MÁQUINAS DE SOPORTE VECTORIAL (SVM)

Monroy Jordan, Edgar David

Pérez Neira, Jairo Enrique

Gómez Vásquez, Eduardo

Director

Universidad Tecnológica De Bolívar

Ingeniería electrónica

Cartagena De Indias

2005

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Este trabajo va dirigido a dar a conocer las máquinas de aprendizaje y dejar una guía de referencia de estas, especialmente de la máquinas de soporte vectorial *SVM,* en una forma didáctica y descriptiva de sus aspectos básicos[*], principalmente se ha presentado e ilustrado el funcionamiento de un nuevo algoritmo basado en vectores soporte con el objetivo posterior de ser utilizado durante el esquema de descomposición de multiclasificación, denominado algoritmo $K-SVCR$. El problema implícito motivador de la presente formulación ha sido la mala disposición de las clases (Categorización) cuando se utilizan las metodologías estándares de una contra una o una contra el resto de clases, la poca eficiencia y complejidad de los algoritmos de otros tipos de máquinas de aprendizaje.

Al tratarse de un problema de clasificación multiclase, toma sentido construir máquinas de aprendizaje que asignen salida $+1 \text{ o } -1$ si el patrón de entrenamiento pertenece a las clases a ser separadas, y salida *0* si por el contrario tiene una etiqueta diferente a las anteriores, forzando al hiperplano separador de las dos clases implicadas a recubrir todos los patrones de entrenamiento pertenecientes a cualquier otra clase que no sean las dos especificadas. El desarrollo abarca los problemas mas conocidos ha dar solución con las *SVM*, se

[*] Tratando al máximo de no profundizar en la matemática tan compleja que las envuelve, dando referencias del desarrollo de estas para tal fin.

cita su arquitectura básica y se describe brevemente los algoritmos y métodos de aprendizaje.

Se estudiaron las definiciones de *SVM* para clasificación y regresión y se realizo una descripción profunda de las *SVM* Mixtas, teniendo en cuenta las principalmente conocidas correspondientes a la *L-SVCR* y al algoritmo *K-SVCR* este ultimo se desarrollo desde los motivos de su surgimiento, la descripción de su funcionamiento, ejemplos y comparación con otras máquinas de aprendizaje como son las redes neuronales y los algoritmos genéticos.

Para relacionar mas fuertemente trabajo teórico con la realidad de estas máquinas se introducen las aplicaciones realizando descripciones de algunas de las más conocidas y se citan numerosos programas concluyendo con descripción del funcionamiento y ejecución del *winSVM*, descripción de las características del *SVM*^{light} y SVMdark.

1 MARCO TEORICO

"La *SVM* fue ideada originalmente para la resolución de problemas de clasificación binarios en los que las clases eran linealmente separables *(Vapnik y Lerner, 1965)*. Por este motivo se conocía también como «hiperplano óptimo de decisión» ya que la solución proporcionada es aquella en la que se clasifican correctamente todas las muestras disponibles, colocando el hiperplano de separación lo más lejos posible de todas ellas. Las muestras más próximas al hiperplano óptimo de separación son conocidas como muestras críticas o «vectores soporte», que es lo que da nombre a la *SVM*"[1].

Aunque se dice que el estudio de las *SVMs* empezó a finales de los años setenta *(Vapnik, 1979)*, sólo está recibiendo ahora la atención y el desarrollo pertinente a todas sus posibilidades.

La máquina de soporte vectorial tiene numerosas aplicaciones, entre las cuales tenemos: reconocimiento de dígitos escritos a mano *(Cortés y Vapnik, 1995; Schölkopf, Burges y Vapnik, 1995; Schölkopf, Burges y Vapnik, 1996; Burges y Schölkopf, 1997)*, reconocimiento de objetos *(Blanz et al., 1996)*, identificación de voz *(Schmidt, 1996)*, reconocimiento imagen facial *(Osuna, Freund y 122 Burges Girosi, 1997a)*, y categorización del texto *(Joachims, 1997)*. Para la estimación de

[1] Sánchez R. (12 de Julio, 2004). Sistemas de Reconocimiento Facial. faq-mac. p.1. Obtenido de la red mundial el 16 Abril del 2005 : http:// www_faq-mac_com Sistemas de Reconocimiento Facial, por Raúl .htm

la regresión del evento, se han comparado *SVMs* en la referencia tiempo, serie, predicción y prueba. *(Müller et al. 1997; Mukherjee, Osuna y Girosi, 1997)*.

En la mayoría de estos casos, el rendimiento de la generalización en la *SVM*[*] coinciden con otros o son significativamente mejores que estos métodos de competencia.

Con respecto a las extensiones, las *SVMs* básicas no contienen ningún conocimiento anterior del problema[**], "(un acto de vandalismo[***] que saldría en la mejor actuación en las severamente desaventajadas redes neuronales)"[2] aunque ya se han terminado muchos trabajos incorporando el conocimiento anterior en las *SVMs (Schölkopf, y Vapnik de Burges, 1996; Schölkopf et al, 1998a; Burges, 1998)*. Aunque las *SVMs* tienen buen desempeño de la generalización, ellas, puede ser considerablemente lentas en la fase de la prueba, un problema se dirigió en *(Burges, 1996, Osuna y Girosi, 1998)*. El Reciente trabajo ha generalizado las ideas básicas *(Smola, Schölkopf y Müller, 1998a; Smola y Schölkopf, 1998)*, y conexiones mostradas a la teoría de regularización *(Smola, y Müller de Schölkopf, 1998b; Girosi, 1998; Wahba, 1998)*, y mostrado cómo las ideas de SVM pueden ser incorporadas en una amplia gama de otros algoritmos *(Schölkopf, Smola y Müller, 1998b; El Schölkopf et al, 1998c)*.

---

[*] Es decir, la rata de error sobre los conjuntos de la prueba

[**] por ejemplo, la mayoría de los casos de *SVMs* para el problema del reconocimiento de imagen daría los mismos resultados si los píxeles fueran permutados primero al azar con cada imagen que sufre la misma permutación.

[***] Extralimitado

[2] Burgues C. (2 de diciembre, 1998). A Tutorial on Support Vector Machines for Pattern Recognition. aya. technion. p.2. Obtenido de la red mundial el 17 Abril del 2005 : http://aya.technion.ac.il/karniel/CMCC/SVM-tutorial.pdf

El prejuicio del intercambio variante de las barreras (Bias) *(Geman y Bienenstock, 1992)*, capacidad de mando *(Guyon et al, 1992)*, demasiado ajustado *(Montgomery y Peck, 1992)* pero la idea básica es la misma. Hablando aproximadamente, para una tarea de aprendizaje dada, con una cantidad finita dada de entrenamiento de datos, el mejor desempeño de la generalización se logrará si en verdad el equilibrio es contundente entre la exactitud y el alcance en ese conjunto de entrenamiento, y "la capacidad" de la máquina, eso es, la habilidad de la máquina de aprender cualquier conjunto de entrenamiento sin error.

Una máquina con demasiada capacidad seria como un botánico con una memoria fotográfica que, al momento de presentar un nuevo árbol concluye que este no es un árbol por poseer un numero diferente de hojas, una máquina con muy poco capacidad está como el hermano perezoso del botánico, quién declara que si es verde, es un árbol. Ni puede generalizar bien. La exploración y la formalización de estos conceptos ha producido uno de las crestas brillantes de la teoría de el aprendizaje estadístico *(Vapnik, 1979)*.

2 INTRODUCCIÓN A LAS MÁQUINAS DE SOPORTE VECTORIAL

Las máquinas de soporte Vectorial *SVM, (Vapnik, 1995)* es uno de los temas mas desarrollados de máquinas de Aprendizaje de la última década. Algunas razones que explican este logro son sus buenas propiedades teóricas en la generalización y convergencia *(Cristianini & Shawe-Taylor, 2000)*. Otra razón es su excelente desempeño en algunos problemas difíciles un ejemplo se puede ver en *(Osuna et al., 1997; Dumais et al., 1998)*.

Al contrario de la estadística convencional y los métodos de la red neuronal, la *SVM* no intenta aproximar la complejidad del modelo de control guardando el número de rasgos pequeño.

Aunque están usándose ***SVMs*** principalmente para las tareas de clasificación, ellas también pueden usarse para aproximar las funciones[*]. Un problema que impide un uso más amplio de ***SVMs*** para la aproximación de la función es, que a pesar de sus buenos acercamientos teóricos, estos no son aplicables en línea, es decir, en casos dónde los datos se obtienen secuencialmente y el aprendizaje tiene que hacerse desde los primeros datos.

[*] Lo que se llama la regresión de ***SVM***

2.1 Probabilidades en las SVMs

La Máquina de Vectores Soporte* estándar no proporciona probabilidades en el sentido de estimar la probabilidad de acertar en las predicciones, es decir, de estimar la distribución condicional $P[Y/X=x]$ con objeto de cuantificar la incertidumbre asociada a una predicción. Por ello, dentro de las *SVMs* se han elaborado diferentes aproximaciones a este problema.

2.2 Problemas más comunes en las SVM

2.2.1 Estimación de Regresiones

Se entenderá por *variable numérica* aquella cuyo valor pertenece a un conjunto sobre el que ha sido definida una relación de orden total, es decir, la relación de orden permite definir una distancia. *Se define el* problema general de estimación de una regresión como el *PGAE en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un subconjunto de la recta real, y* $y \in Y \subseteq \Re$.

2.2.2 Reconocimiento de Patrones

Una *variable categórica* es aquella cuyo valor pertenece a un conjunto finito sobre el que no ha sido definida una relación de orden. Ejemplos de variables categóricas son aquellas que toman valor sobre conjuntos de elementos no numéricos — colores, marcas, tipos,... —.

