UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

Decanato de Ciencias y Tecnología
Coordinación de Postgrado

MÁQUINAS DE VECTORES DE SOPORTE BASADAS EN FUNCIONES NÚCLEOS

Trabajo Especial de Grado presentado por

Yesenia Maigualida Rivas Garcia.

Como requisito final para obtener el título de

Maestría en Ciencias – Mención optimización,

Área de Conocimiento: Matemática aplicada.

Tutor: Dr. Javier Hernández Benítez.

Barquisimeto - Venezuela

Octubre 2016

MÁQUINAS DE VECTORES DE SOPORTE BASADAS EN FUNCIONES NÚCLEOS

Yesenia M. Rivas G.

RESUMEN

En este trabajo estudiamos algunas propiedades y características de las funciones núcleos usadas en problemas de clasificación y regresión, que usan como herramienta las máquinas de vectores de soportes. Realizamos un estudio comparativo de distintos núcleos empleados para la clasificación de distintos tipos de data.

A DIOS, a mis Padres y a mis Hermanos.

Índice general

Índice de figuras

Agradecimientos

Primeramente **A DIOS**, por haberme dado la sabiduría necesaria y el entendimiento para poder lograr culminar mis estudios de maestría. Porque, todo lo que tengo y lo que soy te lo debo a ti Padre Celestial, la Gloria y Honran sean para ti. Este logro es tuyo.

A MIS PADRES, gracias infinitas por su apoyo incondicional, esfuerzo, dedicación, paciencia y orientación cuando mas les he necesitado, por toda la confianza que depositaron en mi, por darme ánimos en los momentos que decaía. "Este logro les pertenece a ustedes". También, A ti abuelita que aunque no estés presente físicamente, lo has estado siempre en cada logro de mi vida.

A mis Hermanos, a ustedes por estar siempre apoyándome, protegiéndome, brindándome la fuerza necesaria para continuar. A mis Sobrinos, por llenarme de alegrías. A mi Amigo y compañero de vida, a ti Rey gracias por tu apoyo, por cada palabra de animo y paciencia infinita para conmigo. Gracias Amor.

A mis amigos: Ifigenia, Carlos, Katty, Laura, Emily , Francis, Marcos, Hector, Henrry, por haber compartido alegrías, y momentos difíciles en este camino gracias por el apoyo y ayuda para alcanzar culminar esta meta.

A todos mis profesores porque de una u otra forma, son parte de lo que soy ahora. A usted profesora Jurancy, gracias por su ayuda incondicional en todo momento, gracias por cada palabra de aliento en momentos de desanimo y por impulsarme a seguir adelante, a no rendirme. Gracias por cada una de sus enseñanzas y por su tiempo. En especial a mi Tutor Javier Hernández. El haberme permitido trabajar una vez mas con usted gracias por cada día enseñarme con la mayor paciencia y dedicación del mundo, gracias porque en mis momentos de inseguridad y angustia allí estaba un “No temáis...” me daba confianza en que junto a usted lo iba lograr. Se les quiere y aprecia mucho, sin ustedes no hubiese sido posible alcanzar esta meta. Son fuente de gran inspiración y admiración. Gracias totales profes Dios me los Bendiga siempre.

Introducción

En este trabajo se desarrolla un estudio sobre las máquinas de vectores de soporte basadas en las funciones núcleos. Las máquinas de vectores de soporte o máquinas de soporte vectorial (SVM por sus siglas en inglés Support Vector Machine) son un conjunto de algoritmos de aprendizaje supervisado desarrollado originalmente por Vapnik (ver [14], [15]) y han sido introducidas como una poderosa herramienta para resolver problemas de clasificación y /ó regresión.

Una Máquina de Vectores de Soporte constituye una técnica de clasificación de datos, que es conformada por dos o mas clases de puntos, dicho conjunto de datos es previamente catalogado con una etiqueta $\{\pm 1\}$. La máquina es capaz de predecir si un punto cuya categoría es desconocida pertenece a una clase o a otra; para ello se cuenta con algoritmos y funciones que permiten realizar la clasificación. En medio de todos los posibles hiperplanos que separan ambas clases etiquetadas como $\{\pm 1\}$, existe solo un hiperplano de separación optima en $\mathbb{R}^n$ que nos separa

la muestra en dos clases, ya que los vectores x_i asociados a los datos de una clase estarán a un lado del hiperplano, y los asociados a otras clases, en otro lado del hiperplano [6], [7].

Pero no siempre es posible encontrar un hiperplano que separe, mas aun cuando los puntos no son separables linealmente es allí cuando entran la funciones núcleos y juegan un papel fundamental en el aprendizaje de máquinas, la idea es construir una función clasificadora que minimice el error en la separación de los objetos dados, el error en clasificación, y maximice el margen de separación (mejorando la generalización del clasificador) [12]. Es importante destacar que la técnica para encontrar el hiperplano que separa, en los casos que no son linealmente separables se desarrolla mediante las funciones núcleos; estas nace a partir de los años 1907 con núcleos reproductivos que aparecen por primera vez por Stanislaw Zaremba en su trabajo [16]; siendo este el primero que introdujo en un caso particular, el núcleo correspondiente a una clase de funciones, y enuncio su propiedad de reproducción.

Mientras que dos años mas tarde en 1908 Mercer [8] descubre que dichos núcleos poseían una propiedad particular, de nombre propiedad reproductiva. A partir de ahí Mercer comenzó a estudiar sobre teoría de las funciones que satisfacen la propiedad de reproducción basadas en la teoría de ecuaciones integrales de Hilbert, nombrando estas funciones como núcleos definidos positivos. Mostró así mismo que estos núcleos definidos positivos tienen buenas propiedades de entre todos los núcleos continuos de ecuaciones integrales. Fue Eliakim. H. Moore en [2], quien amplió la teoría a funciones complejas y realizo un análisis general, bajo el nombre de matriz hermitiana positiva. Posteriormente en 1950, Nachama.

Aronszajn publicó una teoría de núcleos reproductivos donde demostró que cada núcleo reproductivo definido positivo y simétrico, determina un único espacio de Hilbert con núcleo reproductivo; teorema que ahora se conoce como teorema de Moore-Aronszaje [2].

La estructura de este trabajo esta diseñada de la forma siguiente: en el capítulo 1 se muestran formulaciones para SVM como programas cuadráticos convexos. Además se dan las dos tareas principales de una SVM como son; la separación y clasificación de los puntos o datos por una superficie no-necesariamente lineal, maximizando la distancia. Maximizar la distancia entre los planos separadores en el espacio de mayor dimensión consiste en la eliminación de vectores soportes, lo cual, se consigue al reducir al mínimo cualquier norma deseada de los multiplicadores de vectores soportes. Donde la norma puede ser inducida por el núcleo de separación si resulta ser definida positiva o por una euclídea, las mismas conducen a un programa convexo cuadrático, todos ellos con una separación del núcleo arbitraria.

Debido a que las SVM no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real describiremos un sistema para la identificación de SVM no lineales, para lograr esto comenzaremos por introducir a las funciones núcleos en el capítulo 2 donde nos ayudara a comprender mejor la representación por medio de las funciones núcleos, en este capítulo se presenta de manera mas formal y detallada definición propiedades y teoremas. Generalmente las funciones núcleo son usadas cuando el problema no puede ser separado a través de un hiperplano en el espacio de entrada, y esto ocurre muchas veces debido a que la dimensión del problema en el espacio de entrada es baja, para ello es necesario proyectar

los datos originales a un espacio de dimensión superior ó incluso infinita a través de una función ϕ, donde los datos si pueden ser separados a través de un hiperplano. Además expresar como la elección del núcleo puede afectar la precision de la clasificación en las SVM.

En el capítulo 3 se exponen algunas pruebas numéricas usando el modelo propuesto por O. L. Mangasarian en [6] aplicado a una base de datos generadas en forma aleatoria. Es importante resaltar que los conjuntos de datos se le hicieron pruebas con diferentes funciones núcleos entre los que se aplicaron el núcleo general, núcleo Gaussiano, núcleo polinomial homogéneo, núcleo polinomial no homogéneo. Finalmente en el apéndice se encuentran los algoritmos y programas utilizados en MATLAB para el desarrollo de las SVM basadas en las funciones núcleos. Especificando que función cumple cada programa.

1

Máquinas de vectores de soporte (SVM)

Las máquinas de vectores de soporte o máquinas de soporte vectorial (SVM) por sus siglas en inglés (Support Vector Machine) son un conjunto de algoritmos de aprendizaje supervisado desarrollado originalmente por Vapnik [14],[15], y han sido introducidas como una poderosa herramienta para resolver problemas de clasificación y /ó regresión.

Una SVM transforma los puntos de entrada a un espacio de características de mayor dimensión y encuentra un hiperplano que separa los datos y maximiza el margen entre las clases de puntos. Una SVM es un modelo que representa los puntos de una muestra en un espacio, separando las clases de puntos entre sí con la mayor distancia posible, es decir con un margen máximo. Maximizar el margen consiste en maximizar la distancia entre el hiperplano separador y el valor de entrada más cercano, es decir el vector soporte (punto donde se apoya el margen máximo). Una SVM, constituye un modelo capaz de predecir si un nuevo punto (cuya categoría es desconocida) pertenece a una categoría o a la otra.

Es importante resaltar que la formulación de las máquinas de vectores soportes varía dependiendo de la naturaleza de los datos. Supongamos que tenemos un conjunto de datos $(x_1, y_1), (x_2, y_2) \ldots (x_m, y_m) \in X \times \{\pm 1\}$, donde X es un conjunto no vacío llamado espacio de entrada. Los datos x_i son extraídos del conjunto X y los y_i son las denominadas etiquetas; acá tendremos una $x \in X$ y queremos predecir un $y \in \{\pm 1\}$ y esto se logra encontrando un hiperplano que separe ambas clases de puntos; es importante destacar que pueden existir infinitos hiperplanos que separen los datos.

En medio de todos los posibles hiperplanos que separan las clases, existe solo un hiperplano de separación óptima en $X \subset \mathbb{R}^n$. Los vectores x_i asociados a los datos de una clase estarán a un lado del hiperplano, y los asociados a otra clase, en otro lado del hiperplano, ó lo que es lo mismo, existe un par $(\omega, \gamma) \in R^{n+1}$ tal que $x_i^T \omega + \gamma > 0$ si el punto x_i esta en la clase positiva, y $x_i^T \omega + \gamma < 0$ si el punto x_i esta en la clase negativa.(ver la figura 1.1 y 1.2)

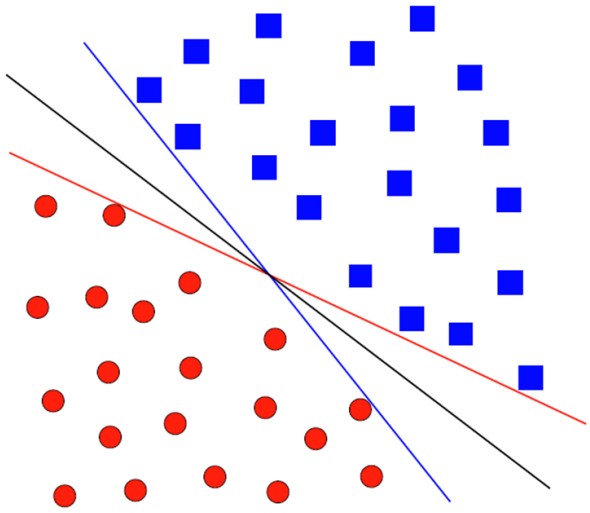


Figura 1.1: Hiperplanos separadores

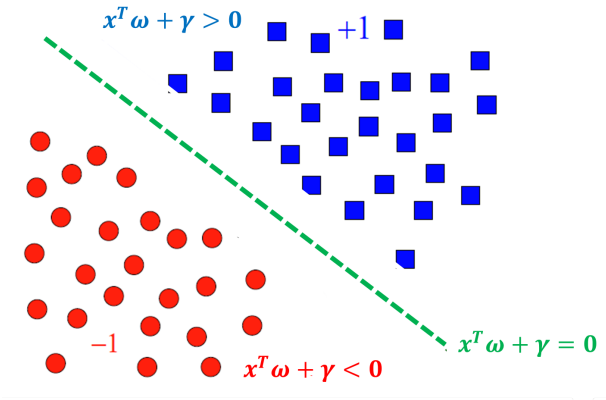


Figura 1.2: Hiperplano separador óptimo

Definición 1.0.1 (Hiperplano Separador) *[10] Dados dos conjuntos C_1 y C_2 , subconjuntos de $\mathbb{R}^n$ no vacíos, un hiperplano*

$$H = \{x \in \mathbb{R}^n / c^T x = \alpha; con\ \alpha \in \mathbb{R}, c \in \mathbb{R}^n\}$$

