Universidad Centroccidental "Lisandro Alvarado"
Decanato de Ciencias y Tecnología
Licenciatura en Ciencias Matemáticas

Trabajo Especial de Grado

MÁQUINAS DE VECTORES SOPORTE CON COMBINACIÓN CONVEXA DE FUNCIONES KERNEL

Br. Graccyela R. Salcedo P.

Tutor: Dr. Javier Hernández
Área de Conocimiento: Optimización

Resumen

Dadas Funciones Kernel, las relacionamos a través de combinación convexa, agregando ésta última como la Función Kernel de la máquina de soporte vectorial (SVM), y los parámetros de dicha combinación como nuevas variables en el problema. Luego de resolver el problema, de manera directa o por vía de esquema alternante, obtenemos una combinación convexa adecuada para la clasificación de la data en cuestión, lo que nos permite conocer cual de las Funciones Kernel que estemos combinando es más efeciente para clasificar la data.

A Norma, Vicky y a Oliver.

Índice general

Agradecimientos

Agradezco primeramente a Dios, porque sin él nada es posible.

Agradezco a mi madre Norma por su apoyo incondicional y su inmenso amor, a mi sobrina Vicky por su ternura y amor, a mis exitosas hermanas Gabriela y Veronica por su gran ejemplo. A Oliver por sus consejos, por su amor y por tanto apoyo.

Agradezco a mi primer tutor Hugo Lara por sus ideas y su confianza. A mi actual tutor Javier Hernández por sus consejos y por su dedicación. A Jacobo por toda su colaboración.

Finalmente, agradezco a los grandes profesores que tuve a lo largo de mi carrera, por enseñarme la rigurosidad matemática.

Introducción

Las *máquinas de soporte vectorial (Support Vector Machine, SVM)* son un conjunto de algoritmos de aprendizaje supervisado desarrollado por Vapnik [8] y su grupo de trabajo a principio de los años 80. La versatilidad de las SVM las hace aplicables en áreas muy variadas como: en visión artificial para el reconocimiento de caracteres, rostros, matrículas y objetos; en medicina, para la clasificación de exámenes radiológicos, TAC (Tomografía Axial Computarizada), y para el diagnóstico de tejido humano; en genética para predicción de genes; en la clasificación de documentos, entre muchas otras áreas.

En este trabajo estudiamos las máquinas de soporte vectorial como un problema de optimización matemática, enfocándonos en uno de sus aspectos fundamentales *las funciones kernel*. Implementar una función kernel adecuada al momento de clasificar una data con máquinas de soporte vectorial, puede no ser tan fácil. Por lo que proponemos agregar al problema una combinación convexa de funciones kernel conocidas, y construimos soluciones numéricas para problemas específicos, con datos de prueba.

El resto del presente documento se describe a continuación: En el primer capítulo presentamos algunos elementos de revisión bibliográfica. Temas de optimización continua, dualidad, condiciones de optimalidad, funciones kernel y máquinas de soporte vectorial para problemas separables y no separables; son brevemente discutidos. En el capítulo 2 exponemos el programa propuesto, para solventar el problema de la elección de un kernel. Y por último, en la sección 4 presentamos experimentos numéricos realizados con diferentes tipos de datas.

Capítulo 1

Preliminares

En esta sección mostramos algunos elementos teóricos tomados de libros como [7] y [3].

1.1. El Problema Dual

Para todos los problemas de programación lineal hay un problema compañero, llamado el "dual" programa lineal, en el que se invierten los papeles de variables y restricciones. Es decir, para cada variable en el programa lineal original o "primal", hay una restricción en el problema dual, y para cada restricción en el primal hay una variable en el dual.

En una aplicación, las variables en el problema primal podrían representar productos, y los coeficientes objetivos podrían representar los beneficios asociados con la fabricación de los productos. De ahí que el objetivo en el primal indica directamente cómo un aumento en la producción afecta a las ganancias. Las restricciones en el problema primal podrían representar la disponibilidad de materias primas. Un aumento en la disponibilidad de materias primas podría permitir un aumento de la producción, y por lo tanto un aumento en sus ganancias, pero esta relación no es tan fácil deducir desde el problema primal. Uno de los efectos de la teoría de la dualidad es hacer explícito el efecto de los cambios en las restricciones sobre el valor del objetivo. Es debido en esta interpretación que las variables en el problema dual a

veces son llamados “precios sombra”, ya que miden los “costos” implícitos asociados a las restricciones. Si bien es posible definir un dual de cualquier programa lineal, la simetría de los dos problemas es más evidente cuando el programa lineal está en su forma canónica. Un problema de minimización está en forma canónica si todas las restricciones del problema son del tipo $\geq$, y todas las variables son no negativas:

$$\begin{aligned} Minimizar\ z &= c^T x \\ sujeto\ a\ Ax &\geq b \\ x &\geq 0 \end{aligned}$$

Nos referiremos a este problema original como el programa lineal primal. El programa lineal dual correspondiente tendrá la forma:

$$\begin{aligned} Minimizar\ w &= b^T y \\ sujeto\ a\ A^T y &\leq c \\ y &\geq 0 \end{aligned}$$

Si el problema primal tiene n variables y m restricciones, entonces el problema dual tendrá m variables (una variable dual para cada restricción primal) y n restricciones (una restricción dual para cada variable primal). Los coeficientes en el objetivo del primal son los coeficientes de la parte derecha del dual, y viceversa. La matriz de restricciones en el dual es la transpuesta de la matriz en el primal.

El problema dual es un problema de maximización, donde todas las restricciones son del tipo $\geq$, y todas las variables son no negativas. Esta forma se conoce como la forma canónica para un problema de maximización.

1.2. Condiciones de Optimalidad KKT

Las condiciones KKT, (Karush (1939),Kuhn y Tucker (1951)) nos aportan condiciones necesarias y suficientes para la optimalidad del programa lineal dual y del primal, en el siguiente teorema:

Teorema: Las siguientes condiciones son equivalentes:

a. x^* es solución del problema primal:

$$\begin{aligned} Minimizar\ w &= c^T x \\ sujeto\ a\ A^T y &\leq b \\ x &\geq 0 \end{aligned}$$

b. x^* es primal factible, algún u^* es dual factible y $c^t x^* = b^T u^*$; esto es, el valor objetivo del primal en algún punto factible primal es igual al valor objetivo del dual en algún punto factible dual. Por tanto x^T es la solución optima del primal y u^T es la solución optima del dual.

c. x^* es primal factible, algún u^* es dual factible y $c^T x^* \leq b^T u^*$; esto es, el valor objetivo del dual es menor o igual que el valor objetivo del dual.

d. Existen x^* y u^* satisfaciendo, $Ax^* \geq b$, $x^* \geq 0$, $A^T u^* \leq c$,$u^* \geq 0$, y $x_i^*(A_i x^* - b_i) = 0$, $x_j^*(c_j - A_j^T u^*) = 0$, para todo i, j.

e. Existen x^* y u^* satisfaciendo, $Ax^* \geq b$, $x^* \geq 0$, $A^T u^* \leq c$,$u^* \geq 0$, y $u^{T*}(Ax^* - b) + x^{T*}(c - A^T u^*) = 0$.

1.3. Funciones Kernel

La función kernel K implícitamente define una aplicación no lineal de $\mathbb{R}^n$ a algún espacio $\mathbb{R}^h$ donde h puede ser mucho más grande que n.