Se define el problema general de clasificación o de reconocimiento de patrones como el *PGAE* en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un conjunto cuyos elementos no están ordenados,

* Máquina de Soporte Vectorial *SVMs*

$y \in Y = \{q_1, \ldots, q_K\}$ y el problema de clasificación binaria como el problema general de clasificación en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea el conjunto de etiquetas $Y = \{1,0\}$.

2.2.3 Regresión Ordinal

Una *variable* ordinal es aquella cuyo valor pertenece a un conjunto finito de etiquetas la cuales poseen un orden o sobre el que ha sido definida una relación de orden. *Se* define *el* problema general de regresión ordinal o de ordenación *como el PGAE en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un conjunto finito cuyos elementos poseen una ordenación*, $Y = \{q_1, \ldots, q_K\}$ con $q_K \succ y\ q_{K-1} \succ y \ldots \succ y\ q_1$.

Se entenderá que la ordenación sobre las salidas permite establecer una ordenación sobre las entradas que sea de utilidad. Como en el caso del reconocimiento de patrones, lo más habitual cuando se trata con problemas de regresión ordinal es transformarlos en problemas de clasificación binaria, teniendo en cuenta durante la transformación la ordenación que poseen las etiquetas[*].

[*] Para obtener ampliación de los temas correspondientes a problemas mas comunes en las *SVM buscar en la referencia* 02capitulo1.pdf. P. 9

2.3 Arquitectura SVM

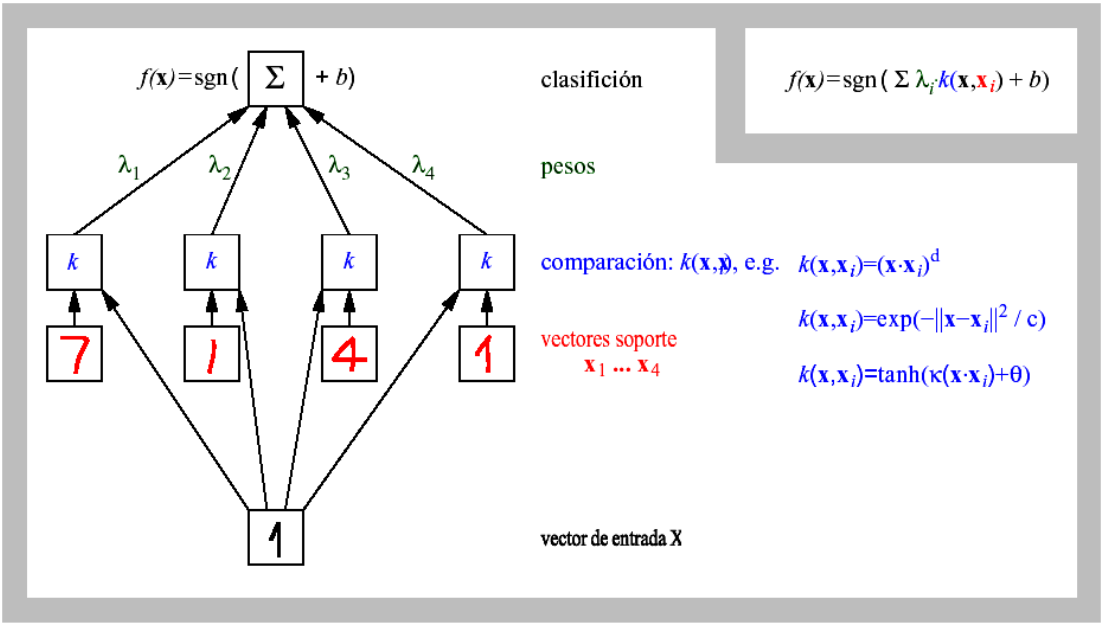


Figura 1. Arquitectura de una SVM

La arquitectura *SVM* es muy similar al las ya conocidas redes neuronales y cada una de las definiciones serán desarrolladas en el contenido de la monografía.

2.4 Algoritmo de aprendizaje

2.4.1 Definición de Algoritmo de Aprendizaje

Un algoritmo de aprendizaje será aquel proceso capaz de dar respuesta al problema de aprendizaje a partir de ejemplos planteados. Continuando la definición de este problema, se sucede la siguiente definición, en congruencia con aquella planteada por *[Vapnik, 1998]* y *[Cherkassky and Mulier, 1998].* Se define como aquel proceso capaz de elegir una única función a partir del conjunto de entrenamiento dando respuesta al problema planteado.

2.4.2 Espacio de Aproximación LM

La amplitud de un espacio de aproximación se define como la capacidad para aproximar cualquier función continua, $f \in C(X,Y)$, con una precisión especificada cualquiera. Se dirá que un espacio de aproximación LM es denso, en $C(X,Y)$, si cumple la propiedad de aproximación universal.

2.4.3 Conocimiento a priori

La necesidad de asumir cierto conocimiento de antemano sobre la forma del modelo buscado es esencial para conseguir la unicidad de la solución. Tal conocimiento será insertado en el algoritmo de aprendizaje en función del principio inductivo seleccionado. Algunos de los requerimientos más comunes suelen ser la exigencia de suavidad en la solución, la reducción en la talla de los pesos, la existencia de una determinada función de distribución de probabilidad conocida y la maximización del margen entre clases.

2.5 Principios inductivos

2.5.1 Minimización del Riesgo Empírico

El principio inductivo de minimización del riesgo empírico — *ERM*, Empirical Risk Minimization — es el más comúnmente utilizado en los procesos de aprendizaje clásico. El principio *ERM* fija su atención en redefinir el funcional de riesgo basándose en el siguiente razonamiento:

"Para obtener una buena generalización es suficiente con elegir los parámetros de la función aproximadora que aseguren el número mínimo de errores sobre el conjunto de entrenamiento"[3].

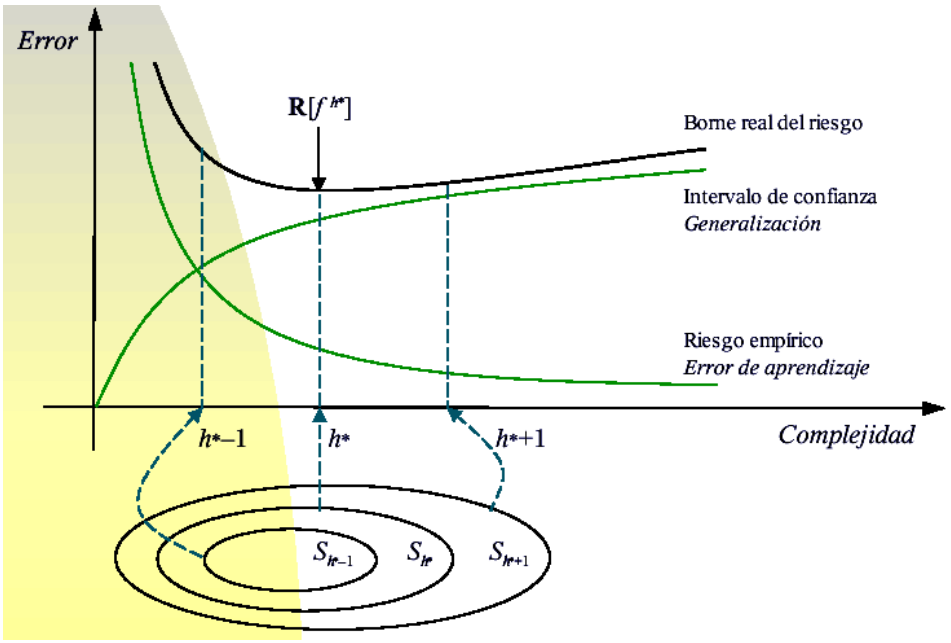


Figura 2. Muestra los limites del borne real de riesgo, el intervalo de confianza y el riesgo empírico

2.5.2 Minimización del Riesgo Estructural

El principio inductivo de Minimización del Riesgo Estructural — *SRM*, Structural Risk Minimization — es un proceso de inferencia desarrollado sobre la Teoría del Aprendizaje Estadístico — *SLT*, Statistical Learning Theory *[Vapnik,1998]*— específicamente para trabajar con problemas de aprendizaje a partir de un conjunto de entrenamiento pequeño.

[3] Angulo C.(Abril, 2001) aprendizaje con máquinas núcleo en entornos de multiclasificación. tdx.cesca. p. 10 Obtenido de la red mundial el 25 Abril del 2005: http:// www.tdx.cesca.es/tesis_upc /available/tdx -0628101-41150/02capitulo1.pdf

2.5.3 Inferencia Bayesiana

La inferencia bayesiana codifica información *a priori* adicional sobre las funciones de aproximación en forma de una *distribución de probabilidad a priori*, la probabilidad de que una función del espacio LM sea la auténtica función desconocida.

2.5.4 Regularización o Penalización

El proceso de inducción de un modelo general a partir de un conjunto de entrenamiento es un problema mal situado en el sentido que no existe una única solución. La técnica de regularización *[Tikhonov and Arsenin, 1977]* asegura, bajo ciertas pequeñas restricciones sobre los espacios de trabajo, que si en vez del funcional de riesgo *R(?)* se minimiza el denominado funcional de riesgo regularizado.*

2.6 Métodos de aprendizaje

Un método de aprendizaje resulta de la implementación constructiva del principio inductivo elegido en el algoritmo de aprendizaje. Generalmente corresponde a un proceso de optimización de un cierto funcional de riesgo que ha sido determinado siguiendo el principio inductivo. Para cada principio inductivo existen muchos métodos de aprendizaje que lo implementan correspondientes a los diferentes espacios de aproximación LM y a las diferentes técnicas de optimización.