Separa los dos conjuntos C_1 y C_2 cuando se verifica que:

$$x_1 \in C_1 \Rightarrow c^T x_1 \geq \alpha$$

$$x_2 \in C_2 \Rightarrow c^T x_2 \leq \alpha$$

Teorema 1.0.2 *[10] Sean C_1 y C_2 conjuntos no vacíos en $\mathbb{R}^n$. Entonces existe un hiperplano H que separa C_1 y C_2 propiamente si y sólo si existe un vector b tal*

que

$$\text{ínf}\{\langle x, b\rangle / x \in C_1\} \geq \sup\{\langle x, b\rangle / x \in C_2\}$$
$$\sup\{\langle x, b\rangle / x \in C_1\} > \text{ínf}\{\langle x, b\rangle / x \in C_2\}$$

El hiperplano H separa a C_1 y C_2 fuertemente si y sólo si existe un vector b tal que

$$\text{ínf}\{\langle x, b\rangle / x \in C_1\} > \sup\{\langle x, b\rangle / x \in C_2\}$$

Como lo que se quiere es la separación óptima de la muestra, es decir; la que sea de mayor distancia posible, consideremos la distancia d_+ (la distancia euclídea mas corta entre el hiperplano y la clase positiva mas cercana) y d_- (la distancia euclídea mas corta entre el hiperplano y la clase negativa mas cercana). Definiremos el margen de separación del hiperplano como $d_+ + d_-$.

Si un hiperplano esta definido por un par (ω, γ), entonces cualquier par de la forma $(\lambda\omega, \lambda\gamma)$ con $\lambda > 0$, define el mismo hiperplano, además, si $x_i^T \omega + \gamma > 0$ para cierto par (ω, γ) entonces va a existir (ω_i, γ_i) que define el mismo hiperplano tal que $x_i^T \omega_i + \gamma_i \geq 1$. De manera similar se procede con la clase negativa.

Por tanto las condiciones pueden ser agregadas en el problema de optimización de la siguiente manera

$$\begin{aligned} x_i^T \omega + \gamma \geq 1 \quad & para\ y_i = +1 \\ x_i^T \omega + \gamma \leq -1 \quad & para\ y_i = -1 \end{aligned} \tag{1.1}$$

Esto es

$$y_i(x_i^T\omega + \gamma) - 1 \geq 0 \quad \forall i \tag{1.2}$$

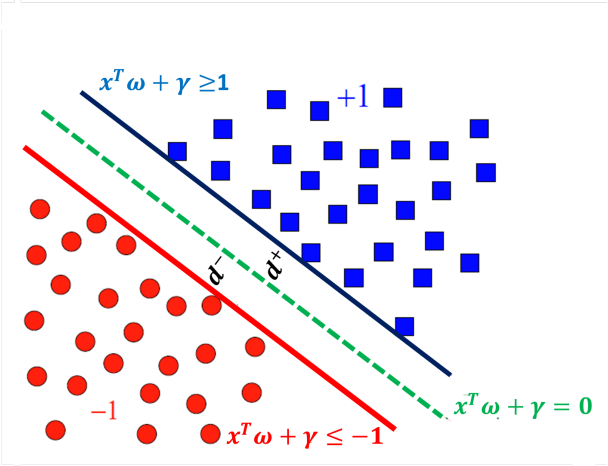


Figura 1.3: SVM estandar

Consideremos los puntos para los que se cumple la igualdad $H_1 : x_i^T\omega + \gamma = 1$, estos puntos están sobre el hiperplano H_1 con vector normal ω y la distancia del hiperplano al origen esta dada por $\frac{|\gamma|}{\|\omega\|}$ donde $\|\omega\|$ es la norma euclídea de ω, análogamente se cumple para el hiperplano $H_2 : x_i^T\omega + \gamma = -1$, por tanto; $d_+ = d_- = \frac{1}{\|\omega\|}$.

Nótese que H_1 y H_2 son paralelos (poseen el mismo vector normal) y que no hay puntos de entrenamientos entre ellos (ver la figura 1.4).

Podemos en definitiva encontrar el par que da el margen máximo, y lo pode-

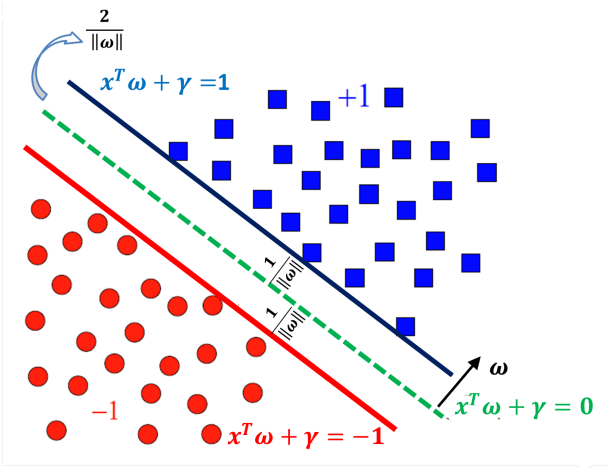


Figura 1.4: Margen máximo

mos hacer mediante la resolución de dicho problema de la forma

$$\begin{aligned} &\max_{\omega,\gamma} \quad \frac{2}{\|\omega\|} \\ &s.a. \quad y_i(x_i^T\omega + \gamma) - 1 \geq 0. \qquad i = 1, \ldots, m, \end{aligned} \tag{1.3}$$

Este problema es equivalente a minimizar la norma euclídea de ω, de esta manera el problema de optimización a resolver nos queda:

$$\begin{aligned} &\min_{\omega,\gamma} \quad \frac{\|\omega\|^2}{2} \\ &s.a. \quad y_i(x_i^T\omega + \gamma) - 1 \geq 0. \qquad i = 1, \ldots, m \end{aligned} \tag{1.4}$$

Resolviendo este problema se obtiene un vector óptimo ω^* que permite calcular el margen máximo de un hiperplano de separación.

Este problema puede ser resuelto usando el Dual de Wolfe, introduciendo los

multiplicadores de Lagrange, es decir; que las restricciones de (1.4) se sustituirán sobre multiplicadores de Lagrange que serán mas fácil de manejar. [6], [7].

El problema anterior puede tener o no solución, en el caso en que tenga solución este problema sería cuadrático convexo y tendría una única solución óptima global. Podemos obtener la solución encontrando las condiciones de optimalidad de primer orden de **KKT** (Karush -Kuhn-Tucker), [10] para esto hacemos uso del Lagrangiano, introduzcamos m multiplicadores de Lagrange que denotaremos por $\alpha_1, \ldots, \alpha_m$ uno por cada restricción de (1.4). De esta manera lo anterior nos lleva a la siguiente función Lagrangiana;

$$L_p(\omega, \gamma, \alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^{m} \alpha_i (y_i (x_i^T w + \gamma) - 1) \tag{1.5}$$

luego,

$$L_p(\omega, \gamma, \alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^{m} \alpha_i y_i (x_i^T w + \gamma) + \sum_{i=1}^{m} \alpha_i \tag{1.6}$$

Así las condiciones de KKT para el problema (1.4) viene dadas por:

$$\begin{aligned} \tfrac{\partial L_p}{\partial w_j} &= 0, \\ w_j - \sum_{i=1}^{m} \alpha_i y_i x_i^T &= 0, \end{aligned} \tag{1.7}$$

$$\begin{aligned} \tfrac{\partial L_p}{\partial \gamma} &= 0, \\ -\sum_{i=1}^{m} \alpha_i y_i &= 0, \end{aligned} \tag{1.8}$$

$$y_i(x_i^T w + \gamma) - 1 \geq 0, \tag{1.9}$$

$$\alpha_i \geq 0, \tag{1.10}$$

$$\alpha_i(y_i(x_i^T w + \gamma) - 1) = 0. \tag{1.11}$$

Como se debe minimizar L_p con respecto a ω y γ podemos resolver en su forma equivalente con su problema Dual. Aplicando dualidad lagrangiana tenemos que

$$L_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j). \tag{1.12}$$

Luego, su problema Dual viene dado por:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\ s.a. \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & \alpha_i \geq 0. \qquad i = 1, \ldots, m \end{aligned} \tag{1.13}$$

En definitiva estamos en presencia de un problema de programación cuadrática convexa, con restricciones lineales. La solución para este problema consiste en obtener los valores óptimos para los multiplicadores de Lagrange α_i, que no es mas que encontrar (ω^*, γ^*), que define el hiperplano de separación óptima.

Nótese que hay un multiplicador de Lagrange para cada muestra de entrenamientos. Además, tras obtener una solución aquellos puntos donde $\alpha_i > 0$, con $i = 1, \ldots, n$, se denominan los vectores soporte y yacen sobre los hiperplanos

H_1, H_2. El resto de los datos tienen $\alpha_i = 0$ y cumplen con

$$y_i(x_i^T w + \gamma) - 1 \geq 0 \quad \forall i$$

Así ω se escribirá como combinación lineal de los vectores soportes, los mismos son los elementos críticos del conjunto de entrenamiento, son lo mas cercano a la frontera de decisión, y si el resto de los puntos no se consideran en un nuevo entrenamiento, el algoritmo encontraría el mismo hiperplano de separación.

En el caso en que problema no tenga solución es decir, los datos no pueden ser separados a través de un hiperplano, Esto es, no se llega a una solución factible pues no existirá un hiperplano que cumpla con la condición $y_i(x_i^T w+\gamma)-1 \geq 0$ es por ello, que nos interesa poder relajar las restricciones, pero únicamente cuando sea necesario, para ello añadimos un nuevo coste a la función objetivo.(ver la figura 1.5)

Se añadirá una penalización como consecuencia de los patrones mal clasificados. Una posible forma de hacerlo es introducir variables de holgura ξ_i con $i = 1, \ldots, m$, en las restricciones para convertirlas en;

$$\begin{aligned} x_i^T \omega + \gamma &\geq +1 - \xi_i \quad \textit{para } y_i = +1, \\ x_i^T \omega + \gamma &\leq -1 + \xi_i \quad \textit{para } y_i = -1, \\ \xi_i &\geq 0. \end{aligned} \tag{1.14}$$

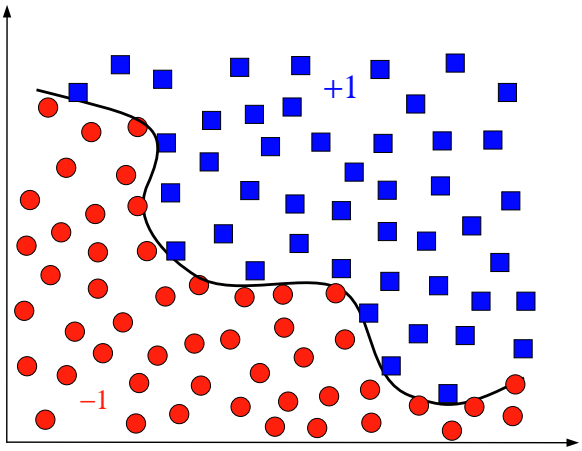


Figura 1.5: Datos no-separables por un hiperplano

esto es;

$$\begin{aligned} y_i(x_i^T\omega + \gamma) &\geq 1 - \xi_i, \\ \xi_i &\geq 0 \qquad\qquad \forall i. \end{aligned} \tag{1.15}$$

Ahora, podemos preguntarnos como es posible aceptar puntos mal clasificados por el hiperplano; para ello penalizamos el error de clasificación a través una combinación lineal con la función objetivo, es decir, sustituir esta expresión

$$\min_{\omega,\gamma} \quad \frac{\|\omega\|^2}{2} \tag{1.16}$$

por la siguiente expresión

$$\min_{\omega,\gamma,\xi} \quad \frac{\|\omega\|^2}{2} + C\sum_{i=1}^{m}\xi_i \tag{1.17}$$

donde C es un parámetro ajustable; note que, si C tiene un valor muy alto equivale

a una mayor penalización de los errores. Para patrones que son correctamente clasificados $\xi_i = 0$. Ahora bien, la noción de suavizar el margen permite tratar datos mas realistas; es por ello, que el problema anterior se reduce a un nuevo problema de programación cuadrática convexa con la propiedad de que tanto los ξ_i como los multiplicadores de lagrange asociados a esta variable no aparecen en el problema Dual.

$$\begin{aligned} \max_{\alpha} \quad & L_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_1 \alpha_j y_i y_j (x_i x_j) \\ s.a \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C. \qquad\qquad i = 1, \ldots, m \end{aligned} \tag{1.18}$$

la solución viene dada por:

$$\omega = \sum_{i=1}^{N_s} \alpha_i y_i x_i. \tag{1.19}$$

donde N_s es el número de vectores soportes. Luego, la solución óptima puede escribirse como una combinación lineal de vectores soporte. Por lo tanto, la única diferencia con el caso separable del hiperplano óptimo es que los valores de α_i están ahora acotados superiormente por C.