Formalmente, una función kernel se define como:

Definición **1 (Función kernel (ver [9]))** *Una función kernel es un producto interno en el espacio de características que tiene su equivalente en el espacio de entrada:*

$K(x,x^T) = \phi(x)^T \phi(x^T)$,

donde ϕ es alguna función de transformación implicita. K es una función simétrica definida positiva y debe cumplir:

Para toda función g tal que $\int g(x)^2 dx < \infty$*, se tiene que:* $\int K(x,y)g(x)g(y)dxdy \geq 0$.

Esta condición es llamada la ***Condición de Mercer****.*

Observe que, si K_1 y K_2 son funciones Kernel, entonces cualquier combinación lineal convexa de ellas también lo será, esto último por la linealidad de la integral. Es decir, si K_1 y K_2 son funciones Kernel, $a_1K_1 + a_2K_2$, con a_1,$a_2 \geq 0$ y a_1+a_2=1 también es una función kernel.

Definición **2 (Matriz kernel (ver [10]))** *Dado un conjunto finito de entrada* $\mathcal{X} = \{x_i\}_{i=1}^n$*. Una matriz kernel es una matriz cuadrada* $\mathrm{K} \in \mathbb{R}^{n\times n}$ *tal que* $\mathrm{K}_{ij} = K(x_i, x_j)$ *con* $x_i, x_j \in \mathcal{X}$ *y* K *una función kernel.*

A continuación, las funciones kernel más comúnmente usadas:

- **Kernel Polinomial no homogéneo**:

$$K(x_i, x_j) = (\langle x_i, x_j\rangle + c)^d, \text{ con } d \in \mathbb{N},\ c \geq 0.$$

  El *kernel lineal*, es un kernel polinomial con grado d=1.

- **Kernel perceptron**:

$$K(x_i, x_j) = \|x_i - x_j\|.$$

- **Kernel Gaussiano**:

$$K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right), \text{donde } \sigma > 0.$$

- **Sigmoide o kernel hiperbólico**:

$$K(x_i, x_j) = \tanh(\delta\langle x_i, x_j\rangle + \theta), \text{ donde } \delta > 0,\ \theta < 0.$$

1.4. Máquinas de vectores soporte

Las máquinas de soporte vectorial o máquinas de vectores de soporte (Support Vector Machine, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollado por Vapnik (ver [8]) y su grupo de trabajo a principio de los años 80. La versatilidad de las SVM las hace aplicables en áreas muy variadas como: en visión artificial para el reconocimiento de caracteres, rostros, matrículas y objetos; en medicina, para la clasificación de exámenes radiológicos, TAC (Tomografía Axial Computarizada), y para el diagnóstico de tejido humano; en genética para predicción de genes; en la clasificación de documentos, entre muchas otras áreas.

Una SVM es un modelo que representa los puntos de una muestra en un espacio, separando estos puntos en clases y separándolos entre sí con la mayor distancia posible, es decir, con un margen máximo. Se consideran dos conjuntos de puntos en $\mathbb{R}^n$ los cuales se etiquetan con P_+ (clase del +1) y P_- (clase del -1). Al denotar cualquiera de estos puntos por x, se puede construir una función f tal que $f(x) > 0$ si $x \in P_+$ y $f(x) < 0$ si $x \in P_-$. La función f es conocida como un clasificador. Así, dado un nuevo punto x, se puede usar f para clasificar x como perteneciente a P_+ (si $f(x) > 0$) o a P_- (si $f(x) < 0$).

Se inicia con la construcción de un clasificador lineal, el cual tiene la forma $f(x) = \omega^T x - \gamma$ donde $\omega \in \mathbb{R}^n$ y $\gamma \in \mathbb{R}$. La situación ideal es que el hiperplano definido por $f(x) = 0$ pudiese separar por completo los conjuntos P_+ y P_-, y de esta forma se obtendría que $f(x) > 0$ para $x \in P_+$ y $f(x) < 0$ para $x \in P_-$. Si tal (ω, γ) existe, se puede redefinir (ω, γ) como:

$$\frac{(\omega, \gamma)}{\text{mín}_{x \in P_+ \cup P_-} |\omega^T x - \gamma|}. \tag{1.4.1}$$

Así, por (1.4.1) se tiene que:

$$\begin{aligned} x \in P_+ \quad &\Rightarrow \quad f(x) = \omega^T x - \gamma \geq 1, \\ x \in P_- \quad &\Rightarrow \quad f(x) = \omega^T x - \gamma \leq -1. \end{aligned} \tag{1.4.2}$$

Ahora, para expresar estas condiciones como un sistema de desigualdades lineales, agrupamos cada uno de los puntos en una fila de la matriz A, la cual tiene $m = |P_+| + |P_-|$ filas, donde $|P_+|$ y $|P_-|$ denotan el número de puntos en P_+ y P_- respectivamente. Luego, se define una matriz diagonal D que etiqueta cada fila de A como:

$$D_{ii} = \begin{cases} 1, & \text{si el punto representado por la fila i de } A \text{ pertenece a } P_+; \\ -1, & \text{si el punto representado por la fila i de } A \text{ pertenece a } P_-. \end{cases}$$

Las condiciones (1.4.2) pueden ser escritas como:

$$D(A\omega - e\gamma) \geq e, \tag{1.4.3}$$

donde $e = (1, 1, \ldots, 1)^T$. La figura (1.4), muestra un hiperplano separador (ver [**?**]) $\omega^T x = \gamma$ para dos conjuntos de puntos, obtenidos hallando un par (ω, γ) que es factible para (1.4.3) junto a los hiperplanos soporte $\omega^T x - \gamma = \pm 1$.

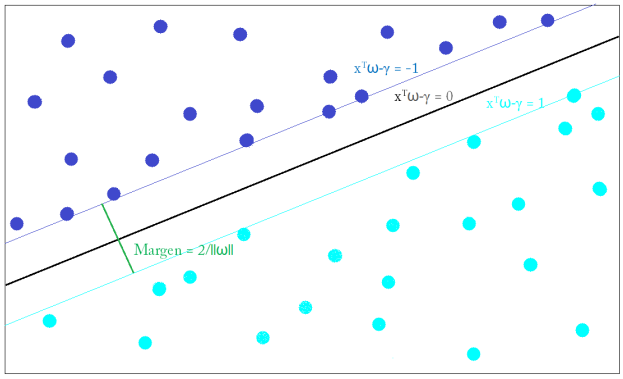

Figura 1.1: Caso linealmente separable

Ahora, si es posible separar las dos clases de puntos, entonces es deseable maximizar la distancia (margen) entre los hiperplanos soportes, los cuales en la Figura 1.1,

son representados por dos lineas azules, la azul claro y la azul oscuro, correspondiente a cada data. Se puede mostrar que el margen de separación para cualquier norma $\|\cdot\|$ es

$$\frac{2}{\|\omega\|'},$$

donde $\|\omega\|'$ denota la norma dual (para detalles ver ([3])). Al utilizar la norma Euclídea la cual es su misma dual, entonces la maximización de $2/\|\omega\|'$ puede ser alcanzada por la minimización de $\|\omega\|$ o $\|\omega\|^2 = \omega^T\omega$. Por lo tanto, podemos resolver el siguiente programa para hallar el hiperplano separador con margen (Euclídeo) máximo:

$$\begin{array}{ll} \underset{\omega,\gamma}{\text{mín}} & \frac{1}{2}\omega^T\omega \\ s.a. & D(A\omega - e\gamma) \geq \quad e. \end{array} \tag{1.4.4}$$

Los vectores soporte son los puntos x que están sobre los hiperplanos soporte que corresponden a los multiplicadores de Lagrange de las restricciones de (1.4.4) que son positivos. Estos corresponden a restricciones activas en (1.4.4). Es decir, tras obtener la solución se pueden determinar los vectores soporte mediante la condición de complementaridad de las condiciones KKT, ya que aquellos puntos en los que su multiplicador de Lagrange asociado cumple $\lambda_i \geq \epsilon > 0$, serán seleccionados como vectores soporte, mientras que los que tengan su multiplicador de Lagrange $\lambda_i \approx 0$ se descartan.