* Para obtener ampliación de los temas correspondientes a problemas mas comunes en las *SVM,* buscar en la referencia 02capitulo1.pdf. P. 16

2.7 La Dimensión VC

La dimensión *VC* es una propiedad de un conjunto de funciones $\{f(a)\}^*$, y puede definirse para varias clases de función *f*. Consideraremos sólo funciones que corresponden a casos de reconocimiento de patrones de dos clases, para que $f(x,a)\in\{-1,1\}\ \forall x,a$. Ahora si un conjunto dado de puntos l puede etiquetarse de las 2^l maneras posibles, y para cada etiquetado, un miembro fijo de $\{f(a)\}$ puede encontrarse qué se asigne correctamente estas etiquetas, decimos que ese conjunto de puntos está siendo eliminado por ese conjunto de funciones. La dimensión *VC* para el conjunto de las funciones de $\{f(a)\}$ se define como el número máximo de puntos de entrenamientos que pueden romper por el $\{f(a)\}$. Note que, si la dimensión *VC* es *h*, entonces allí existe la menor cantidad de puntos de *h* que por lo menos pueden eliminarse, pero en general no será verdad que cada conjunto de puntos de h puede eliminarse.

2.7.1 Eliminación de los Puntos con Hiperplanos Orientado en R^n

Suponga que el espacio en que se encuentran los datos es $\boldsymbol{R}^2$, y el $\{f(a)\}$ fijo consiste en líneas rectas orientadas, para que a una línea dada se asignen todos los puntos en un lado, la clase *1*, y todos los puntos en el otro lado, la clase *-1.* La orientación se muestra en la *Figura 2* por una flecha, especificando en que el lado de la línea los puntos tendrán asignados la etiqueta *1*. Mientras es posible encontrar tres puntos que pueden eliminarse por este conjunto de funciones, no es

* Utilizaremos *a* como un conjunto genérico de parámetros: una opción de a que especifica una función particular

posible encontrar cuatro. Así la dimensión *VC* del conjunto de líneas orientadas en $\boldsymbol{R}^2$ es tres.

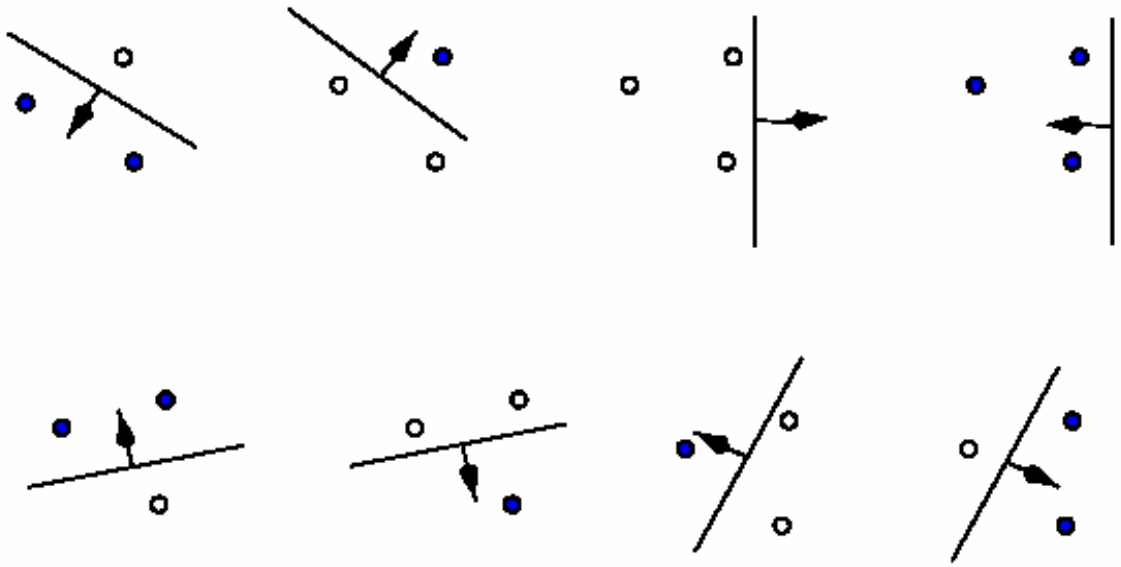


Figura 3. Tres puntos en R^2, estrelló por las líneas orientadas.

La dimensión *VC* del conjunto de hiperplanos orientado en $\boldsymbol{R}^n$ es $n+1$, desde que siempre podemos escoger $n+1$ puntos, y entonces escogemos uno de los puntos como el origen, tal que los vectores de la posición de los puntos de *n* restantes son linealmente independientes, pero nunca puede escoger $n+2$ puntos semejantes (desde que ningún $n+1$ vectores en $\boldsymbol{R}^n$ pueden ser linealmente independientes). Una prueba alternativa de esta conclusión puede encontrarse en *(Anthony y Biggs, 1995),* y sus referencias.

2.7.2 La Dimensión VC y el Número de Parámetros

La dimensión *VC* da en concreto la noción de la capacidad de un conjunto dado de funciones. Intuitivamente, podría dirigir anticipadamente las máquinas de aprendizaje y teniendo muchos parámetros tendríamos una dimensión *VC* alta, mientras máquinas de aprendizaje con pocos parámetros tendrían dimensión *VC* baja. En *E. Levin y J.S. Denker (Vapnik, 1995)* muestran un ejemplo notable de esto: una máquina de aprendizaje con sólo un parámetro, pero con la dimensión de *VC* infinita[*].

2.7.3 Minimizando el Límite por Minimización h

La *figura 4* muestras cómo el segundo término de la derecha de la ecuación de riesgo seguro *(1)*[**] varía con *h*, dando una opción del *95%* de nivel de seguridad *(?= 0:05)* y asumiendo una muestra de entrenamiento de tamaño *10.000.* La seguridad de *VC* es monótona en la función creciente de *h*. Esto será verdad para cualquier valor de l.

$$R(a) \le R_{emp}(a) + \sqrt{\left(\frac{h\left(\log\left(2l/h\right)+1\right) - \log\left(h/4\right)}{l}\right)} \quad (1)$$

Así, dado alguna selección de máquinas de aprendizaje cuyo riesgo empírico es cero, se escogería esa máquina de aprendizaje cuyo asoció de conjunto de funciones tiene la dimensión *VC* mínima. Esto llevará a un límite superior bueno en el error real. En general, para que el riesgo empírico diferente cero, se

[*] Se dice que una familia de clasificadores tiene la dimensión *VC* infinita si puede destruir *l* puntos, no importa lo grande de *l*.

[**] Esta ecuación se encuentra desarrollada y justificada en SVM-tutorial.pdf cuyo titulo es A Tutorial on Support Vector Machines for Pattern Recognition su localizacion se encuentra en la cita No. 4

escogería esa máquina de aprendizaje que minimiza el lado derecho de la ecuación *(1).*

"Note que se esta adoptando esta estrategia, usando sólo la ecuación *(1)* como una guía. La ecuación *(1)* da (con un poco de probabilidad escogida) un límite superior en el riesgo real. Esto no lo previene una máquina diferente con el mismo valor para el riesgo empírico, y de quien el conjunto de la función tiene la dimensión *VC* superior, debe tener un buen rendimiento. De hecho un ejemplo de un sistema que da un buen rendimiento a pesar de tener la dimensión *VC* infinita se mostrara a continuación. También note que las muestras del gráfico para $h/l > 0.37$ (y para $h = 0.05$ y $l = 10{,}000$), la dimensión *VC* excede la unidad, y para que para los valores más altos el límite no se garantiza seguro"[4].

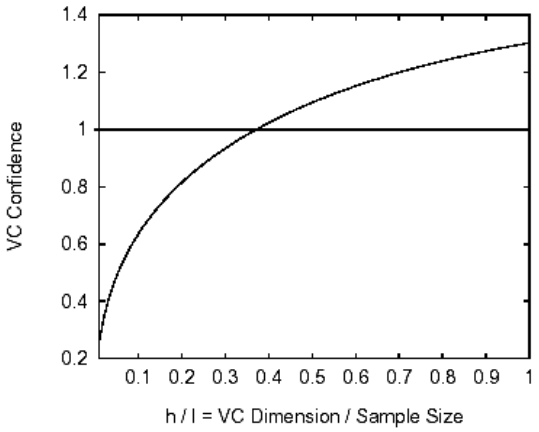


Figura 4. La confianza VC es el monótono en h

[4] Burgues C. (2 de diciembre, 1998). A Tutorial on Support Vector Machines for Pattern Recognition. aya. technion. p.6. Obtenido de la red mundial el 17 Abril del 2005 : http://aya.technion.ac.il/karniel/CMCC/SVM-tutorial.pdf

Ejemplos 1 "Considere clasificar los *k* más cercano, a $k=1$ Este conjunto de funciones tiene la dimensión *VC* infinita y un riesgo empírico de cero, desde cualquier número de puntos, etiquetó arbitrariamente, aprenderá con éxito por el algoritmo*. Así el límite no proporciona la información. De hecho, para cualquier clasificación con la dimensión *VC* infinita, ningún límite es valido. Sin embargo, aunque el límite no es válido, clasifica el más cercano a pesar de eso pueden desempeñarse bien. Así este primer ejemplo es de propósito preventivo: infinita "capacidad" no garantiza un bajo desempeño"[5].