Necesitaremos las condiciones Karush-Kuhn-Tucker para el problema primal. El Lagrangiano primal es,

$$L_p = \frac{\|\omega\|^2}{2} + C \sum_i \xi_i - \sum_i \alpha_i [y_i (x_i^T \omega + \gamma) - 1 + \xi_i] - \sum_i \mu_i \xi_i \tag{1.20}$$

Pues los μ_i representan los multiplicadores de Lagrange asociados a la condición $\xi_i \geq 0$.

Las condiciones KKT para el problema primal son por lo tanto,

$$\frac{\partial L_p}{\partial \omega_\nu} = \omega_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0,$$

$$\frac{\partial L_p}{\partial \gamma} = -\sum_i \alpha_i y_i = 0,$$

$$\frac{\partial L_p}{\partial \gamma} = -\sum_i \alpha_i y_i = 0,$$

$$y_i(x_i^T \omega + \gamma) - 1 + \xi_i \geq 0,$$

$$\xi_i \geq 0, \alpha_i \geq 0, \mu_i \geq 0,$$

$$\alpha_i[y_i(x_i^T \omega + \gamma) - 1 + \xi_i] = 0,$$

$$\mu_i \xi_i = 0.$$

En muchas situaciones, los datos pueden ser separados linealmente, a través de una transformación no lineal del espacio de entrada en un espacio de mayor dimensión llamado espacio de características. La transformación de los datos de un espacio inicial a otro de mayor dimensión se logra mediante, el uso de una función núcleo. En este sentido una función núcleo es un producto interno en el espacio de las características que tiene su equivalente en el espacio de entrada y esta dada por;

$$k(x, x^T) = \langle \Phi(x), \Phi(x^T) \rangle. \tag{1.21}$$

donde k es una función simétrica definida positiva que cumple con las condiciones

de Mercer [8], [12]. A continuación mostramos el problema de optimización a resolver para las SVM con un núcleo general el cual esta definido por un problema de programación cuadrática con restricciones dado por:

$$\begin{aligned} \max_{\alpha} \quad & L_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ s.a \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C. \qquad\qquad i = 1, \ldots, m \end{aligned} \tag{1.22}$$

De manera gráfica se puede observar en la figura, (ver figura 1.6) como la función núcleo permite realizar la separación y traslado de los datos al espacio de características.

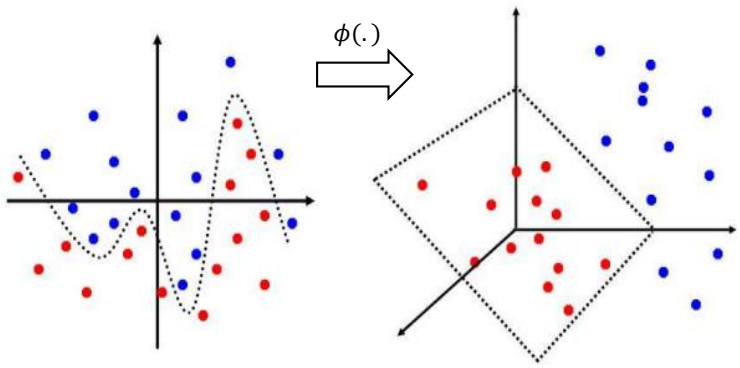


Figura 1.6: Nucleo

2

Núcleos Reproductivos

Las funciones núcleo, son funciones que se emplean en las máquinas vectoriales de soporte (SVM). Estas funciones son las que permiten convertir la información de un espacio de entrada a un nuevo espacio de mayor dimensión llamado espacio de características. Generalmente las funciones núcleo se usan cuando el problema no puede ser separado a través de un hiperplano, y esto ocurre muchas veces debido a que la dimensión del problema en el espacio de entrada es baja, para ello es necesario proyectar los datos originales a un espacio de dimensión superior o incluso infinita a través de una función ϕ, donde los datos si pueden ser separados a través de un hiperplano. No cualquier función puede ser una función núcleo, ella debe satisfacer la condición del teorema de Mercer.

A continuación describiremos de forma detalla una función núcleo y sus propiedades y la utilidad del teorema de Mercer [8].

Definición 2.0.3 (Mapa de características [12]) *La función* Φ *que traslada los*

elementos del espacio de entrada $\mathbb{X}$ *al espacio de características* $\mathbb{H}$ *(Hilbert)*

$$\begin{aligned} \Phi : \mathbb{X} &\longrightarrow \mathbb{H} \\ x &\longrightarrow \Phi(x) \end{aligned} \tag{2.1}$$

se denomina mapa de características

Definición 2.0.4 (Función núcleo [12]) *Se define el producto escalar del espacio* $\mathbb{X} \times \mathbb{X}$ *como el núcleo k en función del mapa* Φ*;*

$$\begin{aligned} k : \mathbb{X} \times \mathbb{X} &\longrightarrow \mathbb{K} \\ (x, x') &\longrightarrow k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathbb{H}} \end{aligned} \tag{2.2}$$

donde $\mathbb{K} = \mathbb{C}$ *o* $\mathbb{K} = \mathbb{R}$

Algunas propiedades que debe cumplir la función núcleo para definir un espacio de características son las siguientes:

1. Simetría $k(x, x') = k(x', x)$
2. Desigualdad de Cauchy-Schwarz $k(x, x')^2 \leq k(x, x) \cdot k(x', x')$

Ahora vamos a imponer algunas condiciones de la función k y antes de introducir la definición del núcleo reproductor, daremos dos teoremas importantes en la teoría de análisis funcional como lo es el el Primer Teorema de Representación de Riesz , este Teorema será de particular interés, debido a que es la base de la

Propiedad Reproductiva; el cual nos permite tener una representación explícita y única de funcionales lineales acotados y el Segundo Teorema de Representación de Riesz , nos permite representar de forma única una función sesquilineal acotada, como un producto interno sobre un espacio de Hilbert.

Teorema 2.0.5 (Primer Teorema de Representación de Riesz [5]) . *Todo funcional lineal acotado f en un espacio de Hilbert H puede ser representado como*

$$f(x) = \langle x, z\rangle \tag{2.3}$$

de forma única donde z depende de f, mas aun, la $\|f\| = \|z\|$

Definición 2.0.6 *Sean X y Y dos espacios vectoriales sobre el mismo cuerpo K. Una forma sesquilineal es una aplicación $h : X \times Y \longrightarrow K$ tal que para todo $x, x_1, x_2 \in X$, todo $y, y_1, y_2 \in Y$ y todo $\alpha, \beta \in K$*

1. $h(\alpha x_1 + \beta x_2, y) = \alpha h(x_1, y) + \beta h(x_2, y)$
2. $h(x, \alpha y_1 + \beta y_2) = \overline{\alpha} h(x, y_1) + \overline{\beta} h(x, y_2)$

Teorema 2.0.7 (Segundo Teorema de Representación de Riesz [5]) *$\mathbb{H}_1$ y $\mathbb{H}_2$ espacios de Hilbert y $h : \mathbb{H}_1 \times \mathbb{H}_2 \longrightarrow K$ una forma sesquilineal acotada. Entonces, existe un único operador lineal acotado $S : \mathbb{H}_1 \longrightarrow \mathbb{H}_2$ tal que*

$$h(x, y) = \langle S_x, y\rangle_{\mathbb{H}_2}$$

para todo $x \in \mathbb{H}_1$ y todo elemento $y \in \mathbb{H}_2$ mas aun $\|h\| = \|S\|$

Introduciremos ahora Espacios de Hilbert con Núcleo Reproductivo. Este concepto es fundamental para las Máquinas vectoriales de soportes, debido a que estos espacios permitirán la clasificación de los datos, cuando éstos no son linealmente separables.

Definición 2.0.8 (Núcleo Reproductor [12]) *Sea* $\mathbb{X}$ *un conjunto no vacío llamado también conjunto de indices y sea* $\mathbb{H}$ *un espacio de Hilbert de funciones* $f : \mathbb{X} \longrightarrow \mathbb{R}$ *una función* $k : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{K}$ *es un núcleo reproductor de un espacio de Hilbert* $\mathbb{H}$ *si y solo si se cumple:*

1. $k(x, \cdot) \in \mathbb{H}$ *para todo* $x \in \mathbb{X}$
2. *Propiedad reproductora:* $\langle f, k(x, \cdot)\rangle_{\mathbb{H}} = f(x)$ *para todo* $f \in \mathbb{H}$ *y* $x \in \mathbb{X}$

El nombre de propiedad reproductora se debe al hecho de que el valor de la función f en el punto x es reproducido por el producto interno de f y k de esta definición esta claro que:

Nota 2.0.9 *Si todos los funcionales de evaluación son acotados, la propiedad reproductiva se sigue directamente del Primer Teorema de Representación de Riesz.*

La siguiente proposición permite definir un tipo especial de espacio de Hilbert basado en núcleo reproductor.

Proposición 2.0.10 *[12] Si k es un núcleo reproductor cumple que:*

$$\langle k(x,\cdot), k(x',\cdot)\rangle = k(x,x') \quad \textit{para todo} \quad (x,x') \in \mathbb{X}\times\mathbb{X}$$

conociendo el mapa de características $\Phi(x) = k(x,\cdot)$*, vemos que el núcleo reproductor es un núcleo.*

Definición 2.0.11 (Espacios de Hilbert con núcleo reproductor [12]) *Un espacio de Hilbert que posee un núcleo reproductor se denomina Espacio de Hilbert con Núcleo Reproductor (RKHS). Su mapa de características se denomina mapa canónico y se define en función del núcleo k,*

$$\Phi(x) = k(x,\cdot)$$

El siguiente teorema es muy importante en la teoría de (RKHS)

Teorema 2.0.12 (Moore Aronszajn) *[12] Un Espacio de Hilbert con Núcleo Reproductor determina un único núcleo reproductivo.*

El uso de la función núcleo hace posible el mapeo de la información de entrada (x, x') al espacio de características $(\Phi(x), \Phi(x'))$ de forma implícita y entrena a la máquina en dicho espacio. La única información necesaria para el entrenamiento es la matriz de Gram, Dicha matriz también es conocida como la matriz núcleo la cual se denota con la letra K.

Definición 2.0.13 (Matriz de Gram [12]) *:Dada una función* $k : \mathbb{X}\times\mathbb{X} \longrightarrow \mathbb{K}$

y patrones $x_1, x_2, \ldots x_m \in \mathbb{X}$, *la matriz* K *de orden* $m \times m$ *con elementos:*

$$K_{ij} := k(x_i, x_j)$$

Es llamada matriz de Gram o matriz núcleo de k *con respecto a* $x_1, x_2, \ldots x_m$.

