro funciona bien en términos de clasificación, pero implica altos costos computacionales, a medida que se toma una base de datos más grande, el proceso se tarda más en obtener resultados y más aún cuando se itera 15 veces para obtener una precisión media y su respectiva desviación estándar.

Cabe anotar aquí que para automatizar la escogencia del parámetro de regularización a partir de la función LOO, se calcula $c(\lambda)$ y se construye $\|LOOE(\lambda)\|_2^2$ generando un intervalo de parámetros de regularización logarítmicos, posteriormente LOO es graficado para estos valores. Entonces el minimizador de LOOE se calcula vía *fminbnd* (función de Matlab para resolver problemas de optimización. Ésta función encuentra el mínimo local contenido en un intervalo definido previamente. La función está disponible en el toolbox *optimization*), por lo tanto, se emplea el minimizador de LOOE como una aproximación inicial.

El clasificador **LDA**, funciona muy bien en términos de clasificación y es un método de bajo costo computacional, pero en términos de regularización, es necesario la escogencia manual del parámetro, para este caso, el parámetro no se varió, se utilizó el que emplea el método por defecto, es decir 0. Por lo tanto, la versión que se probó no es una versión regularizada del algoritmo, lo cual trae como inconveniente que al encontrarse con una matriz de entrenamiento mal condicionada, el método no converge, como se observa en la base de datos **heart_h**.

Los resultados del clasificador Bayesiano **BAYES**, son los más desfavorables en términos de exactitud y al igual que LDA, este método no converge con la base de datos heart_c, aunque cabe destacar el bajo costo computacional.

Como se puede apreciar en la tabla 3.2, las máquinas de vectores de soporte **SVM** funcionan bien como clasificador, en realidad es uno de los clasificadores más competitivos en este campo, pero presenta el inconveniente de la automatización de parámetros, entonces los resultados de la clasificación se encuentran sesgados por la escogencia del parámetro C, puesto que es fundamental la escogencia de un buen parámetro que mantenga el equilibrio entre los errores de clasificación,

sin sobreentrenar el algoritmo, pero que a la vez los penalice. Sin embargo al emplear la función del toolbox PRtools, este realiza la escogencia del parámetro y se obtiene resultados bastante competitivos en términos de desempeño, y se destaca el aumento de tiempo al emplear bases de datos más grandes, como en el caso de letters.

Se muestra en la tabla 3.4, los intervalos de confianza contruidos para la exactitud media medida en todos los clasificadores con todas las bases de datos. Estos intervalos fueron construidos con una confiabilidad del 95 %, empleando la t-student como estadístico, con $n-1$ grados de libertad, donde $n-1=14$, puesto que el tamaño de la muestra se considera como $n=15$, que corresponde al número de ejecuciones o iteraciones de cada método. Los intervalos fueron construidos de la siguiente manera [19]:

$$\overline{x} - t_{\alpha/2,n-1}\, s/\sqrt{n} \leq \mu \leq \overline{x} + t_{\alpha/2,n-1}\, s/\sqrt{n} \tag{3.6}$$

donde $\overline{x}$ representa el promedio en la exactitud medida en la muestra y s su respectiva desviación estándar.

	LDA	**BAYES**	**SVM**	**RLSC-LOO**	**RLSC-GCV**
Wine	0.99 - 1.00	0.92 - 0.96	0.99 - 1.00	0.95 - 097	0.99 - 1.00
Glasses	0.76 - 0.79	0.55 - 0.64	0.73 - 0.76	0.65 - 0.69	0.67 - 0.75
Pimadata	0.77 - 0.78	0.72 - 0.74	0.77 - 0.79	0.68 - 0.69	0.73 - 0.75
Ionosfer	0.91 - 0.92	0.76 - 0.84	0.92 - 0.94	0.85 - 0.87	0.89 - 0.90
Heart_C	0.84 - 0.86	0.82 - 0.84	0.86 - 0.87	0.80 - 0.82	0.84 - 0.86
Heart_H	N/C	N/C	0.84 - 0.86	0.80 - 0.82	0.83 - 0.84
Letters	0.97 - 0.98	0.96 - 0.97	0.99 - 1.00	0.94 - 0.97	0.97 - 0.98

Tabla 3.4. Intervalos de confianza para la exactitud media con una confiabilidad del 95 % para los clasificadores lineales

Se observa en la mayoría de casos de la tabla 3.4 que los intervalos son bastante angostos, lo cual refleja que las 15 iteraciones son suficientes en la mayoría de los casos, para determinar la exactitud media verdadera de cada clasificador.

Como se mostró en las tablas 3.2 y 3.3, los métodos que pueden ser comparados en términos de error y de tiempo son SVM y RLSC-GCV, por la naturaleza de su procedencia, ya que pertencen a la misma familia. Por lo tanto, la gráfica 3.3 presenta una comparación del error contra el tiempo tomado por estos dos clasificadores para ejecutar las 15 iteraciones. Es importante notar que, el la parte izquierda se encuentra la gráfica para 4 bases de datos, cuyos tiempos están ordenados de menor a mayor, y al lado derecho, las 3 restantes, ordenadas también en tiempos de menor a mayor.