En la práctica, usualmente no es posible hallar un hiperplano que separe los dos conjuntos porque no existe tal hiperplano. En tales casos, el programa cuadrático (1.4.4) es infactible, pero se pueden definir otros problemas que identifiquen hiperplanos de separación *tan cerca de separar las clases como sea posible*, en algún sentido. Para ello, se puede definir un vector y cuyas componentes indican la cantidad por la cual las restricciones son válidas como se sigue:

$$D(A\omega - e\gamma) + y \geq e, \quad y \geq 0. \tag{1.4.5}$$

Se puede medir la violación total sumando las componentes de y, y sumando algunos múltiplos de esta cantidad del objetivo (1.4.4), para obtener

$$\begin{array}{ll} \underset{\omega,\gamma,y}{\text{mín}} & \frac{1}{2}\omega^T\omega + \nu e^T y \\ s.a. & D(A\omega - e\gamma) + y \geq \quad e, \quad y \geq 0, \end{array} \tag{1.4.6}$$

donde ν es algún parámetro positivo. El problema (1.4.6) se refiere a una SVM (lineal).

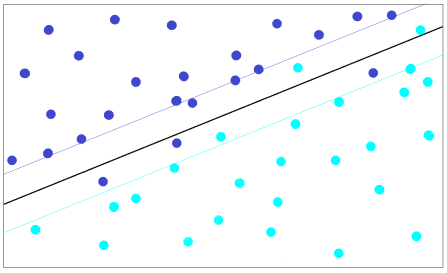

Figura 1.2: Caso linealmente no separable

La figura anterior, muestra dos conjuntos de puntos linealmente no separables y el hiperplano obtenido al resolver el problema (1.4.6). Los hiperplanos soportes son también mostrados. En esta formulación, los vectores soportes son los puntos desde cada conjunto P_- y P_+ que se encuentran en el lado erróneo de los hiperplanos delimitadores (o están en sus respectivos planos de delimitación y sus correspondientes multiplicadores de Lagrange son positivos). Nuevamente, estos puntos corresponden a restricciones activas del conjunto de restricciones $D(A\omega - e\gamma) + y \geq e$.

Usando u para denotar los multiplicadores de Lagrange para las restricciones $D(A\omega - e\gamma) + y \geq e$ en (1.4.6), las condiciones KKT para este problema son:

$$\begin{aligned} 0 &= \omega - A^T D u, \\ 0 &= e^T D u, \\ 0 &\leq \nu e - u \perp y \geq 0, \\ 0 &\leq D(A\omega - e\gamma) + y - e \perp u \geq 0. \end{aligned}$$

Estas son las condiciones KKT para el siguiente problema el cual es el dual de (1.4.6)

$$\begin{array}{ll} \min\limits_{u} & \frac{1}{2}u^T DAA^T Du - e^T u \\ s.a. & 0 \leq u \leq \nu e \quad , \; e^T Du = 0. \end{array} \tag{1.4.7}$$

De hecho, a menudo es más conveniente resolver esta forma del problema dual para u y entonces recuperar ω colocando $\omega = A^T Du$ (a partir de la primera condición KKT) y recuperando γ como el multiplicador de Lagrange para la restricción $e^T Du = 0$ en (1.4.7).

Se puede obtener un programa lineal alternativo a (1.4.6) reemplazando el término cuadrático $\frac{1}{2}\omega^T\omega$ por la norma-1 $\|\omega\|_1$ la cual corresponde a la medición del margen usando la norma-∞. Introduciendo un vector s cuyos elementos son valores absolutos de los correspondientes elementos de ω, obtenemos las siguientes formulaciones:

$$\begin{array}{ll} \min\limits_{\omega,\gamma,s,y} & e^T s + \nu e^T y \\ s.a. & D(A\omega - e\gamma) + y \geq \quad e, \quad s \geq \omega \geq -s, \quad y \geq 0. \end{array} \tag{1.4.8}$$

Una formulación alternativa para (1.4.8) que parece ser más efectiva computacionalmente es el siguiente programa lineal, el cual podría ser resuelto de forma eficiente por el método simplex, o métodos de puntos interiores:

$$\begin{array}{lll} \min\limits_{\omega^+,\omega^-,\gamma,y} & e^T(\omega^+ + \omega^-) + \nu e^T y & \\ s.a. & D(A\omega - e\gamma) + y \geq & e \\ & \omega = \omega^+ - \omega^- & \\ & \omega^+, \omega^-, y \geq 0, & \end{array} \tag{1.4.9}$$

note que $\omega_i^+ + \omega_i^- \geq |\omega_i|$, y en la solución se tendría que al menos un ω_i^+ o ω_i^- es nulo $\forall i$.

Las formulaciones anteriormente mencionadas se refieren a SVM lineales, ya que tratan de separar los puntos por hiperplanos (lineales). Un enfoque más general involucra un mapeo de cada punto en un espacio de dimensión mucho mayor vía una función Φ y así poder determinar el hiperplano que separa en este espacio de dimensión mayor $\mathbb{R}^h$. Cuando este hiperplano es mapeado de nuevo en el espacio original

$\mathbb{R}^n$, él describe una superficie no lineal, dando mayor flexibilidad y un medio más potente para la clasificación.

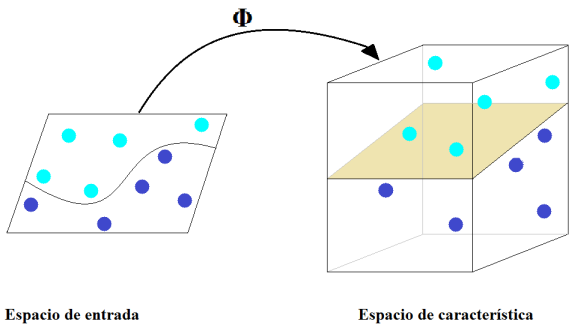

Figura 1.3: Función Φ

La dificultad en el cálculo de Φ y la posible alta dimensionalidad de $\mathbb{R}^h$ han sido importantes factores en el uso del kernel K como un generador de la superficie de separación no lineal en el espacio original de entradas $\mathbb{R}^n$ (ver [5]), pero la cual es lineal en el espacio de dimensión mayor $\mathbb{R}^h$.