Definición 2.0.14 (Matriz definida positiva [12]) *:Una matriz* K *simétrica de orden* $m \times m$ *satisfaciendo*

$$\sum_{ij} c_i \overline{c_j} K_{ij} > 0$$

para todo $c_i \in \mathbb{K}$ *se llama definida positiva (PD)*

Nota 2.0.15 *Una matriz simétrica es definida positiva si y solo si todos sus valores propios son no negativos.*

Definición 2.0.16 (Núcleo definido positivo[12]) *Diremos que un núcleo* k *es definido positivo si la matriz de Gram es definida positiva para todo* $x_1, x_2, \ldots x_m \in \mathbb{X}$ *con* $m \in \mathbb{N}$.

Ahora que tenemos los conocimientos básicos la pregunta natural es cuando una función k en $\mathbb{X} \times \mathbb{X}$ es un núcleo reproductor, una caracterización de núcleos reproductores vine dada por el siguiente teorema

Teorema 2.0.17 *[12] Un núcleo reproductor en un espacio de Hilbert es una función definida positiva.*

Para que sea de utilidad un núcleo reproductor en una SVM, este debe satisfacer cierta condición conocida en la literatura como es la condición de Mercer. El mismo ha jugado un papel crucial en la composición de las SVM y proporciona información valiosa sobre la geometría de los espacios de funciones. Así mismo, éste provee las condiciones que debe cumplir una función k para ser un núcleo.

Teorema 2.0.18 (Mercer [12]) *Suponga $k \in \mathcal{L}_\infty(\mathbb{X}^2)$ es una función a valores reales simétrica tal que el operador integral*

$$T_k : \mathcal{L}_2(\mathbb{X}) \longrightarrow \mathcal{L}_2(\mathbb{X})$$

$$(T_k f)(x) := \int_{\mathbb{X}} k(x, x')f(x')d\mu(x')$$

es definido positivo, es decir, para todo $f \in \mathcal{L}_2(\mathbb{X})$ tenemos

$$\int_{\mathbb{X}^2} k(x, x')f(x)f(x')d\mu(x)d\mu(x') \geq 0.$$

Sean $\psi_j \in \mathcal{L}_2(\mathbb{X})$ funciones propias ortonormales de T_k asociadas a los autovalores $\lambda_j \geq 0$. ordenados en forma no creciente. entonces:

i $(\lambda_j)_j \in l_1$

ii $k(x, x') = \sum_{j=1}^{N_H} \lambda_j \psi_j(x)\psi_j(x')$.

La condición de Mercer permite determinar las propiedades que debe cumplir un producto interno en el espacio de características para que se pueda escribir

a través de un cierto núcleo; a continuación mostramos dos resultados que nos ayudan

Proposición 2.0.19 (Mapa de núcleo de Mercer[12]) *Si k es un núcleo que satisface la condición del teorema de Mercer podemos construir ϕ tal que*

$$\langle \phi(x), \phi(x') \rangle = k(x, x'), \forall x, x' \in \mathbb{X} \tag{2.4}$$

opcionalmente, dado $\epsilon > 0$, existe un mapeo ϕ_n en un espacio n-dimensiónal con producto interno, tal que:

$$|k(x, x') - \langle \phi^n(x), \phi^n(x') \rangle| < \epsilon, \quad \forall x, x' \in \mathbb{X}. \tag{2.5}$$

Teorema 2.0.20 (Condición de Mercer[12]) *Existe una transformación ϕ y una expansión en series $k(x_i, x_j) = \sum \phi(x_i)\phi(x_j)$ si y solo si para cualquier $g(x)$ con $g \in \mathcal{L}_2$ se tiene*

$$\int_{\mathbb{X}^2} k(x, y)g(x)g(y)dxdy \geq 0$$

equivalentemente;

$$\begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots \\ k(x_2, x_1) & \ddots & \\ \vdots & & \end{bmatrix}$$

Es definida positiva.

Nota 2.0.21 *Utilizando la función núcleo no es necesario calcular explícitamente el mapa $\phi : X \longrightarrow H$ para el entrenamiento en el espacio de las características.*

Tanto el núcleo de Mercer como el núcleo definido positivo pueden ser representados como un producto interno en un espacio de Hilbert la siguiente proposición muestra el caso donde estos núcleos coinciden.

Proposición 2.0.22 *[12] Sea* $\mathbb{X} = [a, b]$ *un intervalo compacto y sea* $k : [a, b] \times [a, b] \longrightarrow \mathbb{C}$ *continua entonces k es un núcleo definido positivo si y solo si*

$$\int_a^b \int_a^b k(x, x')f(x)f(x')dxdx' \geq 0 \tag{2.6}$$

para cada función continua $f : \mathbb{X} \longrightarrow C$

Proposición 2.0.23 (continuidad del mapa de características[12]) *Si X es un espacio topológico y k es un núcleo continuo definido positivo en* $\mathbb{X} \times \mathbb{X}$*, entonces existe un espacio Hilbert* $\mathbb{H}$ *y un mapeo* $\phi : \mathbb{X} \longrightarrow \mathbb{H}$ *y un mapeo* $\phi : \mathbb{X} \longrightarrow \mathbb{H}$ *continuo tal que* $\forall x, x' \in \mathbb{X}$*,tenemos:*

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{2.7}$$

Por otro lado, veamos a continuación una caracterización de los núcleos definidos positivos . En el caso que no se obtiene un núcleo definido positivo se puede relajar la condición de definido positivo y obtener otras clases de núcleos como son los condicionalmente definido positivo;

Definición 2.0.24 (Matriz condicionalmente definida positiva[12]) *Una matriz*

K simétrica de orden $m \times m$ $(m \geq 2)$ *tomando valores en* $\mathbb{K}$ *que satisface;*

$$\sum_{i,j=1}^{m} C_i \overline{C}_j K_{ij} \geq 0 \quad \forall C_i \in \mathbb{K}, \quad con \sum_{i=1}^{m} C_i = 0.$$

es llamada condicionalmente definida positiva (CDP)

Definición 2.0.25 (Núcleo Condicionalmente definido positivo[12]) *Diremos que un núcleo k es núcleo condicionalmente definido positivo si la matriz de Gram es condicionalmente definida positiva para todo* $m \geq 2$, $x_1, x_2, \ldots, x_m \in \mathbb{X}$.

Proposición 2.0.26 (Construcción de Núcleo DP desde un núcleo CDP[12]) *Sea* $x_0 \in \mathbb{X}$, *y sea k un núcleo simétrico en* $\mathbb{X} \times \mathbb{X}$ *entonces*

$$\widetilde{k}(x, x') := \frac{1}{2}(k(x, x') - k(x, x_0) - k(x_0, x') + k(x_0, x_0)) \tag{2.8}$$

es definido positivo si y solo si k es condicionalmente definido positivo.

En lo que sigue vamos a ilustrar la mejor escogéncia de la función núcleo para una SVM con ciertos datos de entrenamiento. Queremos expresar como la elección del núcleo puede afectar la precision de la clasificación en las SVM; esto se logra con el uso de una función kernel donde la principal característica es que envía los datos a un espacio de dimensión más alta con la esperanza de que en este espacio los datos si sean separables, además de no existir restricciones sobre la forma de esta asignación, que incluso podría llevar a espacios infinito-dimensiónales. Sin embargo, en la parte computacional es todavía complicado

depende de la cantidad de patrones de entrenamiento para proporcionar una distribución de los datos correcta, ya que en problemas donde la dimensión es alta generalmente requiere un conjunto de entrenamiento grande.

2.1. Tipos de Funciones Núcleos (Kernel)

Los núcleo son funciones continuas, simétricas y preferiblemente deben tener una matriz de Gram definida positiva o (semi definida positiva). Como se mencionó antes se debe tener en cuenta que los núcleos deben satisfacer el teorema de Mercer es decir; sus matrices tienen únicamente valores no negativos; esto es importante pues el uso de un núcleo definido positivo asegura que el problema de optimización será convexo tendrá solución y será única. Sin embargo, muchas funciones núcleo que no son estrictamente definidas positivo también han demostrado funcionar muy bien en la práctica. Un ejemplo es el núcleo sigmoide, que, a pesar de su amplio uso, no es semi definido positivo para ciertos valores de sus parámetros. Además; elegir el núcleo más adecuado depende altamente del problema en cuestión y sus parámetros porque depende de lo que estamos tratando. La motivación detrás de la elección de un núcleo en particular puede ser muy intuitiva dependiendo de qué tipo de información esperamos extraer acerca de los datos. A continuación presentamos algunos tipos de funciones núcleo de la literatura existente.

2.1.1. Funciones núcleos de base radial (RBF)

Una función de base radial (RBF) es una función cuya característica principal es por una salida que disminuye (o aumenta) monótonamente con la distancia de la entrada respecto a un punto fijo llamado centro. Además, poseen la particularidad de generar superficies suaves a partir de un gran número de puntos de datos. Dichas funciones producen buenos resultados para superficies con una variación suave. Una (RBF) son kernel que se pueden escribir en la forma

$$k(x, x') = f(d(x, x')),$$

donde d es una métrica en $\mathbb{X}$, y f es una función en $\mathbb{R}^+$ ver [12]. Por lo general, la métrica surge del producto escalar;

$$d(x, x') = \|x - x'\|.$$

A continuación mencionamos algunos núcleos de base radial pero antes daremos la definición formal de un RBF.

Definición 2.1.1 (Funciones de base radial (RBF)[12]) *Se conocen como funciones de base radial al conjunto de las funciones ϕ que satisfacen la siguiente definición:*

$$\phi : \mathbb{R}^n \mapsto \mathbb{R} : x \mapsto \phi(\|x\|).$$

Donde $x \in \mathbb{R}^n$ y $\|x\|$ es la norma euclidiana en $\mathbb{R}^n$.

- **Núcleo Gaussiano** Este núcleo es conocido comúnmente con una forma

gaussiana;

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \tag{2.9}$$

donde σ determina el ancho del núcleo gaussiano. En estadística, cuando consideramos la probabilidad gaussiana se llama la desviación típica estándar y σ^2.Alternativamente, podría aplicarse también mediante

$$k(x, y) = \exp(-\gamma||x - y||^2) \tag{2.10}$$

El parámetro γ es ajustable desempeña un papel importante en el rendimiento del núcleo y debe ajustarse cuidadosamente al problema en cuestión. Si es sobrevalorado, la exponencial se comporta casi linealmente y la proyección al espacio de dimensión mayor comenzará a perder su poder no-lineal. Por otro lado, si se subestima, la función carece de regularización y el límite de decisión va a ser muy sensible al ruido en datos de entrenamiento.

- **Núcleo exponencial** Este se relaciona estrechamente con el kernel gaussiano, se diferencia a este solo en el cuadrado de la norma;

$$k(x, y) = \exp\left(-\frac{||x - y||}{2\sigma^2}\right). \tag{2.11}$$

Éste produce una solución lineal a trozos.

- **Núcleo laplaciano** Es completamente equivalente a el Núcleo exponencial, excepto por ser menos sensible a los cambios en el parámetro σ.

$$k(x, y) = \exp\left(-\frac{||x - y||}{\sigma}\right) \tag{2.12}$$

Es importante señalar que las observaciones realizadas sobre el parámetro σ para el núcleo gaussiano también se aplican a los núcleos exponenciales y Laplaciano.

- **Núcleo ANOVA** Este núcleo es muy similar a los núcleos gaussianos y laplacianos. Se dice que poseen un buen desempeño en los problemas de regresión multidimensiónales ([13]).

$$k(x, y) = \sum_{d=1}^{n} exp(-\sigma(x^j - y^j)^2)^d \tag{2.13}$$

- **Núcleo de tangente hiperbólica (sigmoide)** El núcleo tangente hiperbólica también se conoce como el núcleo sigmoide y como el núcleo Perceptrón multicapa (MLP). El núcleo sigmoide proviene del campo de las redes neuronales, donde la función sigmoidea bipolar se utiliza a menudo como una función de activación de las neuronas artificiales. Este núcleo viene dado por

$$k(x, y) = \tanh(\langle \alpha x^T, y \rangle + c) \tag{2.14}$$

Es interesante observar que un modelo SVM utilizando una función kernel sigmoide es equivalente a una de dos capas, red neuronal o perceptrón. Este núcleo fue muy popular para las máquinas de vectores soporte debido a su origen de la teoría de redes neuronales. Además, a pesar de ser sólo condicionalmente definida positiva, se ha encontrado para llevar a cabo bien en la práctica. Hay dos parámetros ajustables en el núcleo sigmoide, el valor escalar α y el valor constante c. Un valor común para α es $\frac{1}{N}$, donde N es la dimensión de datos.