La proximidad de cada gráfico al punto inicial $(0, 0)$, denota ganacia tanto en tiempo como en error. Por tal razón, se puede observar cómo los errores en RLSC-GCV son en la mayoría de casos compensados con un mejor desempeño en cuanto al tiempo de ejecución.

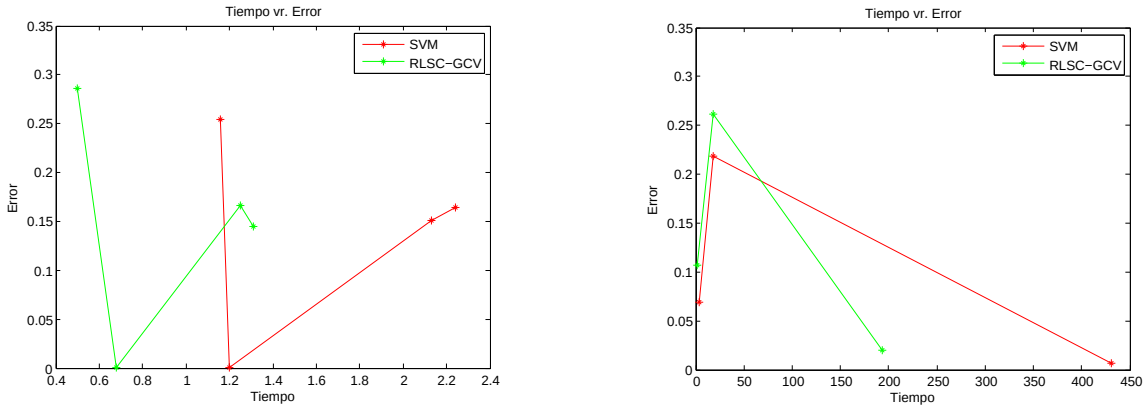


Figura 3.3. Tiempo vr. Error de SVM y RLSC-GCV

Las figuras 3.4 y 3.5 muestra las gráficas del espacio ROC y las líneas de exactitud medidas para las bases de datos pimadata y heart_c, con las cuales se puede realizar una análisis para el desempeño de los diferentes métodos de clasificación.

Las gráficas ROC son bidimensionales, en las cuales la tasa de falsos positivos se grafica en el eje X y la tasa de verdaderos positivos en el eje Y. Esta gráfica representa el equilibrio entre los beneficios (TP), y el costo (FN). Cabe anotar que un clasificador *discreto*, como en el caso de RLSC-LOO y RLSC-GCV, es aquél que produce únicamente etiquetas, en este caso, el par $(FPtasa, TPtasa)$, correspondiente a un único punto en el espacio ROC. En la gráfica pueden detallarse por tal razón 4 puntos, donde cada uno de ellos representa un clasificador, ver [9] y [8].

Las coordenadas $(0, 0)$, indican que el clasificador nunca emite una clasificación positiva, tal clasificador no comete falsos positivos, pero tampoco gana en verdaderos positivos. La estrategia opuesta para emitir incondicionalmente clasificaciones positivas se encuentran en el punto $(1, 1)$. El punto $(0, 1)$ representa la clasificación perfecta, por tanto, el punto $(1, 0)$ representará una clasificación totalmente incorrecta. Informalmente un punto es mejor que otro, si este se encuentra más al nor-occidente del otro (TPtasa es mayor, FPtasa es menor), [9].

La gráfica de exactitud, en cambio, compara la sensibilidad (o tasa de verdaderos positivos) vs. especificidad (o tasa de verdaderos negativos). Esta gráfica se genera con el fin de determinar si existe un equilibrio entre el desempeño del algoritmo para generar etiquetas tanto positivas como

negativas.

Una sensibilidad del 100 % indica un reconocimiento de todos los valores positivos, mientras que una especificidad del 100 % lo hace con los negativos, por lo tanto, para determinar si existe un balance entre estas dos medidas, se realiza la gráfica de exactitud y se calcula el área bajo la línea de exactitud generada por cada clasificador. Esta gráfica muestra que a medida que el área de la antidiagonal se aproxime a 0.5, puede considerarse mejor el desempeño clasificador, puesto que al estar más próximos los puntos a los extremos $(0, 1)$ y $(1, 0)$, indica una mayor sensibilidad y especificidad del método, ya que se tendría una alta proporción de etiquetas positivas y negativas clasificadas correctamente.