Sea $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times l}$. El kernel $K(A, B)$ mapea $\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times l}$ en $\mathbb{R}^{m\times l}$. En particular si x e y son columnas en $\mathbb{R}^n$ entonces $K(x^T, A^T)$ es un vector fila en $\mathbb{R}^m$ ($x^T \in \mathbb{R}^{1\times n}, A^T \in \mathbb{R}^{n\times m}$), $K(x^T, y)$ es un número real y $K(A, A^T)$ es una matriz $m \times m$. Cuando se escribe $K(x, y)$ donde x, y son vectores nos referimos a K como una función, mientras que la expresión $K(A, A^T)$ es una matriz inducida por una función kernel, ya que sus entradas vienen dadas por la eveluación de la misma.

Ahora, involucremos el concepto de función kernel en las SVM. Sea $A \in \mathbb{R}^{m\times n}$ la representación matricial de m puntos en el espacio n-dimensional $\mathbb{R}^n$. Cada punto A_i, $i = 1, \ldots, m$ pertenece a la clase del 1 o a la clase del -1 dependiendo si D_{ii} es 1 o -1. La meta es tratar de discriminar entre las clases 1 y -1 por una superficie no lineal, inducida por algún kernel $K(A, A^T)$, como sigue:

$$K(x, A^T)Du = \gamma,$$

donde $K(x, A^T) \in \mathbb{R}^m$. Los parámetros $u \in \mathbb{R}^m$ y $\gamma \in \mathbb{R}$ son determinados resolviendo

un programa matemático, típicamente cuadrático o lineal, como ya se han mencionado anteriormente. Un punto $x \in \mathbb{R}^n$ es clasificado en la clase 1 o -1 de acuerdo a la función de decisión

$$K(x, A^T)Du - \gamma\,,$$

la cual vale 1 o -1 respectivamente.

A continuación, se plantea un programa matemático que genera tal superficie para un kernel general, una máquina de soporte vectorial general vendría dada por:

$$\begin{array}{ll} \min\limits_{u,\gamma,y} & \nu e^T y + f(u) \\ s.a. & D(K(A, A^T)Du - e\gamma) + y \geq \quad e \\ & y \geq 0, \end{array} \tag{1.4.10}$$

aquí f es alguna función convexa sobre $\mathbb{R}^m$, típicamente una norma o seminorma, y como ya se ha visto ν es algún parámetro positivo de los pesos del error de separación $e^T y$ contra el parámetro u de la superficie de separación.

La minimización de u puede ser interpretada en una de dos maneras, puede ser vista como la minimización de los vectores soporte, es decir, las restricciones en (1.4.10) con multiplicadores positivos. Una interpretación más convencional es que se trata de la maximización de alguna medida de la distancia o margen entre los planos de delimitación paralelos en $\mathbb{R}^h$, bajo hipótesis apropiadas, tal f es una función cuadrática inducida por un kernel definido positivo.

Ahora, consideremos máquinas de vectores de soporte que incluyan una estándar y las cuales son obtenidas colocando f de (1.4.10) como una función cuadrática convexa $f(u) = \frac{1}{2}u^T Hu$, donde $H \in \mathbb{R}^{m\times m}$ es alguna matriz simétrica definida positiva. El programa matemático (1.4.10) viene a ser un programa cuadrático convexo:

$$\begin{array}{ll} \min\limits_{u,\gamma,y} & \nu e^T y + \frac{1}{2}u^T Hu \\ s.a. & D(K(A, A^T)Du - e\gamma) + y \geq \quad e \\ & y \geq 0, \end{array} \tag{1.4.11}$$

El dual de Wolfe de este programa cuadrático convexo es:

$$\begin{array}{ll} \underset{r\in\mathbb{R}^m}{\text{mín}} & \frac{1}{2}r^T DK(A,A^T)DH^{-1}DK(A,A^T)Dr - e^T r \\ s.a. & e^T Dr = 0 \\ & 0 \leq r \leq \nu e. \end{array} \tag{1.4.12}$$

Además, la variable primal u está relacionada con la variable dual r por:

$$u = H^{-1}DK(A,A^T)Dr. \tag{1.4.13}$$

En el presente trabajo elaboramos dos algoritmos para resolver (1.4.11). Los cuales se aprecian en la siguiente sección. Cabe destacar, que para los experimentos númericos se trabaja con $H = I$.

Capítulo 2

Máquina de vectores soporte

La escogencia de una función Kernel adecuada para resolver una SVM, con una data determinada, es uno de los mas grandes problemas al momento de implementar máquinas de soporte vectorial. Es por ello, que proponemos considerar K en (1.4.11), como combinación convexa de funciones kernel dadas. Quedando (1.4.11) como sigue:

$$\begin{array}{ll} \underset{u,\gamma,y,t}{\text{mín}} & \nu e^T y + \frac{1}{2} u^T H u \\ s.a. & D(\Sigma_{i=1}^r t_i K_i(A, A^T) D u - e\gamma) + y \geq \quad e \\ & \Sigma_{i=1}^r t_i = 1 \\ & t_i, y \geq 0, \end{array} \tag{2.0.1}$$

Donde K_i, son funciones Kernel previamente conocidas. El parámetro t_i, puede ser interpretado como el porcentaje de que tan bueno es K_i en comparación al resto de las funciones kernel dadas para clasificar la data en cuestión.

La propuesta de considerar el kernel como combinación convexa de funciones kernel conocidas, no la planteamos para incluirla en el programa dual (1.4.12), porque estamos variando la función kernel, de acuerdo a los parámetros de la combinación convexa, lo que implicaría la selección automática del kernel con entradas más pequeñas y no como deseamos, el kernel que mejor separe la data, ya que el kernel se encuentra en la función obejtivo a minimizar.

2.1. Esquema Alternante

Para resolver (2.0.1), proponemos un esquema alternante como en [2] para ser computacionalmente más rápido. Consistiendo en lo siguiente:

Dadas r funciones kernel a combinar, las aplicamos a la matriz $A \in \mathbb{R}^{m\times n}$, donde A almacena m datos en $\mathbb{R}^n$ como en la sección (2.4), por lo que cada matriz kernel resultante $K_i \in \mathbb{R}^{m\times m}$.

Fijamos $t_0 \in \mathbb{R}^r$, con $\Sigma_{i=1}^r t_0(i) = 1$ calculamos $K_0 = \Sigma_{i=1}^r t_0(i)K_i$ y resolvemos el programa cuadrático:

$$\begin{array}{ll} \underset{u,\gamma,y}{\text{mín}} & \nu e^T y + \frac{1}{2}u^T H u \\ s.a. & D(K_0 Du - e\gamma) + y \geq \quad e \\ & y \geq 0, \end{array} \tag{2.1.1}$$

Para obtener valores de u, y e γ, luego fijamos solamente u_0 e γ_0 respectivamente, y seguidamente procedemos a resolver el nuevo programa lineal:

$$\begin{array}{ll} \underset{y,t}{\text{mín}} & \nu e^T y + \frac{1}{2}u_0^T H u_0 \\ s.a. & D(\Sigma_{i=1}^r t_i K_i) Du_0 - e\gamma_0) + y \geq \quad e \\ & \Sigma_{i=1}^r t_i = 1 \\ & t_i \geq 0, y \geq 0, \end{array} \tag{2.1.2}$$

Obteniendo así valores para t y y, fijamos t_0 como el último valor óptimo obtenido de t, para calcular $K_0 = \Sigma_{i=1}^r t_i K_i$, y procedemos a resolver nuevamente (2.1.1). El criterio de parada usado es realizar iteradamente el proceso, hasta que la diferencia entre la función objetivo de (2.1.1) y la función objetivo de (2.1.2) sea menor que cierta tolerancia $\epsilon \geq 0$.