- **Perceptron:**

$$K(x_i, x_j) = \|x_i - x_j\| \tag{2.15}$$

- **Núcleo B-Spline** Es una función de base radial y está definida en el intervalo $[-1, 1]$. Está dado por la fórmula recursiva:

$$k(x, y) = B_{2p+1}(x - y) \tag{2.16}$$

donde $p \in \mathbb{N}$ con $B_{i+1} := B_i \bigotimes B_0$ Alternativamente, B_n se puede calcular utilizando la expresión explícita

$$B_n(x) = \frac{1}{n!} \sum_{k=0}^{n+1} \binom{n+1}{k} (-1)^k (x + \frac{n+1}{2} - k)_+^n$$

2.1.2. Funciones núcleos de base no necesariamente radial

- **Núcleo Lineal** Es la función de núcleo más simple. Es dado por el producto interno $\langle x, y \rangle$ además una constante c.

$$k(x, y) = x^T y + c \tag{2.17}$$

- **Núcleo Polinomial** Es un núcleo no estacionarios. estos núcleos son ideales para problemas donde todos los datos de entrenamiento se normalizan entre estos tenemos

  1. **Polinomial-homogéneo:**

$$K(x_i, x_j) = (\langle x_i, x_j \rangle)^n \tag{2.18}$$

2. **Polinomial No Homogéneo**

$$K(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d \tag{2.19}$$

donde $d \in \mathbb{N}$ y $c \geq 0$ es un parámetro libre, el grado polinómico viene dado por d

- **Núcleo cuadrático racional** El núcleo cuadrático Racional es menos intensivas computacionalmente que el núcleo gaussiana y se puede utilizar como una alternativa cuando se utiliza el Gaussian llega a ser demasiado costoso.

$$k(x, y) = 1 - \frac{||x - y||^2}{||x - y||^2 + c} \tag{2.20}$$

- **Núcleo Multicuadrático:** Se puede utilizar en las mismas situaciones como el núcleo cuadrático Racional. Como es el caso con el núcleo sigmoide, también es un ejemplo de un núcleo no definido positivo.

$$k(x, y) = \sqrt{||x - y||^2 + c^2} \tag{2.21}$$

- **Núcleo inverso multicuadrático** Al igual que con el núcleo gaussiano, resulta una matriz de núcleo con rango completo ([9]) y por lo tanto forma un espacio de características dimensión infinita.

$$k(x, y) = \frac{1}{\sqrt{||x - y||^2 + c^2}} \tag{2.22}$$

- **Núcleo circular** El núcleo circular se utiliza en aplicaciones geostáticas. Es un ejemplo de un núcleo isotrópica estacionaria y es definida positiva en $\mathbb{R}^2$.

$$k(x,y) = \frac{2}{\Pi}\arccos\left(-\frac{||x-y||}{\sigma}\right) - \frac{2}{\Pi}\frac{||x-y||}{\sigma}\sqrt{1-\left(\frac{||x-y||}{\sigma}\right)^2} \tag{2.23}$$

$$si\ ||x-y|| < \sigma,\ 0\ otro\ caso$$

- **Núcleo esférico** Es similar al del núcleo circular, pero es definida positiva en $\mathbb{R}^3$

$$k(x,y) = 1 - \frac{3}{2}\left(\frac{||x-y||}{\sigma}\right) + \frac{1}{2}\left(\frac{||x-y||}{\sigma}\right)^3 \tag{2.24}$$

$$si\ ||x-y|| < \sigma, \qquad 0\ otro\ caso$$

- **Núcleo potencia** También se conoce como el (no rectificado) del núcleo triangular. Es un ejemplo del núcleo en escala invariante (ver [3]) y este es sólo condicionalmente definida positiva.

$$k(x,y) = -||x-y||^d \tag{2.25}$$

- **Núcleo Log** El núcleo de registro parece ser particularmente interesante para las imágenes, pero es sólo condicionalmente definida positiva.

$$k(x,y) = -\log(||x-y||^d + 1) \tag{2.26}$$

- **Núcleo Spline** El núcleo Spline se administra como una sola pieza de un

polinomio cúbico, como se deriva de las obras de ([4])

$$k(x,y) = 1 + xy + xy\,\text{mín}(x,y) - \frac{x+y}{2}\,\text{mín}(x,y)^2 + \frac{1}{3}\,\text{mín}(x,y)^3 \tag{2.27}$$

Sin embargo el significado en realidad es;

$$k(x,y) = \prod_{i=1}^{d} 1 + x_i y_i + x_i y_i\,\text{mín}(x_i,y_i) - \frac{x_i+y_i}{2}\,\text{mín}(x_i,y_i)^2 + \tfrac{1}{3}\,\text{mín}(x_i,y_i)^3 \tag{2.28}$$

con $x, y \in \mathbb{R}^d$

- **Núcleo Bessel** Es bien conocido en la teoría de la función de los espacios de suavidad fraccionada.Viene dado por:

$$k(x,y) = \frac{J_{v+1}(\sigma\|x-y\|)}{\|x-y\|^{-n(v+1)}} \tag{2.29}$$

donde J es la función de Bessel de primera especie. Sin embargo, el núcleo de Bessel se dice que es:

$$k(x,y) = -Bessel^n_{(v+1)}(\sigma(|x-y|^2)) \tag{2.30}$$

- **Núcleo Cauchy** Viene de la distribución de Cauchy. El mismo se puede utilizar para incidir en el largo alcance y la sensibilidad sobre el espacio de alta dimensión.

$$k(x,y) = \frac{1}{1 + \frac{\|x-y\|^2}{\sigma^2}} \tag{2.31}$$

- **Núcleo Chi-cuadrado** El núcleo de chi-cuadrado proviene de la distribución chi-cuadrado

$$k(x, y) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)^2} \tag{2.32}$$

Sin embargo, como señaló comentarista Alexis Mignon, esta versión del kernel es sólo condicionalmente definida positiva (CPD). Una versión definida positiva de este núcleo es dado según ([1]), por;

$$k(x, y) = \sum_{i=1}^{n} \frac{2x_i y_i}{(x_i + y_i)} \tag{2.33}$$

y es adecuado para ser utilizado por métodos distintos de las máquinas de vectores de soporte.

- **Núcleo intersección histograma**

Este núcleo también se conoce como el núcleo Min y se ha demostrado útil en la clasificación de imágenes, posiblemente la forma más sencilla para representar información de color de imágenes es proporcionado por histogramas. Los diferentes espacios de color (RGB, CMYK, HSV, etc.) dan lugar a diferentes representaciones de las imágenes que mejoran o atenuar color y características determinadas. Es dado por;

$$k(x, y) = \sum_{i}^{n} \text{mín}(x_i, y_i) \tag{2.34}$$

- **Núcleo Generalizado Intersección de histograma**

Es construido en base a la intersección de histograma es un núcleo para la

clasificación de imágenes sino que se aplica en una variedad mucho más amplia de contextos ([11]). Viene dada por:

$$k(x, y) = \sum_{i=1}^{m} \text{mín}(|x_i|^{\alpha}, |y_i|^{\beta}) \tag{2.35}$$

- **Núcleo generalizado T-student** Este nucleo se ha demostrado ser un núcleo Merced, teniendo así una matriz del núcleo semi-definida positiva (Boughorbel, 2004). Viene dado por:

$$k(x, y) = \frac{1}{1 + ||x - y||^{d}} \tag{2.36}$$

- **Nucleo Bayesiano** El núcleo Bayesiano podría ser dado como:

$$k(x, y) = \Pi_{l=1}^{N} k_l(x_l, y_l) \tag{2.37}$$

donde,

$$k_l(a, b) = \Sigma_{c\in 0,1} P(Y = c|X_l = a)P(Y = c|X_l = b) \tag{2.38}$$

Sin embargo, depende es del problema que se está modelando.

- **Núcleo Wave** El núcleo de la onda también es semidefinida positiva simétrica (Huang, 2008).

$$k(x, y) = \frac{\theta}{||x - y||} \sin \frac{||x - y||}{\theta} \tag{2.39}$$

- **Núcleo wavelet:** El núcleo Wavelet (Zhang et al, 2004) proviene de la teoría

Wavelet y se da como:

$$k(x,y) = \prod_{i=1}^{N} h(\frac{x_i - c}{a})h(\frac{y_i - c}{a}) \tag{2.40}$$

Donde a y c son la dilatación y traslación en coeficientes de onda, respectivamente. Una versión invariante por traslación de este núcleo se puede dar como:

$$k(x,y) = \prod_{i=1}^{N} h(\frac{x_i - y_i}{a}) \tag{2.41}$$

Cuando en ambas $h(x)$ denota una función madre wavelet. En el artículo de Li Zhang, Weina Zhou, y Licheng Jiao, el autor sugiere un posible $h(x)$ como:

$$h(x) = \cos(1{,}75x)\exp(\frac{-x^2}{2}) \tag{2.42}$$

3

Resultados Numéricos

En este capítulo presentamos los resultados del proceso de clasificación en varios conjuntos de datos usando una base de datos generadas en forma aleatoria para una SVM. La implementación algorítmica que se realizó fue utilizando el software Matlab (R2011a), en un computador con procesador (Intel (R) Core (TM)i3 CPU y RAM 2 GB). Con el objetivo, de resolver el problema de programación cuadrática planteado en el modelo propuesto por O. L. Mangasarian (ver [6]) en el capítulo 1 para ello, se utilizó el programa **quadprog.m** del toolbox de optimization de Matlab. Además, se presentan diversas formas en que pueden estar los puntos en el plano que no necesariamente son separables a través de una superficie lineal.

Las pruebas consisten en aplicar clasificación de la data con diferentes funciones Kernel a saber; núcleo general, gaussiano, polinomial homogéneo, y polinomial no homogéneo para así comparar el comportamiento de estos en la data.

A continuación por cada muestra en los diferentes conjunto de datos a clasi-

ficar se presentara una tabla donde se especificará la función núcleo empleada, la cantidad de datos a clasificar por tipo, mas si la función clasificó o no, el número de vectores soportes y el tiempo que tardó el programa en ejecutarse y hacer la separación. Luego se ilustrara las imágenes la cual señala el núcleo utilizado los parámetros escogidos para la clasificación y el número de iteraciones en que se realizó, relacionado con los resultados expuestos en cada una de las tabla.

Ejemplo 3.0.2 *Consideremos la primera data de nombre* **Número siete***, con una cantidad de puntos a clasificar de* 1075 *puntos, y una tolerancia de* $10e^{-3}$*, con un número máximo de iteraciones* $\nu = 10000$;.