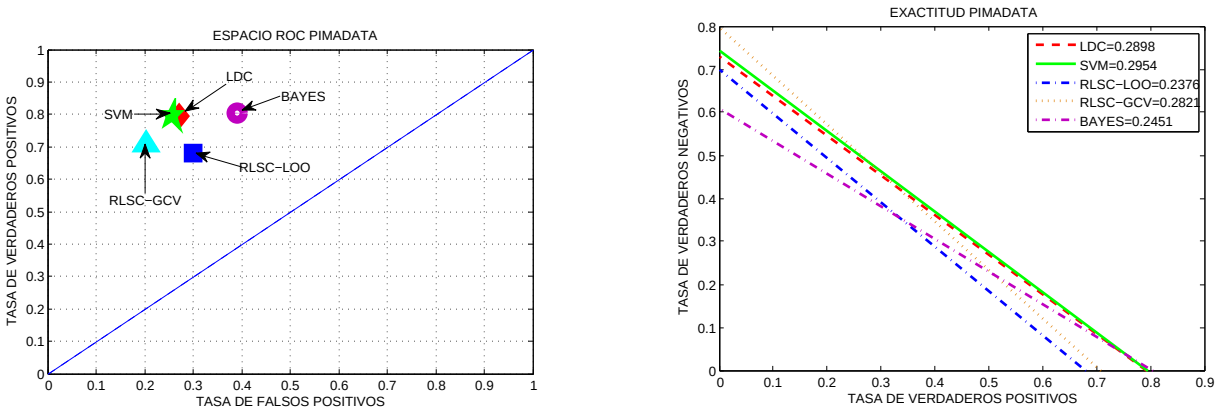


Figura 3.4. A la izquierda se encuentra la gráfica ROC para la base de datos Pimadata y a la derecha la línea de exactitud de la misma base de datos)

En la figura 3.4 se puede observar la grafica del espacio ROC y la línea de exactitud generadas por los 4 métodos de clasificación para la base de datos Pimadata. La gráfica del espacio ROC muestra que el clasificador RLSC-GCV, aunque no tiene la tasa más alta de verdaderos positivos, funciona bien al momento de clasificar las etiquetas negativas, es decir, tuvo la tasa de falsos positivos más baja para los 4 métodos, es decir el costo más bajo. LDC, SVM y RLSC-LOO mantuvieon una tasa similar de falsos positivos, pero LDC y SVM fueron superiores al momento de acertar en las etiquetas de la clase positiva.

Para la base de datos Pimadata, se puede observar que el área más grande fué obtenida por SVM, seguidas por los clasificadores LDA y RLSC-GCV, hablando de manera general ambos poseen la misma exactitud, pues logran un área de 0.28, mientras RLSC-LOO está 4 puntos por debajo de estos resultados.

Se puede concluir entonces que aunque LDA y SVM son un poco más precisos para lograr clasificaciones, en este caso de etiquetar positivos, RLSC-GCV y LDC poseen exactitudes iguales, que contempla tanto clasificación correcta de positivos como de negativos, pero RLSC-GCV genera costos más bajos.

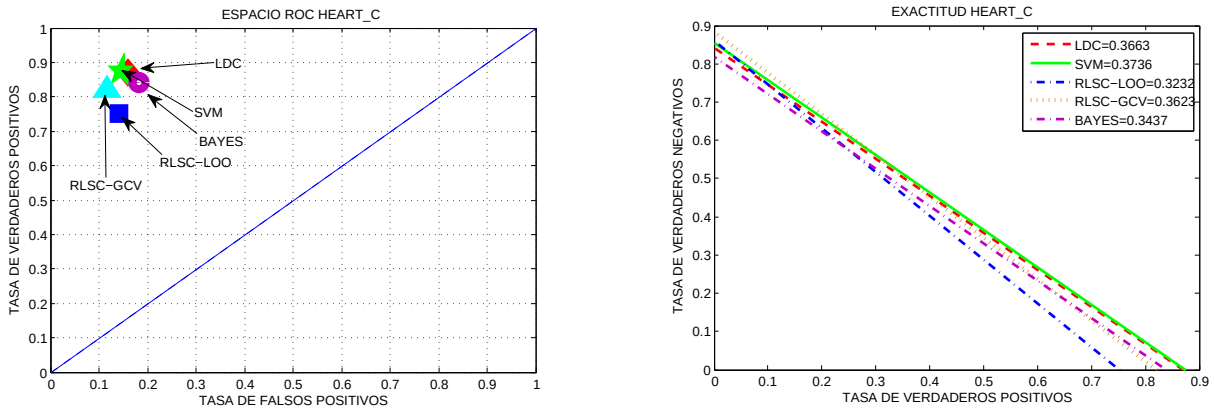


Figura 3.5. A la izquierda se encuentra la grafica ROC para la base de datos Heart_c y a la derecha la linea de exactitud de la misma base de datos)

Por otra parte, la figura 3.5, muestra el espacio ROC para la base de datos Heart_c, donde se puede apreciar un comportamiento similar al de la base de datos Pimadata, puesto que RLSC-GCV sigue siendo mejor al clasificar en menor proporción los falsos positivamente, es decir menor costo.

La línea de exactitud para este caso muestra nuevamente que LDA y RLSC-GCV presentan datos de exactitud similares, mientras SVM esta un punto por encima y RLSC-LOO siguen estando por debajo.