Ejemplo 2. "Un clasificador para el límite que esta establecido, pero qué viola el límite. Queremos el lado izquierdo de la ecuación *(1)* para hacerlo tan grande como sea posible, y el lado derecho para hacerlo tan pequeño como sea posible. Así que queremos una familia de clasificación que den el peor riesgo real posible *0.5*, se observan en algunos entrenamientos riesgo empírico de cero, y cuya dimensión *VC* es fácil computar y es menor que l (para cuando el límite sea trivial). Un ejemplo es el siguiente llamado "el clasificador del cuaderno." Este clasificador consiste en un cuaderno con bastante capacidad para apuntar a bajo las clases de m entrenamientos de observaciones, dónde $m \leq 1$. Para todos los parámetros subsecuentes, el clasificador dice simplemente que todos los modelos tienen la misma clase. También suponga que los datos tienen tanto el positivo $(y=+1)$ como el negativo $(y=-1)$ ejemplos, y las muestras son escogidas al

* Con tal de que ningunos dos puntos de clasificación opuestos a la derecha quede encima del otro
[5] Ibid., p.7

azar. El clasificador del cuaderno tendrá en cero el riesgo empírico para las observaciones de *m; 0.5* de error de entrenamiento para todas las observaciones subsecuentes; *0.5* de error real, y la dimensión *VC h = m*. Sustituyendo estos valores en la ecuación *(1),* el límite se vuelve:"[6]

$$\frac{m}{4l} \le \ln\left(2l/m\right)+1-\left(1/m\right)\ln\left(h/4\right) \qquad (2)$$

Qué se reúne ciertamente para todo *?* si

$$f(z)=\left(\frac{z}{2}\right)\exp^{\left(z/4-1\right)} \le 1, \quad z \equiv \left(m/l\right), \quad 0 \le z \le 1 \qquad (3)$$

Cual es verdad, desde *f(z)* incremento monótono, *y* $f\left(z=1\right)=0.236$.

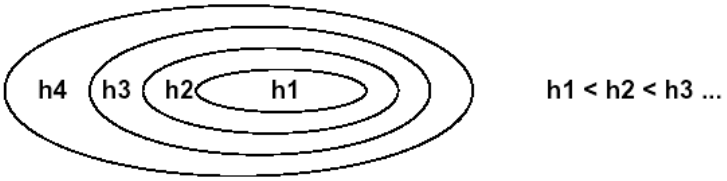


Figura 5. Anidó los subconjuntos de funciones, ordenados por la dimensión de VC.

[6] Ibid., p.7

3 LAS MÁQUINAS DE SOPORTE VECTORIAL

3.1 Las máquinas de soporte vectorial lineales

3.1.1 El Caso Separable

"Empezaremos con el caso más simple: las máquinas lineales entrenando en los datos separables*. La etiqueta nueva de entrenamiento de datos $\{x, y\}; i = 1, ..., l, y_i \in \{-1, 1\}, x_i \in R^d$. Ahora, suponga que tenemos algún hiperplano que separa los positivos de los ejemplos negativos ("hiperplano de separación"). Los puntos *x* que quedan en el hiperplano satisfacen $w \cdot x + b = 0$, dónde *w* es normal al hiperplano, $|b|/\|w\|$ es la distancia perpendicular del hiperplano al origen, y $\|w\|$ es la norma Euclideana de *w*. Sea d_+ y d_- la distancia más corta permitida del hiperplano de separación a las muestras positivas y negativas mas cercanas respectivamente. Ejemplo. Defina "Margen" de un hiperplano de separación como la suma $d_+ + d_-$ para el caso linealmente separable, el algoritmo de vector soporte busca el hiperplano de separación simplemente con el margen más grande. Esto puede formularse como sigue: suponga que todos los datos de entrenamiento satisfacen las condiciones siguientes:"[7]

* para el caso general las máquina no lineales entrenan en datos no separables, los resultados en un problema de programación cuadrática son muy similares

[7] Ibid., p.7

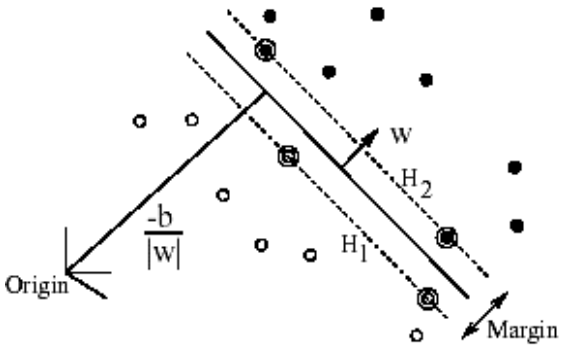


Figura 6. El hiperplanos de separación Lineal para el caso separable. Los vectores de soportes se rodean.

$$x_i \cdot w + \geq +1 \quad para \quad y_i = +1 \quad (4)$$

$$x_i \cdot w + \geq +1 \quad para \quad y_i = -1 \quad (5)$$

Éstos pueden combinarse dentro de un conjunto de desigualdades:

$$y_i\left(x_i \cdot w + b\right) - 1 \geq 0 \quad \forall i \quad (6)$$

Ahora considere los puntos para que la igualdad en la ecuación *(4)* se mantenga[*]. Estos puntos quedan delante del hiperplano $H_1 : x_i \cdot w + b = -1$ con normal w y distancia perpendicular al origen $|b|/\|w\|$. Similarmente, los puntos para que la igualdad en la ecuación *(5)* mantenga el hiperplano de decisión $H_2 = x_i \cdot w + b = -1$, nuevamente con la normal *w*, y la distancia perpendicular de el origen $|b|/\|w\|$ por lo tanto $d_+ = d_- = 1/\|w\|$ y el margen simplemente es $2/\|w\|$. Note que H_1 y H_2 son paralelos (ellos tienen la misma normal) y que ningún punto de entrenamiento

[*] Requiriendo que exista un punto semejante que sea equivalente al escoger una balanza para *w* y *b*

caiga entre ellos. Así nosotros podemos encontrar el par de hiperplanos que da el máximo margen minimizando $\|w\|^2$, sujeto a restricciones *(6)*.

Esperamos la solución para un típico caso de dos dimensiones que tiene la forma que se muestra en la *Figura 5.* Este entrenamiento apunta a sostener la igualdad en la ecuación *(6)* (es decir que aquellos términos que quedan por encima de los hiperplanos H_1, H_2), y es quien haría cambiar la solución encontrada, estos son llamados vectores soporte; indicados en la *Figura 6* por los círculos extras. Ahora se cambiara a una formulación Lagrangiana del problema. Hay dos razones para hacer esto. La primera es mantener la ecuación *(6);* se reemplazará por las restricciones en los multiplicadores de Lagrange que serán mucho más fácil de manejar. La segunda es la reformulación del problema, los datos de entrenamiento sólo aparecerán (en el entrenamiento real y algoritmos de la prueba) en la formula el producto punto entre los vectores. Ésta es una propiedad crucial que nos permitirá generalizar el procedimiento del caso no lineal.

Así, se presentara los multiplicadores de Lagrange positivos; $a_i, i = 1,.....,l,$ uno para cada una de las restricciones de desigualdad *(6).* Recordando que la regla para las restricciones del C_i de la formula $C_i \geq 0,$ las ecuaciones de restricciones son multiplicadas por los multiplicadores de Lagrange positivos y deducidos de la función objetiva, para formar el Lagrangiano. Para las restricciones de igualdad, los multiplicadores de Lagrange son espontáneos. El Lagrangiano queda:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} a_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} a_i \quad (7)$$

Debemos minimizar L_P ahora con respecto a *w, b,* y simultáneamente requiere que el derivado de L_P con respecto a todo los a_i desaparezcan, todos sujeto a las restricciones $a_i \geq 0$, llamemos este particular puesto de restricciones C_1. Ahora éste es un problema de la programación cuadrática convexa, desde que el objetivo de la función es de convexación, y esos puntos que también satisfacen las restricciones forman un conjunto convexo*. Esto significa que podemos resolver al siguiente equivalentemente problema "dual": Aumente L_P al máximo, sujeto a las restricciones la pendiente de L_P con respecto a *w* y *b* desaparece, y también sujeta a las restricciones que el $a_i \geq 0$ (llamemos a ese conjunto particular de restricciones C_2). Esta formulación dual particular del problema se llama el *Wolfe dual (Fletcher, 1987).* Tiene la propiedad que el máximo de L_P, esta sujeto a las restricciones C_2, ocurre a los mismos valores del *w, b* y *a*, como el mínimo de L_P, sujeto a los restricciones C_1. Requiriendo que la pendiente de L_P con respecto a *w* y *b* desaparece dé las condiciones:

$$w = \sum_i a_i y_i x_i \quad (8)$$

$$\sum_i a_i y_i = 0 \quad (9)$$

Desde que estén los requisitos de igualdad en la formulación dual, se podrá sustituirlos en la ecuación *(7)* para obtener

* Cualquier restricción lineal define un conjunto convexo, y un conjunto de *n* restricciones lineales simultáneas definen la intersección de *n* conjuntos convexos que también son un conjunto convexo.

$$L_D = \sum_i a_i - \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j X_i \cdot X_j \quad (10)$$

Note que ahora se ha dado unas etiquetas diferentes al Lagrangiano (*P* para principal, *D* para dual) se enfatiza que las dos formulaciones son diferentes: L_P y L_D se levantan con el mismo objetivo pero con las restricciones diferentes; y la solución se encuentra minimizando L_P o aumentando al máximo L_D. También note que si formulamos el problema con $b = 0$ que las cantidades a requerir en todos los hiperplanos contienen el origen, la restricción *(9)* no aparezca. Ésta es una restricción apacible para los altos espacios dimensiónales, desde que la cantidad a reducir del número de grados de libertad por uno.