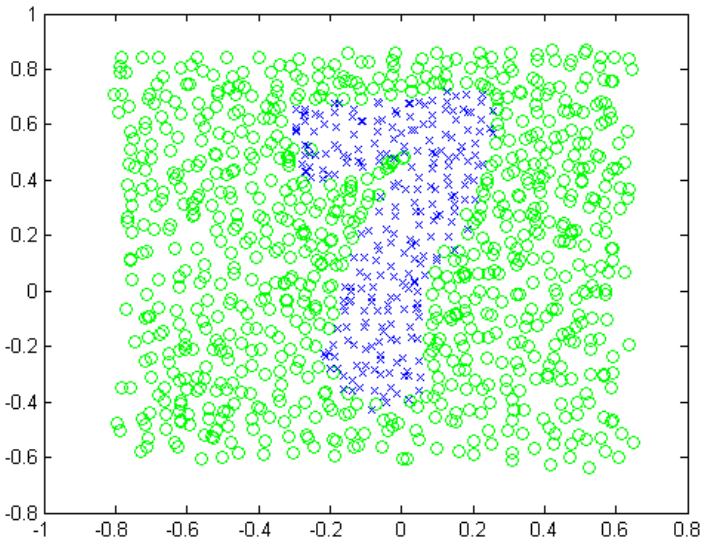
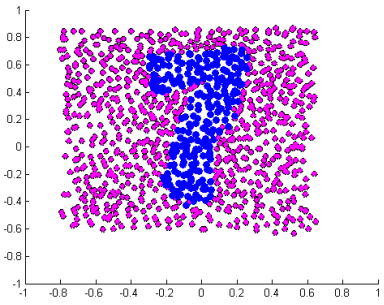
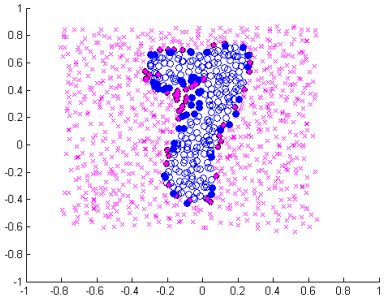


Figura 3.1: Número siete

Núcleo	Clasifica	Vectores Soporte	Tiempo
General	No	1075	191,24 seg
Gaussiano	Si	103	110,91 seg

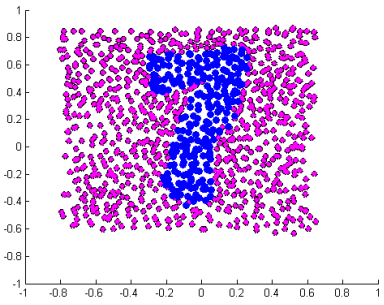


Núcleo General

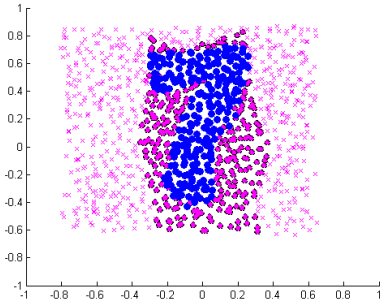


Núcleo Gausiano

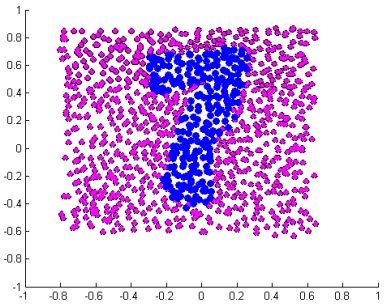
Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
homogéneo	k=7	No	1075	153,91 seg
homogéneo	k=8	No	527	72,49 seg
homogéneo	k=9	No	1075	47,67 seg
homogéneo	k=10	No	627	121,25 seg



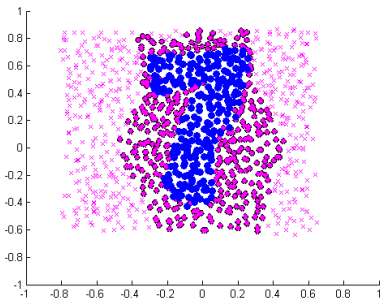
Núcleo polinomial homogéneo $k = 7$



Núcleo polinomial homogéneo $k = 8$

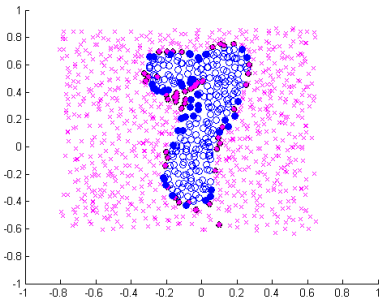


Núcleo polinomial homogéneo $k = 9$

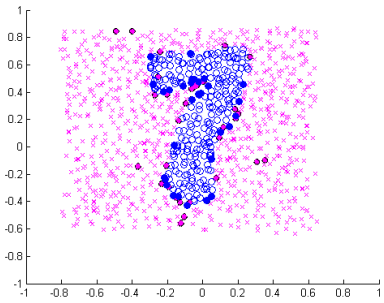


Núcleo polinomial homogéneo $k = 10$

Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
No homogéneo	d=6, c=4	Si	101	480,17 seg
No homogéneo	d=10, c=3	Si	68	605,35 seg



Núcleo polinomial no homogéneo $d = 6, c = 3$



Núcleo polinomial no homogeny $d = 10, c = 3$

Como se pudo observar en esta data el comportamiento de los núcleos refleja lo siguiente; en el núcleo general y el núcleo polinomial homogéneo para el parámetro de índice par e impar no clasificó no tuvo separación la data, además se observo que los mejores comportados en este caso fueron el núcleo gaussiano y el núcleo polinomial no homogéneo con una cantidad minima de vectores soportes.

Ejemplo 3.0.3 *Consideremos la segunda data de nombre* **Escalera***, con una cantidad de puntos a clasificar de* 1057 *puntos, y una tolerancia* $10e^{-3}$*, con un número máximo de iteraciones* $\nu = 10000$;

Núcleo	Clasifica	Vectores Soporte	Tiempo
General	Si	340	340,24 seg
Gaussiano	Si	260	113,57 seg

Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
homogéneo	k=3	Si	289	263,50 seg
homogéneo	k=6	No	474	226,24seg
homogéneo	k=10	No	783	211,98 seg

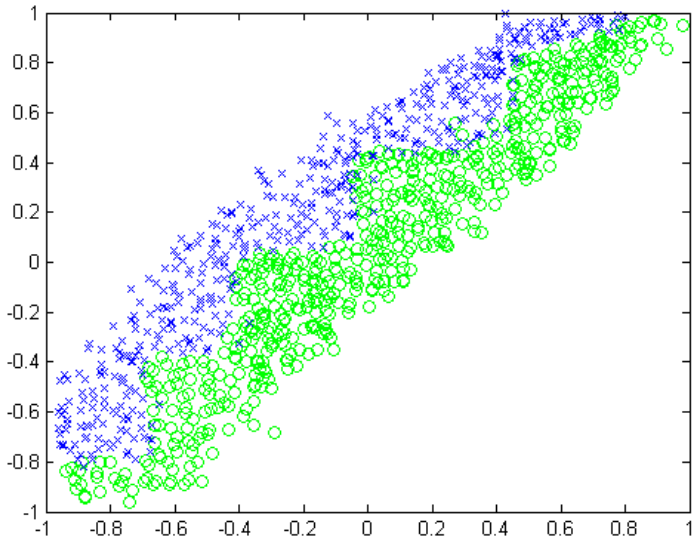
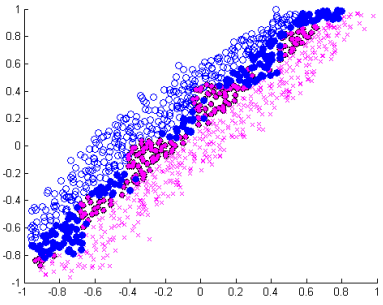
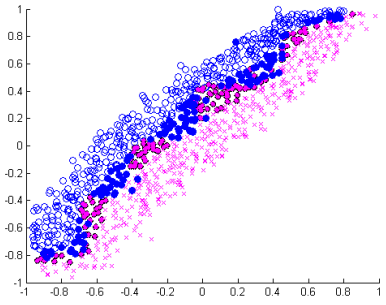


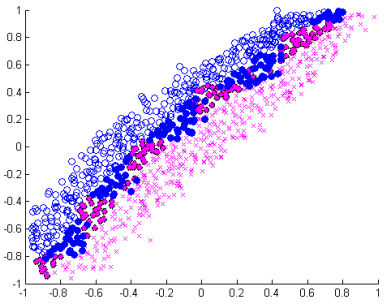
Figura 3.2: Escalera



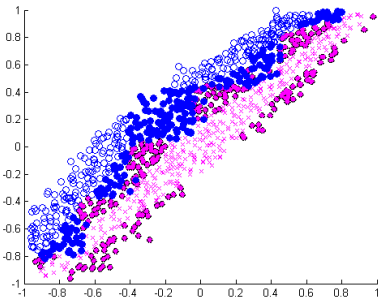
Núcleo General



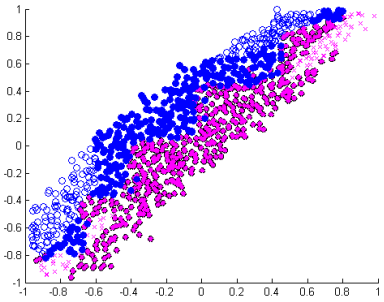
Núcleo Gausiano



Núcleo polinomial homogéneo $k = 3$

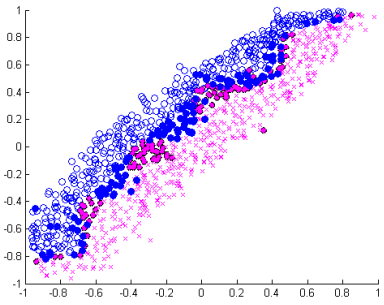


Núcleo polinomial homogéneo $k = 6$

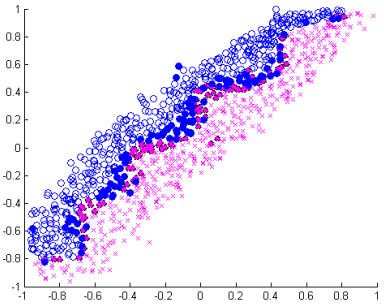


Núcleo polinomial homogéneo $k = 10$

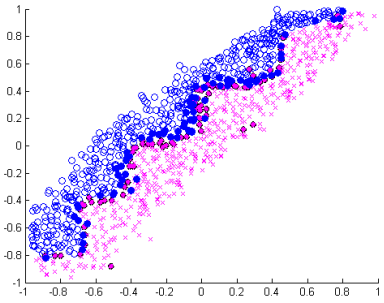
Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
No homogéneo	d=8, c=3	Si	229	453,28seg
No homogéneo	d=9, c=2	Si	202	400,18 seg
No homogéneo	d=10, c=2	Si	162	360,74 seg
No homogéneo	d=15, c=1	Si	114	482,28 seg
No homogéneo	d=19, c=1	Si	100	296,81 seg



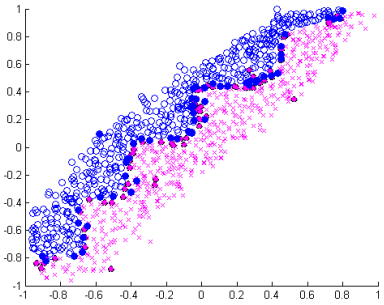
Núcleo polinomial no homogéneo $d = 8, c = 3$



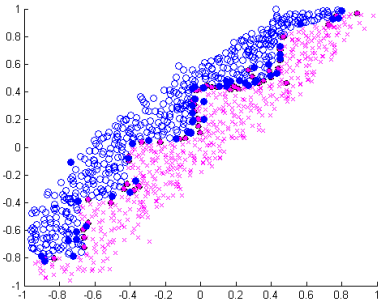
Núcleo polinomial no homogéneo $d = 9, c = 2$



Núcleo polinomial no homogéneo $d = 10, c = 2$



Núcleo polinomial no homogéneo $d = 15, c = 1$



Núcleo polinomial no homogéneo $d = 19, c = 1$

En esta data se pudo observar el comportamiento de los núcleos donde en el caso del núcleo general y núcleo gaussiano clasificó, aunque en este caso se mantuvieron muchos vectores soportes. Mientras que en núcleo polinomial homogéneo hay dos casos ; en el caso que el parámetro es de índice par no clasificó, pero en el caso de índice impar si clasificó pero con una gran cantidad de vectores soportes. Por otra parte, el núcleo polinomial no homogéneo en las primeras pruebas con parámetro pequeños de tanto de índice par como impar clasifica pero con un mayor número de vectores soportes, mientras que en el caso de parámetro mas alto los datos fueron mejor comportados logrando clasificar con una cantidad bastante minima de vectores soportes.

Ejemplo 3.0.4 *Consideremos la tercera data de nombre* **Dona***, con una cantidad de puntos a clasificar de* 1187 *puntos, y una tolerancia de* $10e^{-3}$*, con un número máximo de iteraciones* $\nu = 10000$;.