Para tener una visión concreta del desempeño de los clasificadores, la tabla 3.5, muestra las áreas bajo las líneas de exactitud calculadas para todas las bases de datos. En la tabla 3.6 se promediaron las áreas de los 5 métodos, lo cual muestra valores similares para SVM y LDA, pero cabe resaltar aquí, que LDA y SVM superán en desempeño a RLSC-GCV, pero cómo se mencionó arriba, las mayores ventajas que presenta RLSC-GCV es que permite a partir de bajos costos computacionales obtener resultados útiles y estables cuando se trabaja con problemas mal condicionados, y evita sesgos en los resultados, al ser una versión automática del método.

La gráfica 3.10, muestra el diagrama de barras para las áreas de los clasificadores bajo la línea de exactitud con todas las bases de datos.

CLASIFICADOR	VIN	VID	PIM	ION	HE_C	HE_H	LET
LDC	0.50	0.30	0.29	0.43	0.36	N/C	0.48
BAYES	0.44	0.17	0.24	0.31	0.34	N/C	0.47
SVM	0.49	0.28	0.26	0.41	0.37	0.34	0.49
RLSC-LOO	0.46	0.22	0.24	0.37	0.32	0.32	0.46
RLSC-GCV	0.50	0.26	0.28	0.40	0.36	0.34	0.48

Tabla 3.5. Áreas bajo la línea de exactitud de los clasificadores lineales

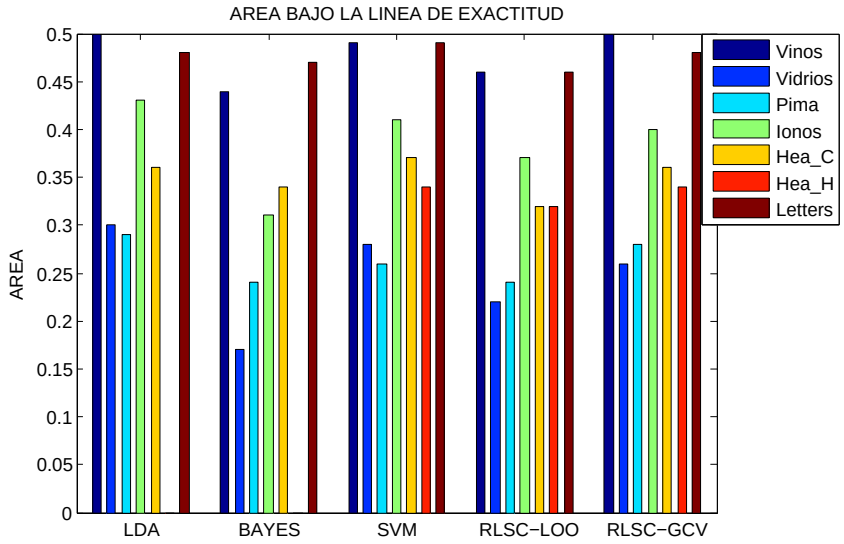


Figura 3.6. Diagrama de barras para el área bajo la línea de exactitud, clasificadores lineales

	LDA	BAYES	SVM	RLSC-LOO	RLSC-GCV
Área Promedio	* 0.39	* 0.33	0.39	0.34	0.37

Tabla 3.6. Área Promedio bajo la línea de Exactitud de los Métodos de Clasificación Lineales

*Area promedio calculada sin tomar en cuenta la base de datos en la cual el método no converge.

3.4. Marco experimental RLSC no lineal

En esta sección, para probar y comparar el desempeño del método de clasificación por mínimos cuadrádos regularizados en su versión no lineal, serán empleadas las mismas bases de datos probadas para las versiones lineales. Se considerarán RLSC-GCV iterativo y RLSC-2DGCV.

3.4.1. Experimento

Se prueba el rendimiento del clasificador de mínimos cuadrados regularizado, con su versión iterativa y 2D, y se compararán los resultados con el clasificador de vecinos más cercanos K-NN, el cual se ha considerado como una herramienta bastante poderosa en términos de clasificación. K-NN es una regla de decisión bien conocida, extensamente empleada en el reconocimiento de patrones. La regla de K-NN para la tasa de error de clasificación, se acerca asintóticamente a la tasa de error óptima de Bayes a medida que k, el número de vecinos, aumenta. Es particularmente efectivo cuando las distribuciones de probabilidad de las variables de la función no son conocidas, haciendo ineficiente la regla de decisión de Bayes, [27].

La rutina de K-NN empleada para realizar comparaciones, fue la contenida en el toolbox *prtools* [7], llamada **KNNC**, debido a que en este caso se hace necesario la escogencia del parámetro k correspondiente a la cantidad de vecinos que se consideran, y prtools, tiene automatizada la selección de este parámetro a partir de validación cruzada, por lo tanto, no se presentarán problemas que puedan ocasionar sesgos en la medición del desempeño.