Entrenando el vector soporte*, por consiguiente las cantidades al aumentar al máximo L_D con respecto al a_i, sujeto a las restricciones *(9)* y posibilidades de a_i, con la solución dada por *(8)*. Nos damos cuenta que hay un multiplicadores de Lagrange a_i para cada punto de entrenamiento. En la solución, esos puntos para que $a_i \geq 0$ se llaman "vector soporte", y quedan sobre uno de los hiperplanos H_1, H_2. Todos los otros puntos de entrenamiento tienen $a_i \geq 0$ y queda en H_1 o H_2**, o en ese lado de H_1 o H_2 tal que la desigualdad de la ecuación *(6)* se mantenga exacta. Para estas máquinas, los vectores soportes son los elementos críticos de los conjuntos de entrenamiento. Ellos quedan más cerca al límite de decisión; si

* Para el caso lineal separable

** Tal que la igualdad en la ecuación *(6)* se mantiene

todos los otros puntos de entrenamiento estaban alejados[*], y el entrenamiento fue repetido, los mismos hiperplanos de separación se encontrarían.

3.1.2 Las Condiciones de Karush-Kuhn-Tucker

Las condiciones Karush-Kuhn-Tucker *(KKT)* juegan un papel importante en la teoría y práctica de optimización de la restricción. Para el problema original sobre, las condiciones de *KKT* se pueden encontrar en *(Fletcher, 1987*):

3.1.3 La Fase de la prueba

Una vez que se entrena la Máquina de soporte Vectorial, ¿cómo puede usarse? Simplemente se determina en que lado estará el límite de decisión[**] dado un patrón x mentiras se asigna la etiqueta de la clase correspondiente, es decir se toma la clase de x para ser el *sgn (w·x + b).*

3.1.4 El Caso No separable

"El algoritmo anterior es para los datos separables, cuando se aplica a los datos no separables, no encontrará una solución factible: esto se evidenciará por la función objetiva[***] creciendo arbitrariamente grande. Así ¿cómo se pueden extender estas ideas para ocuparse de datos no separables? Se disminuirá las restricciones *(4)* y *(5),* pero sólo cuando sea necesario, es decir, introduciremos un costo extenso[****]. ¿Para que hacer eso? Esto puede hacerse introduciendo

[*] O moviéndose alrededor, pero para no cruzar H_1 o H_2

[**] Mentiras estas quedan en la mitad del hiperplano entre H_1 y H_2 y paralelas a ellos

[***] Es decir el Lagrangiano dual

[****] Es decir un aumento en la función objetiva original

variables positivas lentamente $x_i, i = 1,....,l$ en las restricciones *(Cortés y Vapnik, 1995)*, qué entonces se vuelven:"[8]

$$x_i \cdot w + b \geq +1 - x_i \quad para \quad y_j = +1 \quad (11)$$

$$x_i \cdot w + b \leq -1 - x_i \quad para \quad y_j = -1 \quad (12)$$

$$xi \geq 0 \forall i \quad (13)$$

Así, para cuando ocurra un error, el correspondiente *?i* debe exceder la unidad, entonces $\sum i x_i$ es un límite superior sobre el número de errores de entrenamiento. Una manera natural de asignar un costo extra para los errores, es cambiar la función objetiva a ser minimizada $\|w\|^2 / 2$ *a* $\|w\|^2 / 2 + C\left(\sum_i x_i\right)^k$ dónde *C* es un parámetro a ser escogido por el usuario, con un *C* más grande corresponde a asignar una multa más alta a los errores. Como él está en la posición, éste es un problema de la programación convexo para cualquier entero positivo *k,* para $k = 2$ y $k = 1$ también es un problema de la programación cuadrático, y se elige *k = 1* tiene la ventaja extensa que ni el *?i*, ni sus multiplicadores de Lagrange, aparezca en el problema dual *Wolfe*: Aumente al máximo:

$$L_D = \sum_i a_i - \frac{1}{2} \sum_{i,j} a_i a_j y_i y_j X_i \cdot X_j \quad (14)$$

Sujeto a:

$$0 \leq a_i \leq C \quad (15)$$

$$\sum_i a_i y_i = 0 \quad (16)$$

[8] Ibid., p.14

La solución se da de nuevo por

$$\sum_{i=1}^{Ns} a_i y_i x_i \quad (17)$$

Donde N_S es el número de vectores soportes. Así la única diferencia del caso óptimo del hiperplano es que los a_i tienen un límite superior de C. que La situación se resume esquemáticamente en *Figura 7*. Se necesita las condiciones de *Karush-Kuhn-Tucker* para el problema original. El Lagrangian original es

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_i x_i - \sum_i a_i\{y_i(x_i \cdot w + b) - 1 + x_i\} - \sum_i m_i x_i \quad (18)$$

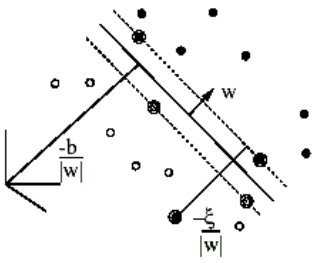


Figura 7. Los hiperplanos de separación Lineal para el caso no separable.

Donde los $?_i$ son los multiplicadores de Lagrange introducidos para dar fuerza positivamente a los $?_i$. La *KKT* condiciona el problema original por consiguiente (note que *i* se ejecutan desde *1* al número de los puntos entrenando, y *v* de *1* a la dimensión de los datos)

3.1.5 Una Analogía Mecánica

Considere el caso en que los datos están en $\boldsymbol{R}^2$. Suponga que los *i* vectores soporte ejercen una fuerza $F_i = a_i y_i \boldsymbol{w}$ en una lamina tiesa que queda a lo largo de la superficie de decisión (“lamina de decisión”)

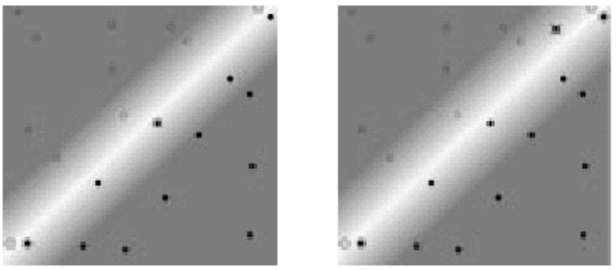


Figure 8. El caso lineal, separable (izquierdo) y no (el derecho). El fondo que distorsiona la forma de la muestra de decisión de la superficie.

(Se denota w como la unidad del vector en dirección w).

Entonces la solución *(17)* satisface las condiciones de equilibrio mecánico:

$$\sum Forces = \sum a_i y_i \hat{w} = 0 \quad (19)$$

$$\sum Forces = \sum s_i \wedge (a_i y_i \hat{w}) = \hat{w} \wedge w = 0 \quad (20)$$

(Aquí los s_i son vectores soporte, y ? denota el producto del vector.) Para los datos en $\boldsymbol{R}^n$, claramente la condición de la suma de las fuerzas halladas desaparece cuando se encuentran. Uno puede mostrar fácilmente que el torqué también desaparece.

Esta analogía mecánica sólo depende de la solución de la formula *(17)*, y por consiguiente se mantienen para ambos casos separables y no separables. De hecho lo que esta analogía contiene en general[*]. La analogía da énfasis en un punto interesante que es el "más importante" los puntos de datos son los vectores soporte con un alto valor de *a*, ya que ellos ejercen las fuerzas más grandes en la tabla de decisión. Para el caso no separable, $a_i \leq C$ el límite superior

[*] Es decir, también para el caso no lineal que se describió

corresponde a un límite superior en la fuerza que cualquier punto dado que se permita ejercer en la tabla. Esta analogía también proporciona una razón* para llamar estos vectores particulares "los vectores soportes."

La figura 9 muestra dos ejemplos de dos clases de problema sobre patrones de reconocimiento, uno separable y uno no. Las dos clases se denotan por los círculos y discos respectivamente. Se identifican los vectores soportes con un círculo extra. El error en el caso no separable se identifica con una cruz. Al lector se le invita a usar *Lucent SVM Applet (Burges, Knirsch y Haratsch, 1996)* experimentar y crear los cuadros como éstos.

3.2 Máquina de soporte vectorial no lineales

"¿Cómo los métodos anteriores pueden generalizarse al caso dónde la función de decisión no es función lineal de los datos? *(Boser, Guyon y Vapnik, 1992),* mostró que es un truco bastante viejo *(Aizerman, 1964)* puede usarse para llevar a cabo esto en una manera increíblemente directa. Primero la única manera en que los datos aparezcan en el problema de entrenamiento, ecuaciones *(14)* - *(16)*, está en producto punto de la formula, $x_i \cdot x_j$. Ahora suponga que primero asignamos los datos a algún otro (posiblemente el infinito dimensional) Euclidean espacio *H*, usando una asignación que nosotros llamaremos Φ:"[9]

$$\Phi : R^d \mapsto \mathrm{H}. \quad (21)$$

* Tan bueno como cualquier otro

[9] Ibid., p.17

Entonces claro el algoritmo de entrenamiento sólo dependería de los datos a través de los productos punto de *H*, es decir en las funciones de la forma $\Phi(x_i)\cdot\Phi(x_j)$. Ahora si había una "función Kernel" *K* tal que $K(x_i \cdot x_j) = \Phi(x_i)\cdot\Phi(x_j)$, se necesita usar *K* en el algoritmo de entrenamiento, y nunca se necesitaría igualar explícitamente sabiendo que es Φ.