Es interesante notar, que en este esquema el programa (2.1.2) es lineal, lo que acelera la solución al momento de iterar. Además, el programa (2.1.1) es un programa SVM estándar como en (1.4.11) con $K = K_0$ correspondiente a cada iteración. También observe que la variable y es calculada en los dos programas, esto porque de

no ser así en (2.1.2) la función objetivo sería constante, o bien, para minimizar la violación con respecto al hiperplano generado por $K = \Sigma_{i=1}^{r} t_i K_i$.

Capítulo 3

Experimentos Numéricos

Para implementar numéricamente el programa propuesto por esquema alternante en la sección (2.1), trabajamos en lenguaje MATLAB haciendo uso del código abierto MPT3 (Multi-Parametric Toolbox 3), basado en la caja de herramientas de Matlab para la optimización paramétrica, la geometría computacional y el control predictivo basado en modelos.

Es importante destacar que en los experimentos numéricos a continuación se consideró $H = I$, ν=0.02; para el criterio de parada del esquema alternante ϵ=0.0001 y 60 como número máximo de iteraciones. Como valor inicial t_0=(0.2,0.2,0.2,0.2,0.2) y las funciones kernel consideradas:

- $K_1(x_i, x_j) = \langle x_i, x_j \rangle$. (Kernel lineal)
- $K_2(x_i, x_j) = \langle x_i, x_j \rangle^2$. (Kernel cuadrático)
- $K_3(x_i, x_j) = \langle x_i, x_j \rangle^3$. (Kernel cúbico)
- $K_4(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2\right)$. (Kernel gaussiano con $\sigma = \frac{1}{\sqrt{2}}$)
- $K_5(x_i, x_j) = \tanh(\langle x_i, x_j \rangle)$.(Kernel sigmoide con $\delta = 1$, $\theta = 0$)

En los siguientes experimentos FO1 y FO2 refieren al valor de la función objetivo del programa (2.1.1) y (2.1.2) respectivamente.

3.1. Data separable linealmente

De forma aleatoria seleccionamos la DATA 1 (ver figura 3.1) de 500 datos en total. El esquema alternante propuesto en 29.093239 segundos y 8 iteraciones entre los programas (2.1.1) y (2.1.2), nos da t=(1,0,0,0,0), junto a los valores de las funciones objetivos de los programas respectivos 4.5177 y 4.5176. El valor de t nos dice inmediatamente que el kernel mas efectivo para separar la data 1 es únicamente el kernel lineal. En la figura 4 (a la derecha), se aprecia la data 1 separada con el kernel lineal.

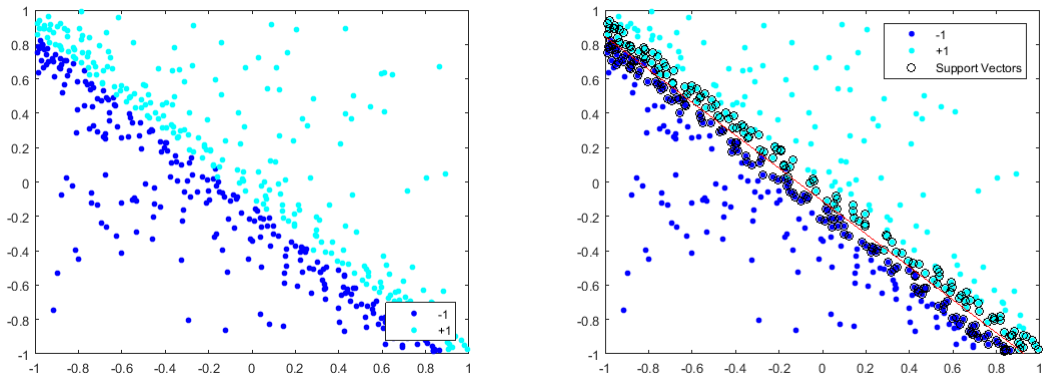

Figura 3.1: DATA 1

3.2. Data separable por una curva

Graficamente nuestra segunda data de 500 puntos:

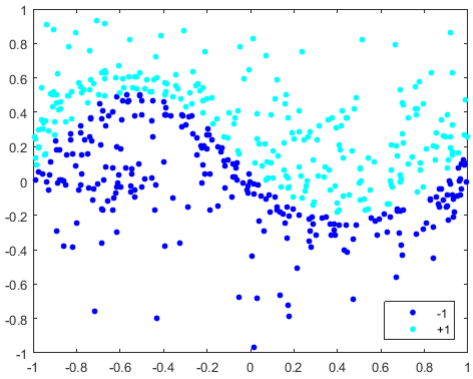

Figura 3.2: DATA 2

En este caso el programa (ver sección 2.1) demoró 99.745990 segundos y 58 iteraciones entre una función objetivo y la otra, para arrojar los siqueientes resultados FO1=6.4301, FO2=6.4302 y t=(0,0,0.3252,0.6748,0). Graficamente la separación generada para este valor de t se ve a la izquierda de la figura 3.3. Cambiando el kernel gaussiado de $K_4(x_i, x_j) = \exp(-\|x_i - x_j\|^2)$ a $K_4(x_i, x_j) = \exp(-2\|x_i - x_j\|^2)$ obtenemos una nueva separación como se refleja a la derecha de la figura 3.3, la que aparentemente es mejor por la disminución de vectores soporte. Este ejemplo numérico nos ilustra que los parametros de cada kernel tambien juegan un papel importante al momento de clasificar una data.

3.3. Data X

En forma de X, aleatoreamente simulamos una data de 500 puntos. Para esta data, el programa demoró un poco mas que en los experimentos anteriores 123.303924 segundos y alcanzando el número máximo de iteraciones, 60 iteraciones; con FO1=8.0638, FO2=8.0614 y t=(0.0722,0,0.5175,0.4103,0). La data y su respectiva separacón se puede observar en la figura 3.4.