En la tabla 3.7se presentan los desempeños de los 3 clasificadores para las 7 bases de datos, cada uno con sus respectiva exactitud media y desviación estándar, medida en las 15 rotaciones del algoritmo, en términos porcentuales, mientras que la tabla 3.8 muestra el tiempo de ejecución de cada clasificador con todas las bases de datos. Se puede observar como los desempeños de los métodos RLSC-GCV iterativo y RLSC-2DGCV son muy similares, y superan en la mayoría de los casos los desempeños de K-NN, aunque este presenta en la mayoría de los casos un costo computacional más bajo. Ampliando un poco más el análisis, RLSC no lineal, supera en gran medida el desempeño de los clasificadores lineales.

Base de datos	K-NN AV-STD	RLSC-GCV Iterativo AV-STD	RLSC-2DGCV AV-STD
Wine	94.9 ± 1.4	100 ± 0.0	100 ± 0.0
Glasses	96.4 ± 5.4	92.6 ± 7.8	96.1 ± 9.5
Pimadata	78.4 ± 1.0	79.3 ± 1.0	80.4 ± 1.1
Ionosfer	97.4 ± 5.5	100 ± 0.0	100 ± 0.0
Heart_C	71.5 ± 3.0	87.0 ± 1.7	85.9 ± 1.5
Heart_H	76.5 ± 8.2	89.8 ± 4.5	93.6 ± 4.1
Letters	100 ± 0.0	100 ± 0.0	100 ± 0.0

Tabla 3.7. Exactitud de la Clasificación de los métodos no lineales, medida en porcentaje, donde AV (Promedio) y STD (Desviación Estándar)

En la tabla 3.9, se construyen los intervalos de confianza para la exactitud media de los clasifica-

Base de datos	K-NN AV-STD	RLSC-GCV Iterativo AV-STD	RLSC-2DGCV AV-STD
Wine	1.40	1.89	2.54
Glasses	0.54	2.41	2.97
Pimadata	10.48	114.96	59.27
Ionosfer	2.82	13.35	9.06
Heart_C	2.68	8.87	8.21
Heart_H	2.68	7.81	5.58
Letters	110.16	430.06	426.42

Tabla 3.8. Tiempo de ejecución de los clasificadores no lineales, medido en segundos

dores no lineales medida en todas las bases de datos, con las mismas consideraciones propuestas en la sección 3.3.

Base de datos	K-NN AV-STD	RLSC-GCV Iterativo AV-STD	RLSC-2DGCV AV-STD
Wine	0.94 - 0.96	0.99 - 1.00	0.99 - 1.00
Glasses	0.93 - 0.99	0.87 - 0.96	0.91 - 1.00
Pimadata	0.78 - 0.79	0.79 - 0.80	0.80 - 0.81
Ionosfer	0.94 - 1.00	0.99 - 1.00	0.99 - 1.00
Heart_C	0.70 - 0.73	0.86 - 0.88	0.85 - 0.87
Heart_H	0.72 - 0.81	0.87 - 0.92	0.91 - 0.96
Letters	0.99 - 1.00	0.99 - 1.00	0.99 - 1.00

Tabla 3.9. Intervalos de confianza para la exactitud media con una confiabilidad del 95 % para los clasificadores no lineales

La figura 3.7 muestra la comparación del tiempo tomado por los 3 métodos para ejecutar las 15 iteraciones, contra el error medio obtenido en la clasificación. Se puede notar cómo, en la mayoría de casos RLSC-2DGCV y RLSC-GCV iterativo, compensa el tiempo tomado en la ejecución con el bajo error de clasificación. Las gráficas poseen las mismas características mencionadas para el caso lineal.