Núcleo	Clasifica	Vectores Soporte	Tiempo
General	No	1187	169,76 seg
Gaussiano	Si	45	182,58seg

Cuadro 3.1: Datos 3: Dona

Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
homogéneo	k=6	No	261	164,67 seg
homogéneo	k=7	No	1187	101,53 seg
homogéneo	k=9	No	1187	115,32 seg
homogéneo	k=10	No	822	133,11seg

Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
No homogéneo	d=10, c=2	Si	35	578,76 seg
No homogéneo	d=15, c=3	Si	31	880,33 seg
No homogéneo	d=20, c=1	Si	32	234,55 seg

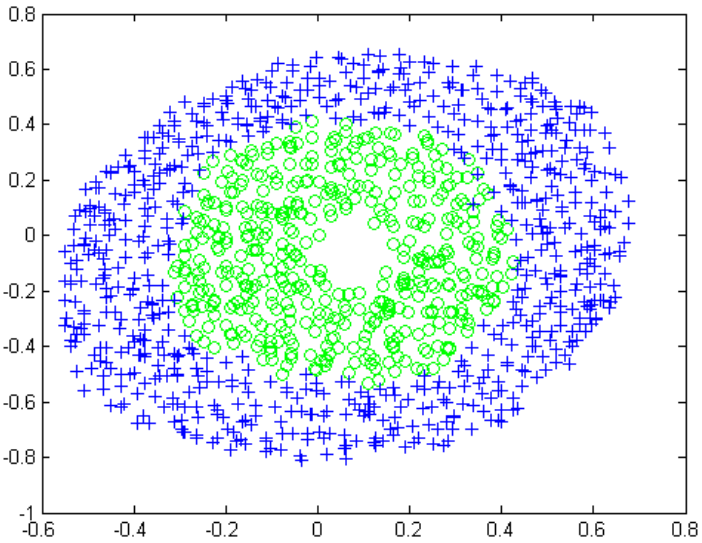
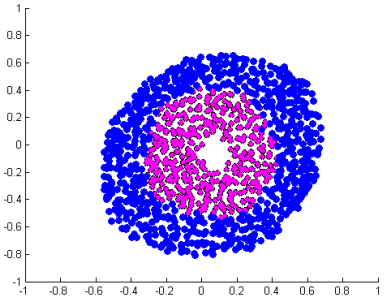
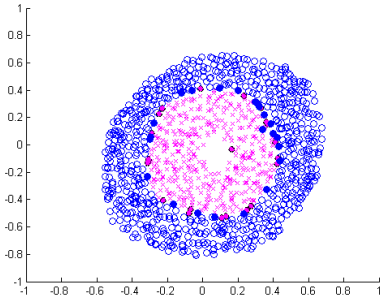


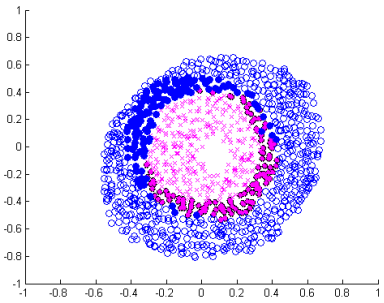
Figura 3.3: Dona



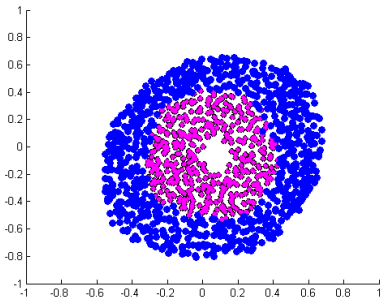
Núcleo General



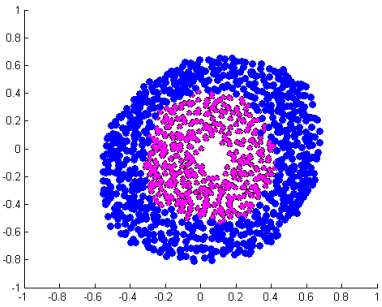
Núcleo Gausiano



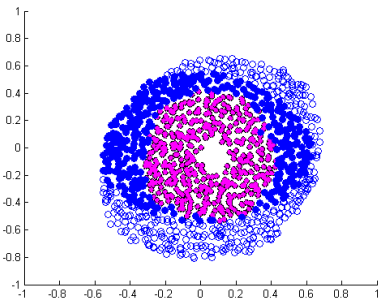
Núcleo polinomial homogéneo $k = 6$



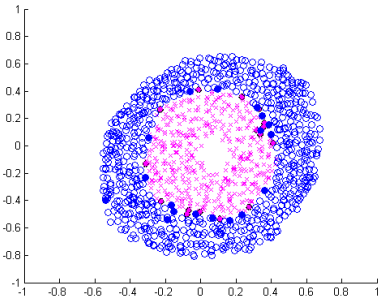
Núcleo polinomial homogéneo $k = 7$



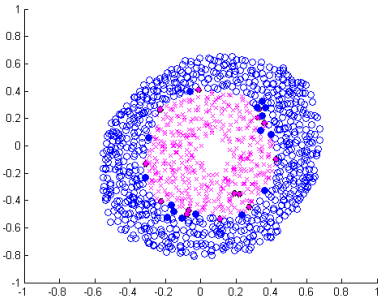
Núcleo polinomial homogéneo $k = 9$



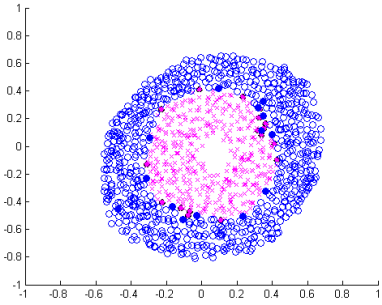
Núcleo polinomial homogéneo $k = 10$



Núcleo polinomial no homogéneo $d = 10, c = 2$



Núcleo polinomial no homogéneo $d = 15, c = 3$



Núcleo polinomial no homogéneo $d = 20, c = 1$

En esta data se pudo observar que las funciones núcleos aplicadas se comportaron de la siguiente forma; en el núcleo general no clasificó la data, con el núcleo gaussiano si clasificó con una cantidad bastante pequeña de vectores soportes, además en núcleo polinomial homogéneo en ambos casos donde los parámetros toman valor tanto par e impar no clasificó. Mientras que el núcleo polinomial no homogéneo si clasifico y de hecho fue el que mejor hizo la separación en comparación con el núcleo gaussiano puesto que la cantidad de vectores soportes fue mínima.

Ejemplo 3.0.5 *Consideremos la Cuarta data de nombre* **Letra Y**, *con una cantidad de puntos a clasificar de* 880 *puntos, y una tolerancia de* $10e^{-3}$, *con un número máximo de iteraciones* $\nu = 10000$;.

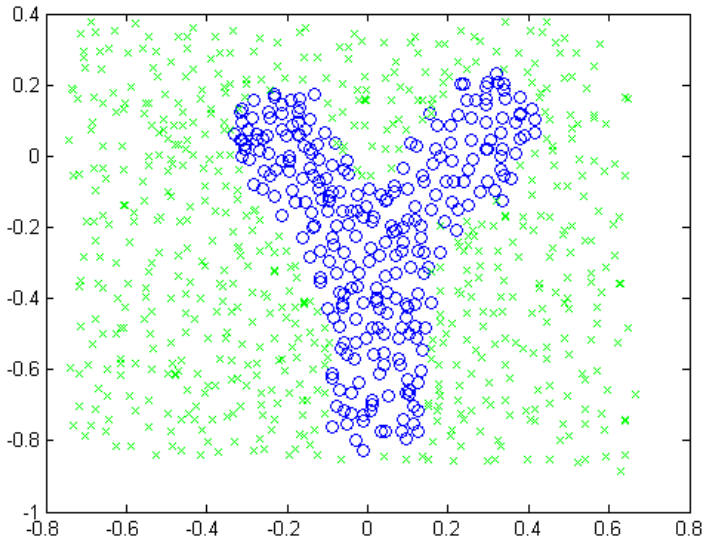
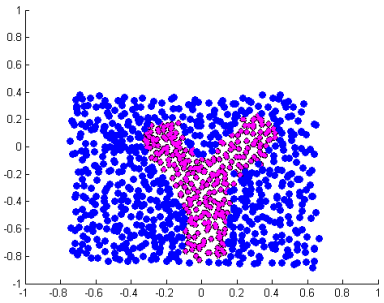
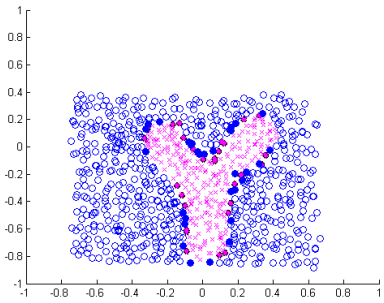


Figura 3.4: Letra Y

Núcleo	Clasifica	Vectores Soporte	Tiempo
General	No	880	105,88 seg
Gaussiano	Si	61	86,04 seg

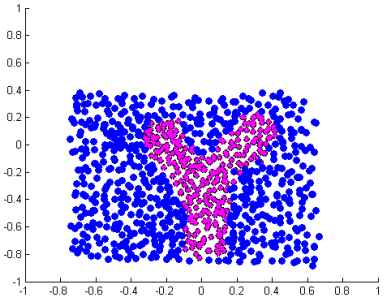


Núcleo General

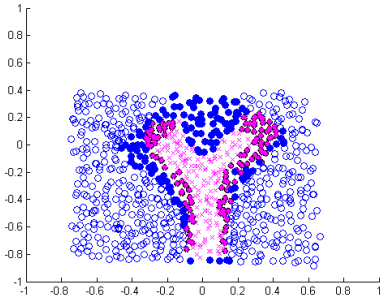


Núcleo Gausiano

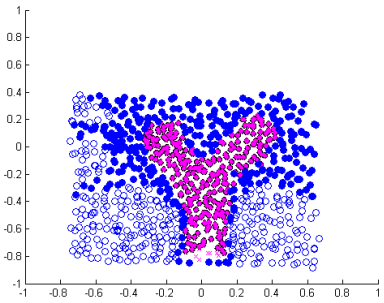
Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
homogéneo	k=3	No	880	83,33 seg
homogéneo	k=4	No	263	95,07 seg
homogéneo	k=6	No	293	80,95 seg
homogéneo	k=7	No	559	118,22 seg



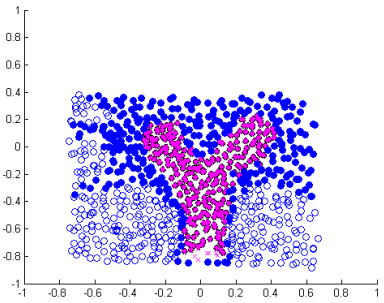
Núcleo polinomial homogéneo $k = 3$



Núcleo polinomial homogéneo $k = 4$

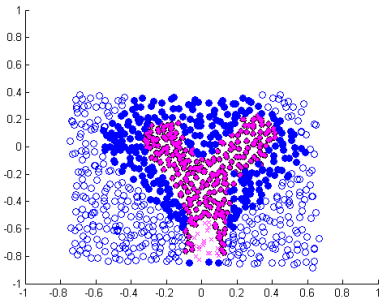


Núcleo polinomial homogéneo $k = 6$

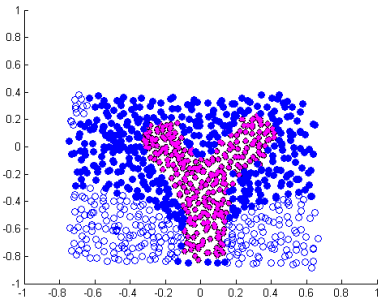


Núcleo polinomial homogéneo $k = 7$

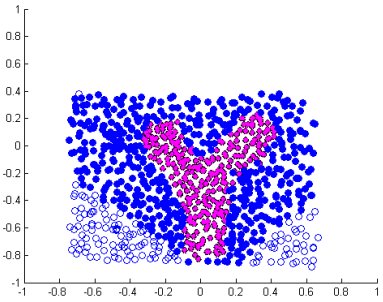
Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
homogéneo	k= 10	No	520	69,74 seg
homogéneo	k= 15	No	642	202,65 seg
homogéneo	k=20	No	748	84,52 seg