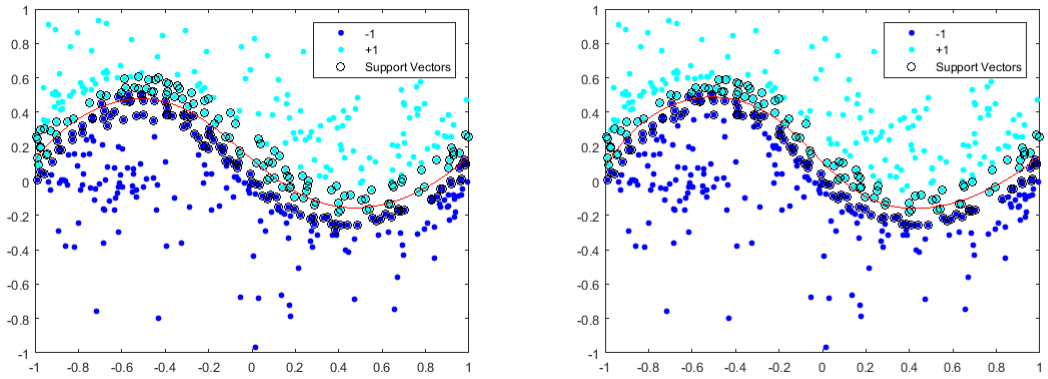

Figura 3.3: DATA 2 separada

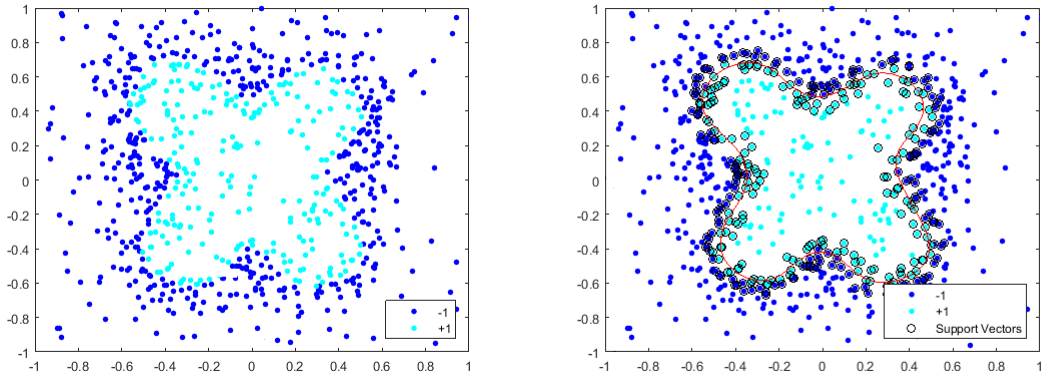

Figura 3.4: DATA 3

3.4. Data separable por una circunferencia

De 500 datos generamos uniformemente la data 4, de los cuales 250 se cuentran dentro de la circuenferencia de radio 0.5, centrada en (0,0). En 94.471107 segundos y 60 iteraciones entre una función objetivo y la otra, obtuvimos FO1=5.8309, FO2=5.8228 y t=(0,0,0,0.3379,0.6621). La separación respectiva a este resultado se aprecia graficamente en la siguiente figura:

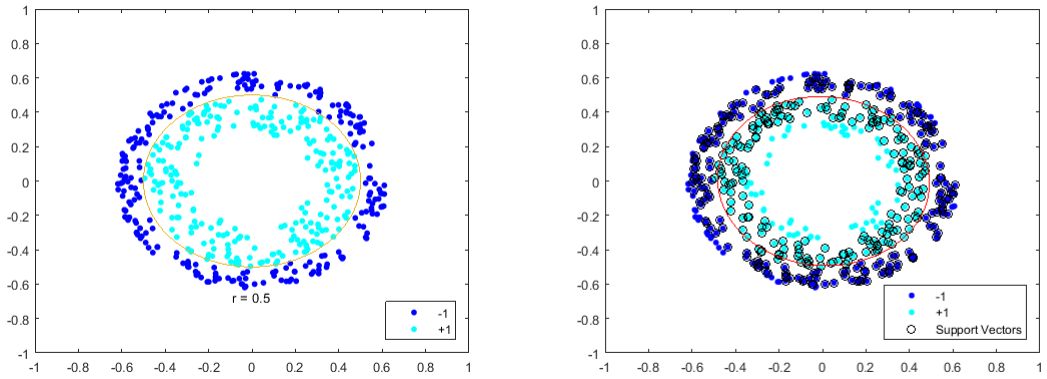

Figura 3.5: DATA 4

3.5. Data por cuadrante

La data 5, fue simulada uniformemente con 130 datos en cada cuadrante. La solución se obtuvo en 9.131293 segundos y 2 iteraciones entre las dos funciones objetivos, FO1=1.8154, FO2=1.8149, y el resultado más interesante t=(0,1,0,0,0) este valor de t implica que entre los 5 kernel con los que estamos experimentando, el kernel cuadrático es suficiente para separar esta data. Ver la siguiente figura:

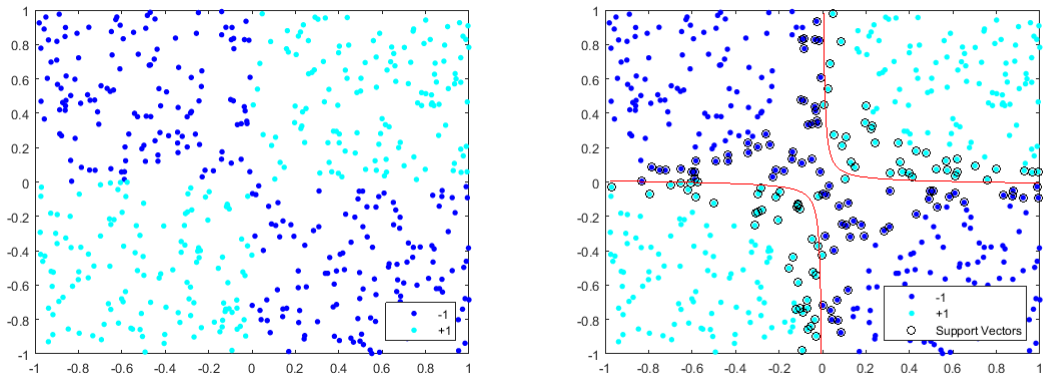

Figura 3.6: DATA 5

3.6. Data 2x3

La data 6 es de 504 puntos, 42 en cada rectángulo. FO1=4.5954, FO2=4.5944 t t=(0,0,0.6074,0.3926,0) fueron obtenidos en 30.763853 segundos y 6 iteraciones.

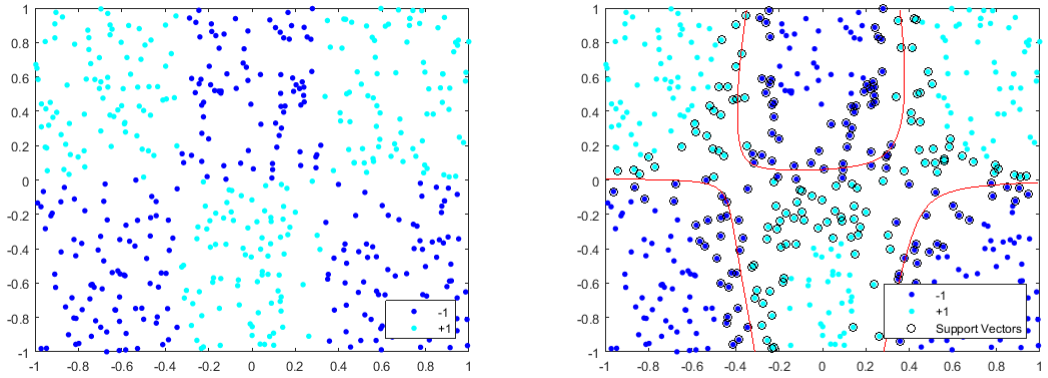

Figura 3.7: DATA 6

3.7. Data 3x3

Simulamos una data uniformente 513 datos, 57 en cada cuadro. El programa obtuvo una solución óptima en 19 iteraciones y 64.884794 segundos: FO1=8.1744, FO2=8.1743 y t=(0,0.4298,0,0.3401,0.2301). Para este valor de t la data 7 se obtiene una separación con $K_4(x_i, x_j) = \exp(-20\|x_i - x_j\|^2)$, como se ve a la derecha de la figura 3.8.