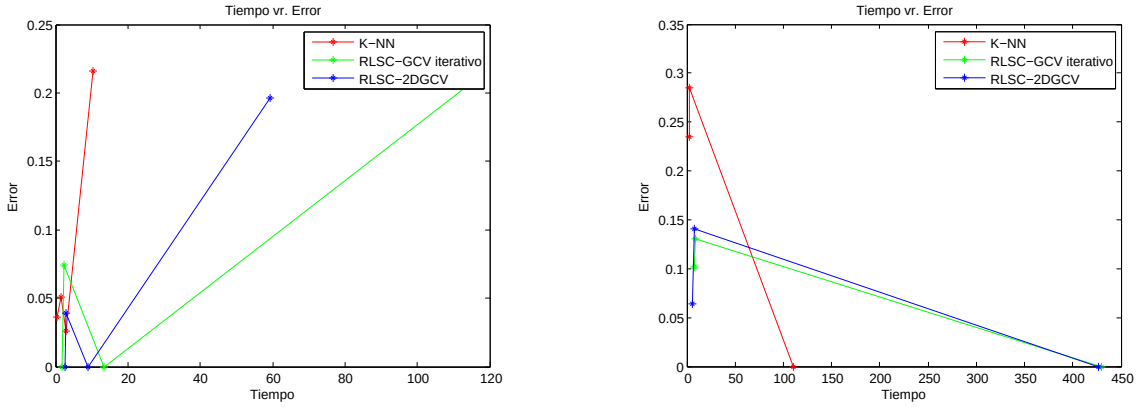


Figura 3.7. Tiempo vr. Error de los clasificadores no lineales

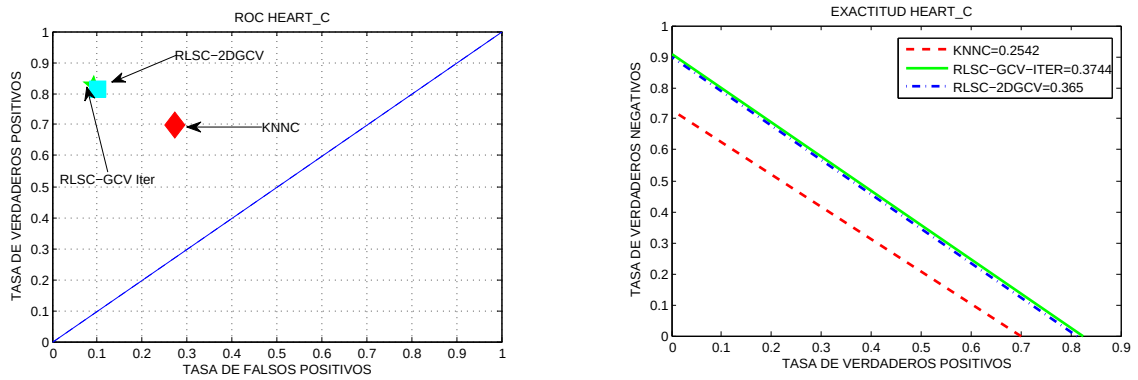


Figura 3.8. A la izquierda se encuentra la gráfica ROC para la base de datos Heart_c y a la derecha la linea de exactitud de la misma base de datos

La gráfica 3.8 permite ver el espacio ROC y la línea de exactitud lograda por los tres clasificadores con la base de datos Heart_c. Se puede notar en la gráfica del espacio ROC, como se genera un empate entre RLSC-GCV iterativo y RLSC-2DGCV, y de qué forma estos dos métodos superan en precisión al K-NN, puesto que se acercan en gran medida al punto de clasificación perfecta, tienden a generar una tasa muy alta de verdaderos positivos y una muy baja de falsos negativos.

La gráfica de Exactitud para heart_c revela que efectivamente estos dos métodos superan en gran medida al clasificador K-NN, es decir se genera un balance bastante satisfactorio entre la proporción de datos correctamente clasificados, tanto positivos como negativos.

CLASIFICADOR	VIN	VID	PIM	ION	HE_C	HE_H	LET
K-NN	0.45	0.47	0.30	0.48	0.25	0.29	0.50
RLSC-GCV ITER	0.50	0.43	0.27	0.50	0.37	0.38	0.50
RLSC-2DGCV	0.50	0.46	0.28	0.50	0.37	0.42	0.50

Tabla 3.10. Áreas bajo la línea de exactitud de los clasificadores no lineales

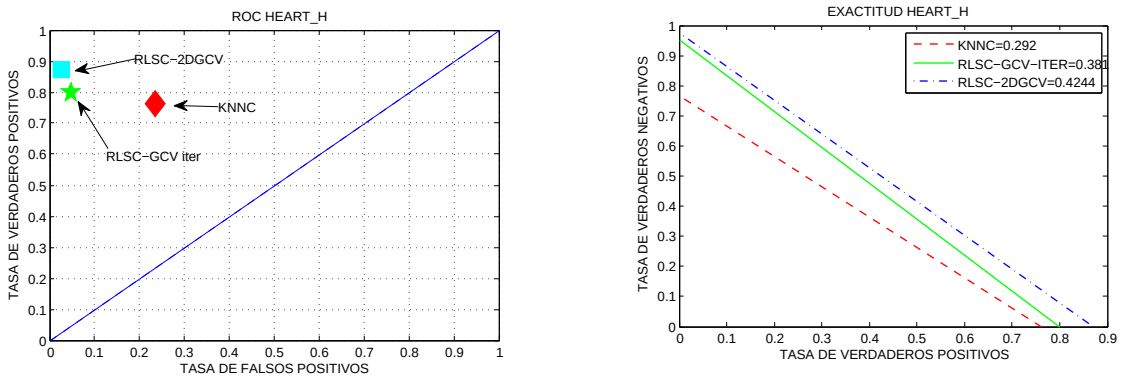


Figura 3.9. A la izquierda se encuentra la gráfica del espacio ROC para la base de datos Heart_c y a la derecha la línea de exactitud de la misma base de datos

Por su parte, la grafica 3.9 muestra en el espacio ROC, cómo RLSC-2DGCV es superior en precisión, y tiende a generar una clasificación casi perfecta, lo cual también se ve reflejado en la línea de exactitud mostrada por el método.