Se llamara el espacio en que residen los datos, L. (de aquí en adelante usaremos L como un código nemotécnico para "dimensional bajo", y *H* para "dimensional alto": normalmente el caso en el que el rango Φ es de dimensión más alta que su dominio). Note que, además del hecho que *H* este en función *w*, no habrá ningún vector en general en L que asignar, en la tabla Φ. Si estuvo ahí, *f(x)* en la ecuación *(36)* podría computarse en un paso, mientras evita la suma (y haciendo el *SVM* N_S correspondiente en un tiempo más rápido, dónde N_S es el número de vectores soporte). No obstante, pueden usarse ideas a lo largo de estas líneas para acelerar la fase de prueba de las *SVMs* significativamente *(Burges, 1996)*. También Note que es fácil encontrar kernels Por ejemplo, kernels que son funciones de los productos punto de x_i y L tal que el algoritmo de entrenamiento y la solución son independiente de la dimensión de L y *H.*

La literatura en *SVMs* normalmente se refiere al espacio *H* como un espacio de *Hilbert*. Se puede pensar en un espacio de *Hilbert* como una generalización de espacio de Euclidean que se comporta en una forma moderada. Específicamente, es cualquier espacio lineal, con un producto interno definido que también está

completo con respecto a la norma correspondiente (es decir, cualquier sucesión de *Cauchy* de puntos convergentes a un punto en el espacio). Algunos autores (por ejemplo *(Kolmogorov, 1970))* también requiere que sea separable*, y algunos no lo hacen (por ejemplo *(Halmos, 1967)*). Es principalmente una generalización porque su producto interno puede ser cualquier producto interno, no sólo el escalar ("punto") el producto usado aquí y en los espacios de Euclidean en general es interesante.

La antigua literatura matemática (por ejemplo *(Kolmogorov, 1970)*) también requirió espacios *Hilbert* con dimensiones infinitas, y que matemáticos estaban bastante cerca al definir los espacios dimensiónales de Euclidean. Estudie sobre los espacios de *Hilbert* en los centros de operaciones, desde que las propiedades básicas tienen mucho tiempo desde que se funcionado. Puesto que algunas personas entienden claramente la definición de los espacios de *Hilbert*.

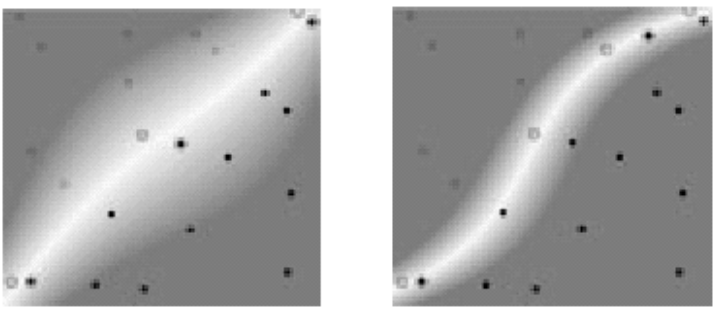


Figure 10. El polinomio de Grado 3 kernel. El color del fondo muestra la forma de la superficie de decisión.

* Es decir, debe tener un subconjunto contable cuyo cierre es el propio espacio

"La *Figura 10* muestra los resultados del problema de reconocimiento de patrones como se mostró en la *Figura 8*, pero dónde kernel fue escogido por ser un polinomio cúbico. Note que, aunque el número de grados de libertad es más alto, para el caso linealmente separable (tablero izquierdo), la solución es aproximadamente lineal, mientras indica que la capacidad está controlándose; y que el caso linealmente no separable (tablero derecho) llega a hacer separable. Finalmente, note que aunque los clasificadores de *SVM* descritos anteriormente son clasificadores binarios, ellos, se combina para ocuparse fácilmente del caso de las multiclases. Un simple entrenamiento de combinaciones eficaces de *N* uno contra el resto de clasificadores* para el caso del *N* clases y toma la clase para un punto de la prueba correspondiendo a la distancia positiva más grande *(Boser, Guyon y Vapnik, 1992).* La efectiva combinación de los *N* entrenamientos uno versus el resto de clasificadores para el caso del N"[10].

3.2.1 Las Soluciones globales y particulares

¿Cuándo esta la solución del vector soporte para entrenamiento de problema global, y cuándo es particular? Por "global", queremos decir que no existe ningún otro punto en la región factible a que la función objetiva toma el valor más bajo. Nos dirigiremos dos tipos de maneras en que la singularidad no puede sostener: las soluciones para las que $\{w,b\}$ son únicos, pero para la cual la expansión de *w* no es, ecuación *(17);* y cuya solución $\{w,b\}$ difiere. Ambos son interesantes: aun cuando el par $\{w,b\}$ es único, si los a_i no son, puede haber expansiones

* Es decir, "uno" positivo, "el resto" negativo

[10] Ibid., p.23

equivalentes de *w* que requiere menos vectores soporte (un ejemplo trivial de esto se da debajo), y qué por consiguiente requiere menos instrucciones durante la fase de prueba.

Resulta que cada solución local también es global. Ésta es una propiedad de cualquier problema de programación convexo *(Fletcher, 1987).* Además, la solución garantiza ser única si la función objetiva *(ecuación (14))* es estrictamente convexa, qué en nuestros caso significa que el Hessiano debe ser definido positivo (note que para la función objetiva cuadrática *F*, el Hessiano es definido positivo si y sólo si *F* es estrictamente convexo; esto no es valido para el cuadrático de *F*: allí, un Hessiano definido positivo implica una función objetiva estrictamente convexa, pero no viceversa (considere $F = x^4$ *(Fletcher, 1987)).* Sin embargo, aun cuando el Hessiano es el semidefinido positivo, la solución todavía puede ser única: considere dos puntos a lo largo del la línea real con las coordenadas $x_1 = 1$ y $x_2 = 2$, y con las polaridades + *y* -. Aquí el Hessiano es el semidefinido positivo, pero la solución ($w = -2; b = 3; x_i = 0$ en la ecuación *(11), (12), (13)*) es única. También es fácil de encontrar soluciones que no son únicas en el sentido de que las a_i en la expansión de w no son únicas. Por ejemplo, considere el problema de cuatro puntos separables en un cuadrado en $R_2: x_1 = [1,1], \quad x_2 = [-1,1], \quad x_3 = [-1,-1]$ y $x_4 = [1,-1]$ con las polaridades $[+,-,-,+]^*$ respectivamente. Una solución es $w = [1,0], b = 0, \quad a = [0.25, 0.25, 0.25, 0.25]$; otro

* Polaridades de las entradas

tiene el mismo *w* y *b*, pero $a = [0.5, 0.5, 0, 0]$; (Note que ambas soluciones satisfacen las restricciones $a_i > 0$ *y* $\sum_i a_i y_i = 0$). Cuando esto ocurra en general dando algunas soluciones a *a*, escoja un *a*' que está en el espacio nulo del Hessiano $H_{ij} = y_i y_j X_i \cdot X_j$ y requiere que ese *a*' sea ortogonal a todos los vectores cuyos componentes son *1*. Agregando *a*' a a en la ecuación *(14)* quedara L_D igual. Si $0 < a + a' \leq C$ y *?* satisface la ecuación *(9),* entonces $a + a'$ también es una solución. ¿De que modo las soluciones de $\{w, b\}$ no son únicas? (se hace énfasis a esto porque puede pasar en un principio si el Hessiano no es definido positivo, e incluso entonces, las soluciones son necesariamente globales). El siguiente teorema muy simple muestra que si ocurren soluciones no únicas, entonces la solución a un punto óptimo está deformada continuamente en la solución de otro punto óptimo, de tal manera que todos los puntos del intermedio también son las soluciones.

Teorema: Sea ***X*** una variable en representación de las variables $\{w, b\}$. Para el problema se permite que Hessiano sea semidefinido positivo, para que la función objetiva sea convexa. Sea X_0 y X_1 dos puntos donde la función objetiva logra su valor mínimo. Entonces allí existe una expresión $X = X(t) = (1-t)X_0 + tX_1, \quad t \in [0,1]$ tal que $X(t)$ es una solución para todo los t .

La prueba: dejamos el valor mínimo de la función objetiva en F_{min}. Entonces asumimos, $F(X_0) = F(X_1) = F\min$. Por la convexidad de *F,*

$F(X(t)) \le (1-t)F(X_0) + t\ F(X1) = F\min$. Además, por la linealidad, el $X(t)$ satisface las restricciones de la ecuación *(11), (12):* explícitamente (combinando ambas restricciones de nuevo en uno):

$$\begin{aligned} & y_i(w_t \cdot x_i + b_t) = y_i((1-t)(w_0 \cdot x_i + b_0) + t(w_1 \cdot x_i + b_1)) \\ & \ge (1-t)(1-x_i) + t(1-x_i) = 1-x_i \end{aligned} \quad (22)$$

Aunque simple, este teorema es bastante instructivo. Por ejemplo, uno podría pensar que el problema descrito en la *Figura 11* tiene varias soluciones diferentes que son óptimas (para el caso de las máquinas de soporte vectorial lineales). Sin embargo, dado que no se puede mover el hiperplano fácilmente se propone solucionarlo de otra forma sin generar hiperplanos que no son soluciones, sabemos que estas soluciones propuestas no son de hecho soluciones absolutas. De hecho, para cada uno de estos casos, la única solución óptima está en $w=0$, con una opción conveniente de *b* (qué tiene el efecto de asignar la misma etiqueta a todos los puntos). Note que esto da perfectamente una solución aceptable al problema de la clasificación a cualquiera hiperplano propuesto (con $w \neq 0$), la causa es que la función objetiva original tomara un valor más alto.

Figura 11. Dos problemas, con propósito (incorrecto) las soluciones no únicas.

Finalmente, note que el hecho que el entrenamiento de la *SVM* siempre encuentra una solución global está en las restricciones al caso de redes neuronales dónde muchos mínimos locales normalmente existen.

3.3 SVM para clasificación

Se desea construir un hiperplano que separe las dos clases, etiquetadas $y \in \{-1,+1\}$, de forma que la distancia entre el hiperplano óptimo y el patrón de entrenamiento más cercano — *margen* — sea máxima, con la intención de lograr la generalización de la máquina de aprendizaje *[Burges, 1998], [Smola, 1998], [Vapnik, 1995].* La expansión del método *SVMC* a funciones de decisión no lineales se realiza introduciendo el espacio de entrada $X \subseteq \Re^d$ en otro espacio de mayor dimensión *F*, denominado *espacio de características*, dotado de producto interno, introduciendo una guía no lineal, $i : X \subseteq \Re^d \to F_{\langle \cdot,\cdot \rangle}$ de forma que el hiperplano óptimo corresponde a un núcleo de Hilbert[2], permite definir como función decisión $h(x) = sing(f(x,w))$.