Una data muy interesante es la data 4x4, conocida como la data ajedrez. Esta data por seguir el estilo de la data 3x3, debería poder separarse con el mismo kernel de la data 3x3, por lo que en este trabajo simulamos una data en 4x4 cuadros de tamaño total 800 (50 puntos en cada cuadrado) y la separamos con el predictor generado con t=(0,0.4298,0,0.3401,0.2301), es decir $k = 0{,}4298K_2 + 0{,}3401K_4 + 0{,}2301K_5$, manteniendo $K_4(x_i, x_j) = \exp(-20\|x_i - x_j\|^2)$ y k_2,k_5 como al principio de este capítulo. La curva de separación obtenida se aprecia en la figura 3.9.

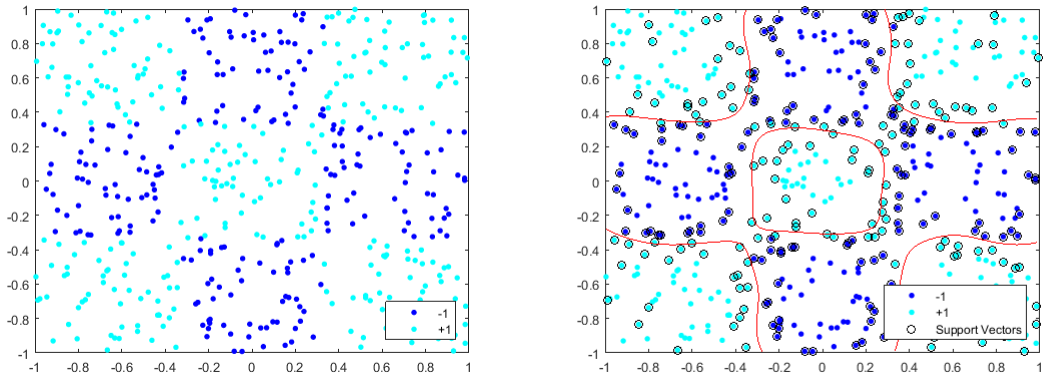

Figura 3.8: DATA 7

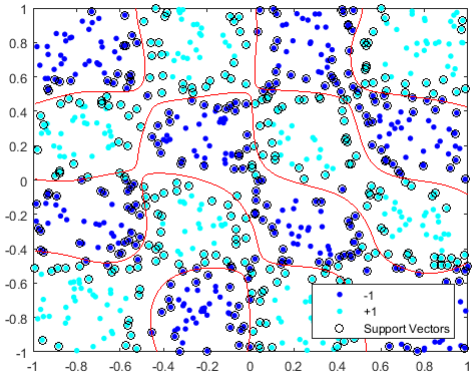

Figura 3.9: DATA 4x4

Extendiendo la data a 5x5 cuadrados, en cada cuadrado 50 puntos, es decir 1250 puntos respectivamente; volvemos a generar la curva de separación (ver figura 3.10) manteniendo la combinación convexa óptima para la data de 3x3 cuadrados. Observe que estas datas en tamaño son mucho mas grandes que las datas experimentadas en los ejemplos anteriores, y por esta razón es mas complicado computacionalmente resolver el programa expuesto en la sección 2.1, sus variables serían de mayor dimensión.

Este último experimento, nos muestra la bondad de resolver el programa propuesto en el capítulo 2, ya que podemos calcular la combinación convexa mas adecuada

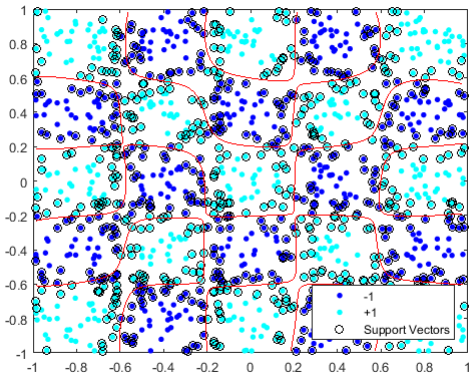

Figura 3.10: DATA tipo tablero de ajedrez

de funciones kernel conocidas, para una data mas sencilla, que inmediatamente procedemos a aplicar a la data más compleja (o más grande) como ya ilustramos.

Capítulo 4

Conclusiones

Durante la elaboración del presente trabajo de investigación, nos encontramos con la problemática de la elección de una buena función kernel al momento de clasificar una data, por lo que proponemos considerar la función kernel de la máquina de soporte vectorial como combinación convexa de funciones kernel conocidas previamente. Esta propuesta, con el objeto de conocer que función kernel es más adecuada para cada tipo de data, lo que nos ilustra el valor de la variable real correspondiente en la combinación convexa.
Al proceder computacionalmente a resolver nuestro programa propuesto (ver, capítulo 2), comprobamos su efectividad para seleccionar un kernel conveniente para las datas experimentadas. Aunque posteriormente nos enfrentamos al problema de cuales son los parámetros de las funciones kernel (influyentes en la combinación convexa resultante) mas adecuados para cada data, como se ilustra en el experimento *Data separable por una curva*. No obstante, aunque el programa computacionalmente no es rápido, como se muestra en el experimento *Data 3x3* la aplicación del kernel óptimo a datas similares resulta ser eficiente para clasificarlas.

Con los exprerimentos numéricos realizados, nos planteamos un nuevo problema, y es la escogencia de los parámetros de las funciones kernel involucradas en la combinación convexa resultante del programa expuesto en la capítulo 2. Lo más conveniente, sería estimar los parámetros de las funciones kernel mas adecuadas entre las

propuestas.

Capítulo 5

Apéndice

5.1. Códigos en lenguaje MATLAB

Para resolver los programas propuestos utilizamos lenguaje MATLAB R2015b, junto a su códico de optimizacón MPT3, el cual se puede instalar desde el link electrónico *http://people.ee.ethz.ch/ mpt/3/*.

5.2. Simular data

A continuación algunos algoritmos utilizados en el trabajo para simular las datas experimentadas.

Para simular data seprable linealmente y por une curva, usamos el siguiente algoritmo:

```
1 v=[-1  1  -1  1];
2 axis(v);
3 [x1,y1]=ginput(n1);  % n1 es el tamaño de la data 1
4 [x2,y2]=ginput(n2);  % n2 es el tamaño de la data 2
5 data1=[x1  y1];
6 data2=[x2  y2];
```

```
data=[data1;data2];
```

Para simular data por cuadrante:

```
rng(1);
% n el tamaño de puntos por cuadrante
data1=[rand(n,1),rand(n,1);-rand(n,1),-rand(n,1)];
data2=[-rand(n,1),rand(n,1);rand(n,1),-rand(n,1)];
```

Para simular data separable por una circunferencia:

```
rng(1);
r = sqrt(0.25*rand(n1,1)); % Radio
t = 2*pi*rand(n1,1);   % Angulo
data1 = [r.*cos(t), r.*sin(t)]; % Puntos
r2 = sqrt(0.25*rand(n2,1))+0.5; % Radio
t2 = 2*pi*rand(n2,1);        % Angulo
data2 = [r2.*cos(t2), r2.*sin(t2)]; % Puntos
data=[data1;data2];
```

Por último, para gráficar la data simulada:

```
tc=[ones(n1,1);-ones(n2,1)]
p=nan(2,1);
figure;
p=gscatter(data(:,1),data(:,2),tc,'bc');
legend(p,{'-1','+1'},'Location','Southeast');
axis ([-1 1 -1 1])
hold off
```