Las áreas obtenidas bajo las líneas de exactitud consignadas por todos los métodos y con todas las bases de datos, se muestran en la tabla 3.10. En ella se destacan los métodos RLSC-GCV iterativo y RLSC-2DGCV, que logran clasificación perfecta para 3 de las 7 bases de datos.

Como se muestra en la tabla 3.11, efectivamente el área media lograda es superior para los métodos RLSC-GCV.

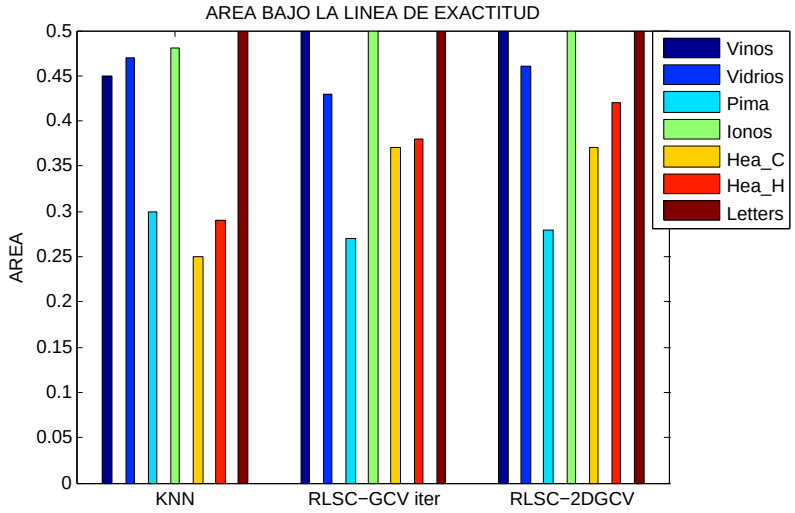


Figura 3.10. Diagrama de barras para el área bajo la línea de exactitud, clasificadores no lineales

	K-NN	**RLSC-GCV iter**	**RLSC-2DGCV**
Área Promedio	0.39	0.42	0.43

Tabla 3.11. Área Promedio bajo la línea de exactitud de los métodos de clasificación no lineales

4 Conclusiones

El algoritmo de clasificación de mínimos cuadrados regularizados basado en kernel es una técnica bastante prometedora, RLSC minimiza un funcional regularizado directamente en un RKHS definido por un kernel. Esta técnica involucra dos consideraciones importantes, las cuales tienen que ver con la selección del parámetro de regularización y parámetros del kernel cuando se trabaja con la extensión no lineal del método.

Se desarrolló el marco teórico del método RLSC empleando GCV para realizar la escogencia del parámetro de regularización de manera automática y se probó la eficacia del mismo, comparándolo contra otros métodos de clasificación, tales como LDA y SVM. En estas pruebas se concluyó que RLSC-GCV ofrece las ventajas de tener un bajo costo computacional, trabajar con datos que provienen de problemas mal condicionados para general soluciones únicas y estables. Además, al ser una versión automática, ofrece realizar las tareas de clasificación de manera práctica, teniendo como base única los datos y sus respectivas etiquetas.

Otro aporte a la base lineal del método RLSC fué la automatización de la escogencia del parámetro de regularización empleando LOO. Pero es importante resaltar el alto costo computacional que implica utilizar este método unicamente es recomendable cuando se cuenta con un pequeño conjunto de datos.

El aporte más importante consiste en el desarrollo del marco teórico para la escogencia de parámetros simultáneos, empleando GCV, es decir, se logró implementar para el método de RLSC, la escogencia automática de los parámetros de regularización y parámetros del kernel, lo cual evita sesgos en los errores de estimación.

Se verifica entonces, la capacidad del método RLSC para lidiar con problemas de clasificación en entornos lineales y no lineales.

Fueron implementadas dos versiones de RLSC-GCV no lineal, una de ellas la versión iterariva, que se vale de una serie corta de pasos para la selección de parámetros y una versión llamada RLSC-2DGCV que emplea la función GCV multidimensional, con dependencia de 2 parámetros.

Se probó en ambos casos que los resultados para el desempeño de la clasificación son similares. Estas implementaciones de GCV permiten entonces, escoger de manera automática y competente los parámetros requeridos para la clasificación con RLSC.

El análisis de resultados empleando los intervalos de confianza, permite establecer que el promedio de la exactitud de RLSC con respecto a los otros clasificadores, se encuentran en rangos aceptables con una confiabilidad del 95 % y el análisis del tiempo vr. error permite establecer la ganancia en costo y exactitud de RLSC con respecto a sus competidores.

Bibliografía

[1] Ancona, N., Maglietta, R., D'Addabbo, A., Liuni, S. y Pesole, G. *Regularized Least Squares Cancer classifiers from DNA microarray data*. BMC Bioinformatics, 6, 2005.

[2] Bauer, F., Pereverzev, S. y Rosasco, L. *On regularization algorithms in learning theory*. Journal of Complexity, 23:52–72, 2007.