Núcleo polinomial homogéneo $k = 10$

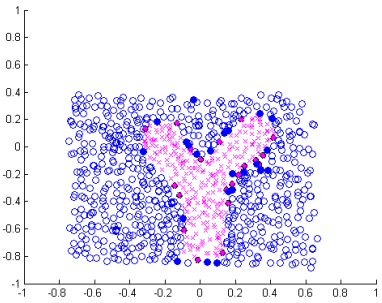


Núcleo polinomial homogéneo $k = 15$

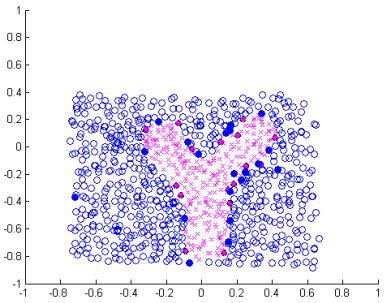


Núcleo polinomial homogéneo $k = 20$

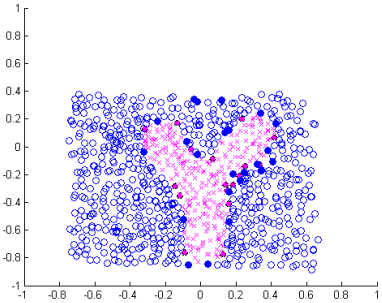
Núcleo polinomial	Parámetros	Clasifica	Vectores Soporte	Tiempo
No homogéneo	d=10, c=3	Si	45	280,44 seg
No homogéneo	d=20, c=2	Si	38	438,78 seg
No homogéneo	d=21, c=1	Si	42	241,08 seg



(b) Núcleo polinomial no homogéneo $d = 10$ y $c = 3$



(a) Núcleo polinomial no homogéneo $d = 20$ y $c = 2$



(b) Núcleo polinomial no homogéneo $d = 21$ y $c = 1$

En estos datos se pudo observar lo siguiente; en el núcleo general no clasificó, mientras que con el núcleo gaussiano si clasificó con una cantidad bastante pequeña de vectores soportes y en poco tiempo, además en núcleo polinomial homogéneo en ambos casos donde los parámetros toman valor par e impar no clasifico. Sin embargo con el núcleo polinomial no homogéneo si clasificó con una cantidad minima de vectores soportes.

Conclusiones

A lo largo de este trabajo pudimos explicar las SVM en un problema de optimización aplicando funciones núcleos entre ellas las mas comúnmente utilizadas como; general, gaussiano, polinomial homogéneo y no homogéneo. Durante todo el proceso de realización y estudio de los datos se hizo uso de la programación cuadrática convexa. Se estudio y analizo los diferentes funciones núcleos (Kernels) empleados para transformar datos no separables linealmente a un espacio de mayor dimensión en donde si logran ser separados. Además de esto, se estudiaron propiedades estructurales de la función Kernel y su implicación en la máquinas de vectores de soporte realizando un estudio comparativo de distintos núcleos (Kernels), empleado a las máquinas de vectores de soporte para la clasificación de distintos tipos de data.

Entre los trabajo a seguir esta el establecer una estrategia algorítmica alternante donde se pueda decidir que tipo de funciones núcleos es mas eficiente en un conjunto determinado de datos.

Bibliografía

[1] Andrew Zisserman Member Andrea Vedaldi, Member. Efficient additive kernels via explicit feature maps. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. XX, NO. XX*, 2011.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society ,Vol. 68, No. 3, pp. 337-404.*, May, 1950.

[3] F. Fleuret and H. Sahbi. Scale-invariance of support vector machine based on the triangular kernel. In *Proceedings of the workshop on Stadistical...*, 2003.

[4] Steve Gunn. Support vector machines for classification and regression. *ISIS Technical Report*, 1998.

[5] Gerardo R. Chacón Laura Sánchez Gómez. TeorÍa de nÚcleos reproductivos

en espacios de hilbert y aplicaciones a mÁquinas de soporte vectorial. Mayo 2012.

[6] O. L. Mangasarian. Generalized support vector machines. *Advances in Neural Information Processing Systems*, pages 135–146, 1999.

[7] O. L. Mangasarian. Data mining via support vector machines. In E. W. Sachs and R. Tichatschke, editors, *System Modeling and Optimization XX*, pages 91–112. Kluwer Academic Publishers, Boston, 2003.

[8] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

[9] Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation Volume 2*, 1986.

[10] R. T. Rockafellar. *Convex analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[11] Nozha Boujemaa Sabri Boughorbel, Jean-Philippe Tarel. Generalized histogram intersection kernel for image recognition. *INRIA Rocquencourt, IMEDIA, 78153 Le Chesnay, France LCPC, DESE, 58 Bd Lefebvre, 75015 Paris, France*, 2005.

[12] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[13] Bernhard Scholkopf Thomas Hofmann and Alexander J. Smola. Kernel methods in machine learning. *Institute of Mathematical Statistics*, 2008.

[14] V. N. Vapnik. The nature of statistical learning theory. *Springer-Verlag New York, Inc., New York, NY, USA,*, 1995.

[15] V. N. Vapnik. *Statistical learning theory*, volume 1 of *Adaptative and learning systems for signal processing, comunications and control*. Wiley New York, 1998.

[16] Stanisław Zaremba. *Zarys pierwszych zasad teoryi liczb całkowitych*. nakł. Akademii Umiejętności, 1907.

A

Apéndice

A.1. graficadepuntos.m

En este programa La función graficadepuntos permite crear el conjunto de datos; representado en matrices de las dos clases de puntos, donde $A0$ y $A1$ son los puntos en la forma y a gusto que desee la persona, generados mediante el uso del cursor simultáneamente en un plano.

```
1 function [A0,A1,A]=graficadepuntos(p,Datos)
2  % p=ejes permite el ajuste de los ejes p=[-a b -c d].
3  axis(p)
4  sw=0;
5  A0=[];A1=[];
6  while sw==0
7    grid on
8    [x0 y0]=ginput(); %grafica de una clase puntos
```

```
9     A0=[A0;x0 y0];
10    plot(A0(:,1),A0(:,2),'o','MarkerSize',6,'...
          MarkerFaceColor','m','LineWidth',2,'MarkerEdgeColor'...
          ,'k')
11    axis(p)
12    grid on
13    [x1 y1]=ginput(); %grafica de una otra clase puntos
14    A1=[A1;x1 y1];
15    hold on
16    plot(A1(:,1),A1(:,2),'o','MarkerSize',6,'...
          MarkerFaceColor','b','LineWidth',2,'MarkerEdgeColor'...
          ,'k')
17    nt=size(A0)+size(A1);
18    NT=nt(1)
19    sw=input('satisfecho?')
20    %%si estas satisfecho con el tamano de la data.
21  end
22 A=[A0;A1];
23 save(Datos,'A0','A1','A') % guardar la data
```

A.2. svmgeneral.m

Este algoritmo consigue la solución a un problema de programación cuadrática convexa que describe una svm estándar descrito por [6], [7] en sus trabajos.

Dicha máquina tine dos tareas principales clasificación de los puntos o datos por una superficie no-necesariamente lineal en el espacio original de entradas, maximizando la distancia entre la separación de los planos en un espacio de mayor dimensión. Es alli donde entran las funciones núcleos en la svm estandar planteada por [6], [7], haciendo un llamado en el programa como otra función *KA* de nombre nucleomenu.m. Realizando el mismo una separación bastante optima en el conjunto de datos generados de forma aleatoria. Es importante destacar que se hace uso del toolbox de optimización de MATLAB llamado quadprog, por lo que, los cálculos realizados dentro del algoritmo son desarrollado en [6] planteado como en (1.22).

```
1  function [u,gamma,y,aux,FVAL,EXITFLAG,OUTPUT,LAMBDA,AA0,...
       AA1,tiempo]=svmgeneral(A0,A1,nu,p)
2  tic;
3  A=[A0;A1]; %data generada por generadoradedatos
4   n0=size(A0)
5   n1=size(A1)
6   d=[ones(n0(1),1);-ones(n1(1),1)];
7   D=diag(d);
8   KA=nucleomenu(A); %KA funcion nucleo menu
9   DAD=D*KA*D;
10  m=n0(1,1) + n1(1,1);
11  E=ones(m,1);
12  DE=D*E;
13  Id=eye(m,m);
14  O1=zeros(m,m);
```

```
15 O2=zeros(m,1);
16 %E1=ones(m,1);
17 B1=[-DAD DE -Id];
18 B2=[O1 O2 -Id];
19 AB=[B1;B2];
20 C=[-E;O2];
21 O3=zeros(m,1);
22 H= eye(m,m); %H ES SIMETRICA DEFINIDA POSITIVA
23 H1=[H O2 O1];
24 H2=[O3' 0 O3'];
25 H3=[O1 O2 O1];
26 HH=[H1; H2; H3];
27 f= nu*[O3; 0; E];
28 options=optimset('maxIter',10000,'Algorithm','interior-...
      point-convex');
29 [X,FVAL,EXITFLAG,OUTPUT,LAMBDA]=quadprog(HH,f,AB,C...
      ,[],[],[],[],[],options);
30 u=X(1:m);gamma=X(m+1);y=X(m+2:length(X));
31 %XX=[X(1:m+1);O2];
32 aux=B1*X+E
33 %abs(aux(k))
34 %BUSQUEDA DE VECTORES SOPORTES
35 %para A0
36 cont0=0;
37 hold on
```

```
38  for k=1:n0(1)
39      if abs(aux(k))<1.e-2
40          cont0=cont0 + 1;
41          AA0(cont0,:)=A0(k,:);
42          plot(AA0(:,1),AA0(:,2),'x','MarkerSize',4,'...
                MarkerFaceColor','m','LineWidth',4,'...
                MarkerEdgeColor','m')
43          axis(p)
44         % pause
45      end
46  end
47   plot(AA0(:,1),AA0(:,2),'kx','LineWidth',4)
48   %BUSQUEDA DE VECTORES SOPORTES
49   %para A1
50   cont1=0;
51    for j=1:n1(1)
52       if abs(aux(n0(1)+j))<1.e-2
53  % j
54  % aaa=aux(j)
55          cont1=cont1 + 1;
56          AA1(cont1,:)=A1(j,:);
57          plot(AA1(:,1),AA1(:,2),'o','MarkerSize',4,'...
                LineWidth',4)
58          axis(p)
59         % pause
```

```
60      end
61    end
62   %cont1
63 % AA0
64 % AA1
65    plot(A0(:,1),A0(:,2),'xm')
66    %'MarkerFaceColor','m'
67    hold on %grafica de ambos vectores
68    plot(A1(:,1),A1(:,2),'ob')
69    plot(AA0(:,1),AA0(:,2),'x','MarkerSize',4,'...
         MarkerFaceColor','m','LineWidth',4,'MarkerEdgeColor',...
         'm')
70    plot(AA1(:,1),AA1(:,2),'o','MarkerSize',4,'LineWidth',4)...
         %,'MarkerEdgeColor','b','MarkerFaceColor','b')
71    tiempo=toc
72    %para guardar la data
73    Datos=input('ingrese el nombre de la data: ','s')
74     save(Datos,'u','gamma','y','aux','FVAL','EXITFLAG','...
          OUTPUT','LAMBDA','AA0','AA1','tiempo')
```

A.3. nucleomenu.m

Acá se describe las diferentes funciones núcleos empleadas en las data generadas de forma aleatorios, permitiendo escoger que Kernel usar.

```
function KA=nucleomenu(A)

choice = menu('menu','general','gausiano','Phomogeneo','...
   Pno-homogeneo')
n=size(A);
switch choice
   case 1
%nucleo general
KA=A*A';

   case 2
%nucleo gausiano
sigma=2*norm(std(A))^2;
for i=1:n(1)
   for j=1:n(1)
      KA(i,j)=exp(-norm(A(i,:)-A(j,:))^2/sigma);
   end
end

   case 3
%nucleo polinomial homogeneo
k=input('ingrese el valor de la k: ')
for i=1:n(1)
   for j=1:n(1)
      KA(i,j)=(A(i,:)*(A(j,:))')^(k);
```

```
25      end
26  end
27      case 4
28  % %nucleo polinomial no-homogeneo
29
30  % d es un natural el grado del polinomio y c es un ...
       parametro libre mayor que 0
31  d=input('ingrese el grado del polinomio d: ')
32  c=input('ingrese el valor del parametro libre c: ')
33
34  KA=(A*A'+c).^d;
35  end
```