5.3. Esquema alternante

El siguiente algoritmo resuelve el problema expuesto en la sección 2.1, con H=I.

```
 [n1,~]=size(data1); %tamaño de la data 1
 [n2,~]=size(data2); %tamaño de la data 2
 m=n1+n2; %tamaño de toda la data
 %clasificando los datos 1 por 1 y los datos 2 por -1:
 d=[ones(n1,1);-ones(n2,1)];
 D=diag(d); %matriz diagonal d
```

```
7  e=ones(m,1);
8  v=0.02;
9  G1=data*data'; % Kernel Lineal
10 G2=G1.*G1; % Kernel Cuadratico
11 G3=G1.^3; % Kernel Polinomial de grado 3
12
13 for i=1:m
14    for j=1:m
15        G4(i,j)=exp(-((data(i,:)-data(j,:))*(data(i,:)-data(j,:))')); %
              Kernel de base radial
16    end
17 end
18 gamma = 1;
19 c=0;
20 G5 = tanh(gamma*G1 + c); % Sigmoid kernel function with slope gamma
       and intercept c
21 epsilon=0.001; % valor para decidir cuando parar
22 diff=2; % diff será la diferencia entre los valores objetivos
23 Nmax=0;
24 K0=(0.2)*G1+(0.2)*G2+(0.2)*G3+(0.2)*G4+(0.2)*G5; % valor inicial K
25 while (abs(diff) > epsilon) && (Nmax < 60)
26     % Primer Problema: Calcular u, y , g
27  % definición de variables de decisión:
28 u = sdpvar(m,1);
29 y = sdpvar(m,1);
30 g = sdpvar(1);
31
32  % Definición de restricciones del problem 1:
33   F1 = [D*((K0)*D*u-e*g)+y >= e];
34   F1 = F1 + [y >= 0];
35
36  % Función objetivo 1:
37 FO1 = v*e'*y+(0.5)*(u)'*u;
38  % min FO1 s.a. F1:
39  % P1 = sdpsettings('solver','quadprog');
40 solvesdp(F1,FO1);
41
```

```
42 valor_FO1=double(FO1);
43 u0=double(u);
44 g0=double(g);
45
46 % segundo problema: calcular K e y
47 % Definición de variables de decisión
48 y = sdpvar(m,1);
49 t = sdpvar(5,1);
50 % Definición de restricciones
51 F2 = [D*((t(1)*G1+t(2)*G2+t(3)*G3+t(4)*G4+t(5)*G5)*D*u0-e*g0)+y >= e];
52 F2 = F2 + [t(1)+t(2)+t(3)+t(4)+t(5) == 1];
53 F2 = F2 + [t >= 0];
54 F2 = F2 + [y >= 0];
55 % Función Objetivo
56 FO2 = v*e'*y+(0.5)*(u0)'*u0;
57 % min FO2 s.a. F2
58 % P2 = sdpsettings('solver','linprog');
59 solvesdp(F2,FO2);
60 valor_FO2=double(FO2);
61 K=t(1)*G1+t(2)*G2+t(3)*G3+t(4)*G4+t(5)*G5;
62 K0=double(K);
63 diff=abs(valor_FO1 - valor_FO2);
64 Nmax=Nmax+1;
65 end
```

5.4. Graficar la curva de separación

Despues de correr computacionalmente el algoritmo 1 o el 2, procedemos a cambiar los parametros t_i por los obtenidos en la función Kernel.m:

```
1 function Ker=Kernel(U,V)
2 Ker=KernelComb(U,V, t1, t2, t3, t4, t5);
3 % colocar los valores de t obtenidos
4 end
```

donde KernelComb.m es:

```
function K= KernelComb(U,V,t1,t2,t3,t4,t5)
[k,~]=size(U); % tamaño de la data 1
[kk,~]=size(V);% tamaño de la data 2

G1=U*V'; % Kernel Lineal
G2=G1.*G1; % Kernel Cuadratico
G3=G1.^3; % Kernel Polinomial de grado 3
% Kernel de base radial, o Gaussiano:
for i=1:k
  for j=1:kk
      G4(i,j)=exp(-((U(i,:)-V(j,:))*(U(i,:)-V(j,:))'));
  end
end
gamma = 1; % parametros del sigmoide que se pueden cambiar
c=0;
G5 = tanh(gamma*G1 + c); % Kernel sigmoide, pentiente gamma e inters
    c
K = t1*G1+t2*G2+t3*G3+t4*G4+t5*G5;
end
```

Luego, procedemos a correr el siguiente algoritmo, que genera la curva de separación:

```
% Debe fijarse en Kernel.m, que los parametros esten como el optimo
% obtenido.

[n1,~]=size(data1); % tamaño de la data 1
[n2,~]=size(data2);% tamaño de la data 2
m=n1+n2;

tc = ones(n1,1);
tc((n1+1):m) = -1;
cl = fitcsvm(data,tc,'KernelFunction','Kernel');

l = 0.009;
[x1Grid,x2Grid] = meshgrid(min(data(:,1)):l:max(data(:,1)),...
min(data(:,2)):l:max(data(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(cl,xGrid);
```

```
17
18 h = nan(3,1); % Preasignación
19 figure;
20 h(1:2) = gscatter(data(:,1),data(:,2),tc,'bc'); % data
21 hold on
22 h(3) = plot(data(cl.IsSupportVector,1),...
23    data(cl.IsSupportVector,2),'ko'); % plot vectores soporte
24 % dibujar la curva de acuerdo al predictor:
25 contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
26 legend(h,{'-1','+1','Support Vectors'},'Location','Southeast');
27 axis ([-1 1 -1 1])
28 hold off
```

Bibliografía

[1] A. F. Baudat. *Kernel-based Methods and Function Approximation.* MEI, Mars Electronics International. Wilson Drive, PA, USA. 1999.

[2] Cortez, J. *Máquinas vectoriales de soporte vía programación semidefinida*, Trabajo de Grado, Venezuela, Barquisimeto, Julio 2014.

[3] Michael C. Ferris, Olvi L. Mangasarian, Stephen J. Wright. *LINEAR PROGRAMMING WITH MATLAB*, first edition, University of Wisconsin, Madison, 2007.

[4] G. Lanckriet; N. Cristianini; P. Bartlett; L. El Ghaoui; M. Jordan. *Learning the Kernel Matrix with Semidefinite Programming*, Journal of Machine Learning Research, 5 (2004) 27–72

[5] O. Mangasarian. *Generalized Support Vector Machines*. University of Wisconsin. Madison. WI. 53706. 1998.

[6] O. Mangasarian. *Data Mining via Support Vector Machines*. University of Wisconsin. Madison. WI. 53706. 2002.

[7] Igor Griva, Stephen G. Nash, Ariela Sofer
Linear and Nonlinear Optimization, second edition, George Mason University Fairfax, Virginia, 2009.

[8] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[9] C. J. C. Burges. *A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery*, 2:121.167, 1998.

[10] G. Lanckriet, N. Cristiani, P. Bartlett, L. El Ghaoui, and M. Jordan. *Learning the kernel matrix with semidefinite programming.* J. of Machine Learning Research 5, pages 27?72, 2004.