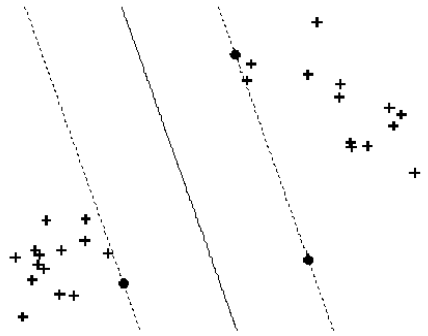


Figura 12. El margen es la distancia perpendicular entre el hiperplano separador y el hiperplano que pasa sobre los puntos más cercanos, los vectores soporte.

En resumen, el hiperplano óptimo en forma canónica del margen mínimo es hallado solucionando el problema de optimización restringida:

$$\arg\min\ R_{SVMC}(w,x) = \frac{1}{2}\|w\|_F^2 + C\sum_{i=1}^{l} x_i \quad (23)$$

3.3.1 Clasificación multiclase con máquinas de soporte vectorial

Es muy habitual cuando se trabaja con problemas de clasificación extraídos de la vida real encontrarse con situaciones de multiclasificación. Tradicionalmente, cuando se realiza el desarrollo teórico de una máquina de aprendizaje, si esta ha sido especialmente diseñada para casos binarios como las *SVMCs,* se soluciona la posibilidad de trabajo sobre un entorno multiclase afirmando que su generalización a tales problemas es "evidente". En otras ocasiones la máquina ya está concebida para el trabajo con múltiples salidas, como en el caso de los *MLPs*[*], pero el grueso del desarrollo teórico se reduce al caso binario por motivos de simplicidad de notación, pues la generalización a casos multiclase se hace "obvia". Por último, existe un tercer tipo de máquina de aprendizaje, como por ejemplo los árboles de decisión, que trabajan directamente con problemas multiclase aunque para ello necesiten de unos nodos de decisión que son, en la mayor parte de las ocasiones, dicotomías.

Un estudio detallado sobre su modo de funcionamiento mostrara tanto las ventajas de su implementación como las dificultades que evidencian cuando es probada su

[*] MLPs (Multicapa perceptrom)

robustez frente a fallos parciales de predicción, la interpretación que realizan de las respuestas, la identificación y/o resolución de errores de trabajo.

Por otra parte, en los últimos años han sido desarrolladas *SVMs* multiclase considerando todas las clases a la vez durante el proceso de aprendizaje, mediante el uso de diferentes algoritmos.

3.3.2 Definiendo la clasificación multiclase

El problema general de clasificación multiclase a partir de ejemplos:

Se define como una particularización del *PGAE*[*] en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un conjunto finito cuyos elementos pueden poseer o no una ordenación, $y = \{1,...,K > 2\}$, pero en cualquier caso el número de estas etiquetas definitorias de clase es estrictamente mayor que dos.

Se trata de un problema general que puede ser tanto de reconocimiento de patrones como de regresión ordinal, con la puntualización de no tratarse de un problema binario.

El problema de clasificación, $K > 2$, es solucionado habitualmente mediante la combinación de funciones de decisión biclase: Un esquema de descomposición inicial transforma las K-particiones en una serie de *L* biparticiones, $f_1,..., f_L$, mientras que un método de reconstrucción posterior realiza la fusión de las predicciones de los *L* clasificadores para seleccionar una de las *K* clases como respuesta final.

[*] Problema general de aprendizaje por ejemplos PGAE

3.3.3 Máquinas SVM multiclase

Se denomina máquina de clasificación multiclase a la arquitectura de máquinas de aprendizaje capaz de responder con una etiqueta de clasificación a cualquier entrada. ''Las arquitecturas multiclase de {descomposición, reconstrucción} con esquema de descomposición en paralelo son las mas utilizadas cuando se trabaja con nodos de dicotomía tipo *SVMC* (Máquinas de multiclasificación). En el método de descomposición, es posible codificar la información en elementos binarios $\{-1,+1\}$ mediante tres esquemas principalmente: los más tradicionales $1-v-1$, $1-v-r$ y el *ECOC*[*]. Este último puede usarse en el modo estándar, generalizado o en doble capa si las salidas de las dicotomías se interpretan como probabilidades.

Los métodos de reconstrucción basados en votación por signo tienen muchas variantes, votos solo positivos, votos positivos y negativos, computo global de todos los votos y computo de solo los votos de los clasificadores implicados.

La finalidad es establecer un grado de pertenencia de un valor de entrada a cada una de las posibles clases para acabar seleccionando el mayor. También es posible utilizarlo en la reconstrucción, ya sea como ayuda para deshacer empates o desde el inicio, el valor numérico de las salidas parciales. Ya se comento que no es una estrategia adecuada si se utilizan *SVMCs*, aunque los intentos de interpretar las salidas como probabilidades a posteriori o de normalización de las

[*] Error Correcting Output Codes (Código de salida de corrección de error)

salidas intentan superar este inconveniente intrínseco en la definición de las *SVMs*. Estudios empíricos muestran la mayor efectividad del método ***1-v-1*** sobre el ***1-v-r***, aunque de este último deben ser superados los inconvenientes de: *(a)* gran número de nodos de dicotomía, *(b)* aprendizaje solo sobre un subconjunto del total y *(c)* robustez contra fallos en las respuestas parciales. El método *ECOC* esta siendo la variante mas utilizada para intentar superar las desventajas *(b)* y *(c)* aunque ello supone un aumento muy significativo en el numero de nodos de dicotomía y por tanto en tiempo de calculo computacional. Otras estructuras posibles son aquellas que disponen una reconstrucción en árbol, ya sea basadas en resultados teóricos de generalización que permiten reducir el tiempo de evaluación, o definidas como una combinación con métodos de programación lineal con objeto de poder extraer información.

Los clasificadores únicos considerando todas las clases a la vez también son una alternativa, aunque conllevan la resolución de un problema de optimización de gran dimensión. Tres alternativas han sido mostradas, cada arquitectura obtenida desde perspectivas diferentes, con problemas de optimización primal parecidos.

Por ultimo, se ha definido un factor de robustez que permite analizar la posibilidad de incorporar fallos en los nodos de dicotomía sin que el resultado correcto final se vea afectado. Aunque la definición no es aplicable en todas las arquitecturas consideradas, se ha observado la nula robustez de la mayoría de máquinas, mientras que sobre aquellas consideradas mas robustas, las de estructura *ECOC*,

no es posible calcular el factor de robustez en forma general debido a la aleatoriedad en la elección de los nodos de dicotomía''[11*].

3.4 SVM para regresión

Para generalizar el método *SV* a la estimación de regresiones — método *SVMR*, *Support Vector Machine for Regresión — [Cortes and Vapnik, 1995]*, *[Smola and Schölkopf, 1998b]*, es necesario construir un elemento análogo al margen en el espacio de valores de salida, $Y \in \Re$, mediante el uso de la función de coste $e - insensitiva$ de *Vapnik*

$$\left| y \cdot f(x,w) \right|_e = \max\left\{0, \left| y \cdot f(x,w) \right| \right\} \cdot e \quad (24)$$

Para un $e \geq 0$ dado, el problema de optimización restringida asociado para el caso de estimación de regresiones es

$$\arg\min R_{SVMC}(w, j(*)) = \frac{1}{2}\|w\|_F^2 + D\sum_{i=1}^{l}\left(j_i + j_i^{(*)}\right), \quad (25)$$

Introduciendo multiplicadores de Lagrange, se obtiene el problema de optimización restringida: hallar multiplicadores a_i, $a_i^* \geq 0$ que hagan mínimo el funcional, la regresión estimada toma la forma

$$f(x) = \sum_{i=1}^{SV}\left(a_i^* - a_i\right) \cdot k(x_i, x) + b \quad (26)$$

[11] Angulo C.(Abril, 2001) aprendizaje con máquinas núcleo en entornos de multiclasificación. tdx.cesca. p. 21 Obtenido de la red mundial el 25 Abril del 2005: http:// www.tdx.cesca.es/tesis_upc/available/tdx-0628101-41150/05capitulo4.pdf

* Para profundización de estos temas remítase la dirección anterior

4 MÁQUINAS SVM MIXTAS

Las máquinas mixtas son aquellas que para su funcionamiento se basan en los dos principios, (Clasificación y Regresión). Existen numerosas máquinas *SVM* mixtas pero por razones de delimitación del trabajo solo tocaremos dos de las mas conocidas, profundizando las máquinas $K-SVCR$ propuesta para el desarrollo de nuestro trabajo.

4.1 Máquinas ℓ -SVCR para multiclasificación

Se introduce un nuevo tipo de *SVM* para la multiclasificación, denominada $\ell-SVCR$ [*], con objeto de evitar el principal inconveniente que presentan las máquinas *1-v-1*, pero manteniendo a la vez todas las ventajas de este tipo de esquema. Con objeto de dar una mayor claridad a los posteriores desarrollos, supongamos se quiere buscar una función que clasifique los vectores entrada de entrenamiento correspondiente a la clase $?_1$ de los de la clase $?_2$. Para ello, realizamos, en primer lugar, una ordenación de los vectores de entrenamiento de tal forma que los n_1 primeros pertenezcan a la clase $?_1$, los n_2 siguientes a la clase $?_2$ y los restantes $(n-n_1-n_2)$ pertenecen al resto de las clases, $\{q_3,.....,q_\ell\}$.