[3] Calvetti, D., Morigi, S., Reichel, L. y Sgallari, F. *Tikhonov regularization and the L-curve for large discrete ill-posed problems*. Journal of Computational and Applied Mathematics, 123:423–446, 2000.

[4] Cawley, G.C. y Talbot, N. L.C. *Efficient leave-one-out cross-validation of kernel Fisher discriminant classifiers*. Pattern Recognition, 36:2585–2592, 2003.

[5] Cesa-Bianchi, N. *Applications of Regularized Least Squares to Pattern Classification*. Theoretical Computer Science, 382:221– 231, 2007.

[6] Chapelle, O., Vapnik, V., Bousquet, O. y Mukherjee, S. *Choosing Multiple Parameters for Support Vector Machines*. Machine Learning,, 46:131–159, 2002.

[7] Duin, R.P.W. *PRTools, Version 3.0: A Matlab Toolbox for Pattern Recognition*. Pattern Recognition Group, P.O. Box 5046, 2600 GA Delft, January 2000.

[8] Fawcett, T. *Using Rule Sets to Maximize ROC Performance*. En *IEEE International Conference on Data Mining*, 2001.

[9] Fawcett, T. *An introduction to ROC analysis*. Pattern Recognition Letters, 27:861874, 2006.

[10] Golub, G.H., Heath, M. y Wahba, G. *Generalized Cross Validation as a Method for Choosing a Good Ridge Parameter*. Technometrics, 21:–, 1979.

[11] Golub, G.H. y Matt, U.V. *Generalized Cross-Validation for Large-Scale Problems*. Journal of Computational and Graphical Statistics, 6:1–34, 1997.

[12] Hansen, P.C. *Rank-Deficient and Ill-Posed Problems,*. SIAM, 1997.

[13] Hansen, P.C., Nagy, J.G. y O'Leary, D.P. *Deblurring Images: Matrices, Spectra and Filtering*. SIAM, 2006.

[14] Hastie, T., Tibshirani, R. y Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Satatistics, 2001.

[15] Kilmer, M.E. y O'Leary, D.P. *Choosing Regularization parameters in Iterative Methods for Ill-Posed Problems*. SIAM J. Matrix Anal. Appl., 22:1204–1221, 2001.

[16] Kreyszing, E. *Introductory Functional Analysis with Applications*. Wiley, 1989.

[17] Lukas, Mark A. *Asymptotic optimality of generalized cross-validation for choosing the regularization parameter*. Numerische Mathernatik. Springer-Verlag, 66:41–66, 1993.

[18] Meyer, C.D. *Matrix Analysis and Applied Linear Agebra*. 2004.

[19] Montgomery, D. C. y Runger, G. C. *Applied statistics and probability for engineers*. 2002.

[20] Rifkin, R., Yeo, G. y Poggio, T. *In Advances in Learning Theory: Methods, Model and Applications, NATO Science Series III: Computer and Systems Sciences*, volumen 190. Suykens BM Horvath. Vandewalle, Amsterdam, 2003.

[21] Rifkin, R. M. y Lippert, R. A. *Notes on Regularized Least Square. Computer Science and Artificial Inteligence Laboratory Technical Report*. Informe técnico, MIT-CSAIL-TR, 2007.

[22] Rifkin, R.M. *Everything Old is New Again: A fresh Look At Historycal Aproaches in Machine Learning*. Tesis de Doctorado, Massachusetts Institute of Technology, 2002.

[23] Schölkopf, B., Burges, C. J. y Smola, A. J. *Advances in Kernel Methods*. The MIT Press, 1998.

[24] Schölkopf, B., Herbrich, R. y Smola, A. J. *Generalized Representer Theorem*. Lecture Notes in Computer Science. Springer Berlin /Heidelberg, 2111:416–426, 2001.

[25] Schölkopf, B. y Smola, A. J. *Learning with Kernel: Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press, 2002.

[26] Vito, E. D., Caponnetto, A. y Rosasco, L. *Model Selection for Regularized Least-Squares Algorithm in Learning Theory*. Foundation Computational Mathematics, páginas 59–85, 2005.

[27] Wu, Y., Ianakiev, K. y Govindaraju, V. *Improved k-nearest neighbor classification*. Pattern Recognition, 35:2311–2318, 2002.

[28] Xu, P. *Iterative generalized cross-validation for fusing heteroscedastic data of inverse ill-posed problems*. Geophys. J. Int., 179:182–200, 2009.

[29] Xue, H., Chen, S. y Yang, Q. *Discriminatively Regularized Least-Squares Classification*. Pattern Recognition, 42:93 – 104, 2009.

[30] Zhang, P. y Peng, J. *Efficient Regularized Least Squares Classification*. En *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2004.

[31] Zhang, P. y Peng, J. *SVM vs Regularized Least Squares Classification*. En *Proceedings of the 17th International Conference on Pattern Recognition*, 2004.