Como en el problema clásico de las *SVMs* buscamos, inicialmente, un hiperplano clasificador $f_{12}(x)=0$ que separe adecuadamente las clases $?_1$ y $?_2$, pero ahora imponemos que se tenga en que se tenga en cuenta el resto de las clases, en la

[*] ℓ -clases de máquinas de soporte vectorial con restricciones de Clasificación y Regresión

construcción del problema de optimización. De esta forma, al hiperplano $f_{12}(x)$ se le exige que deje los vectores entrada de la clase *?1* en la región $x \in \Re^d$, tal que $\{f(x) \geq 1\}$, a los vectores entrada de la clase *?2* en la región $x \in \Re^d$, tal que $\{f(x) \leq -1\}$ y para los vectores entrada restantes se le reserva una región dependiente de un parámetro* $0 \leq d < 1$ de tal forma que caigan en la región $x \in \Re^d$, tal que $\{f_{12}(x) \leq d\}$, es decir a diferencia de las *SVM 1-v-1*, donde se obligaba implícitamente a los vectores de entrenamiento restantes a que pertenecieran al hiperplano clasificador, con el parámetro δ se habilita una región de holgura alrededor de la solución donde incluir todos los vectores restantes de entrenamiento.

Si dicha solución existe considerando un hiperplano clasificador de la forma $f_{12}(x) = \langle w, x \rangle + b,$ entonces se podrá resolver el siguiente problema de optimización** ℓ*-SVCR* sin pérdidas***. Al igual que en las *SVMs* estándar, la solución obtenida en el problema de optimización ℓ*-SVCR* puede ser generalizada al caso no lineal**** utilizando funciones núcleo*****, obteniéndose de esta forma como solución general al problema ℓ*-SVCR*:

$$f_{12}(x) = \sum_{i=1}^{N_{sv}} a_i k(x_i, x) + b \quad (27)$$

* La condición de ser menor que la unidad viene impuesta con objeto de no solapar las diferentes regiones que se construyen. Este parámetro es elegido a priori.
** Si $n_3 = n_1 + n_2$.
*** Es decir, la solución obtenida clasifica perfectamente todos los vectores de entrenamiento.
**** No se buscan necesariamente hiperplanos en el espacio de las entradas.
***** Dado el espacio de los vectores entrada X se considera una transformación f de este espacio en un espacio vectorial dotado de un producto escalar H (denominado espacio característico) en la forma: $f : X \subset \Re^d \rightarrow H \subset \Re^{d'}$

Es decir, el problema de clasificación no se realiza sobre los vectores de entradas directamente sino a través de sus transformados $f(x)$.

4.2 Máquina de aprendizaje K-SVCR

"Las máquinas de soporte vectorial que trabajan en problemas de clasificación están definidas de forma especifica para tratar problemas de clasificación binaria. Un nuevo algoritmo basado en vectores soporte será definido con el objetivo posterior de ser utilizado durante el esquema de descomposición de multiclasificación, denominado algoritmo $K-SVCR$. Cuando se trata un problema de clasificación multiclase, $K>2$, toma sentido construir máquinas de aprendizaje que asignen salida *+1* o *-1* si el patrón de entrenamiento pertenece a las clases a ser separadas, y salida 0 si por el contrario el patrón tiene una etiqueta diferente a las anteriores. Así se está forzando al hiperplano separador de las dos clases implicadas no sólo a poseer características de margen máximo, sino estará restringido a recubrir todos los patrones de entrenamiento "*0* etiquetados", aquellos pertenecientes a cualquier otra clase que no sean las dos iniciales. El problema de programación cuadrática restringido asociado podría ser interpretado como un intermedio entre el método *SVMC* y el método *SVMR* de entrenamiento sobre vectores soporte para estimación de regresiones"[12].

[12] Angulo C.(Abril, 2001) aprendizaje con máquinas núcleo en entornos de multiclasificación. tdx.cesca. p. 1 Obtenido de la red mundial el 25 Abril del 2005: http:// www.tdx.cesca.es/tesis_upc/available/tdx-0628101-41150/05capitulo4.pdf

4.2.1 Por que de las máquinas K-SVCR

El funcionamiento y características que poseen las diferentes arquitecturas de clasificación basada en clasificadores binarios, en concreto *SVMCs,* permiten "elaborar un esquema de propiedades que sería deseable que cumpliera una máquina de clasificación ideal. Estas propiedades podrían resumirse en:

1. Un esquema de descomposición de baja complejidad, que se traduciría en el uso de un número reducido de dicotomías.
2. Un problema de optimización asociado de baja dimensión, que permitiría obtener resultados con un costo computacional no muy elevado.
3. Un conjunto de entrenamiento para cada dicotomía equilibrado y de tamaño similar al conjunto de aprendizaje original.
4. Robustez de las respuestas ante fallos parciales de los nodos de dicotomía.
5. Una base teórica que permita asegurar una cota baja en el error de generalizado.
6. Un tiempo de evaluación de resultados lo más reducido posible"[13].

Muchas de estas propiedades son opuestas las unas de las otras. Un esquema tipo "todas las clases a la vez" posee una descomposición sencilla, una única máquina pero el problema de optimización cuadrática es de gran dimensionalidad. Una mejora de robustez del método *1-v-1* mediante técnicas de *ECOC* implica un aumento significativo en el número de dicotomías del esquema de descomposición de igual forma que el entrenamiento de una dicotomía sobre todo el conjunto de

[13] Ibid., p.2

aprendizaje implica un costo computacional más elevado, el objetivo de basar la clasificación multiclase en árboles que posean una base teórica de buena generalización y reducción en el tiempo de evaluación provocan una arquitectura nada robusta debido a que se disminuyen algunas restricciones.

Hallar la mejora de algunas de estas características de comportamiento, sin provocar una reducción en el grado de cumplimiento de las restantes es el objetivo que motiva la búsqueda de una estructura equilibrada respecto a todas las propiedades de la que pueda preverse un mejor comportamiento global, sin necesidad de hacer referencia explícita a un tipo u otro de 'benchmark de validación aunque su comportamiento sobre estos problemas deberá ser en general mejor que el resto de arquitecturas.

4.2.2 Composición de la Máquina K-SVCR

"Una vez analizada la profundidad del problema de aprendizaje multiclase desde la perspectiva de la buena generalización que ofrecen los biclasificadores *SVMC*, se eligió como esquema base de descomposición el método *1–v–1* debido a la dicotomía obtenidas con esta formulación permiten concentrar el esfuerzo clasificador. Sobre una zona determinada del espacio de clasificación no hay que olvidar sin embargo que la exclusión aportada por las entradas pertenecientes al

resto de clases provoca que la fisonomía del hiperplano de decisión no se adapte a esta"[14].

La idea que permitirá incluir otras clases en el entrenamiento de la dicotomía es una generalización de una propiedad muy empleada en el cálculo de *SVMCs*. Supóngase que un usuario intenta resolver el problema cuadrático de optimización primal asociado a una máquina *SVMC*, de función objetivo, con restricciones que conduce a una función de decisión basada en un hiperplano optimo. Tal como se afirma en *[Burges, 1998]*, es muy común suponer que el termino independiente es nulo, $b=0$, por lo que el hiperplano optimo hallado será del estilo

$$f(x,w)=\langle x,w\rangle_F = k(x,w). \quad (28)$$

4.2.3 Abarcando La Máquina de Aprendizaje K-SVCR

"La definición de la función de decisión ternaria sobre un planteamiento de problema de clasificación con tres clases crea la necesidad de definir una máquina de aprendizaje capaz de triclasificar basada en las *SVMCs*. Siguiendo el método de exposición usual, se comenzará por definir la nueva máquina en el caso separable para luego generalizar el caso no separable. De hecho, la exposición de máquinas *SVMC* suele comenzar con el caso linealmente separable y continuar con el no linealmente no separable. En la presente formulación se hace siempre necesario introducir el espacio de entrada *X en un espacio de características de*

[14] Ibid., p.2

mayor dimensión para asegurar el éxito de la clasificación por lo que los casos lineales no son tratados[*],[16].

4.2.4 Experimentación

"Para comprobar las características de funcionamiento y las prestaciones de la nueva máquina de aprendizaje $K-SVCR$ se utilizarán una serie de problemas artificiales tanto para el caso separable como el no separable. El objetivo principal de los experimentos será de una parte analizar el papel desempeñado por el nuevo factor de insensitividad δ introducido en la nueva algorítmica y por otra destacar el mejor comportamiento de la nueva máquina respecto a un diseño de multiclasificación con máquinas *SVMR*"[17].

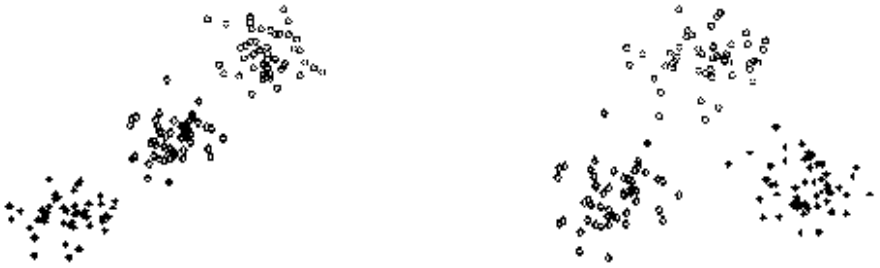
Se ha de recordar que las máquinas *SVMC* solo pueden biclasificar, por lo que la generalización hacia el caso multiclase podría desarrollarse mediante *SVMRs*.

Se probará que la utilización de la máquina *SVMR* significa llevar la generalización hacia la multiclasificación demasiado allá y que resulta mas correcto una máquina intermedia, como la *K-SVCR*, con restricciones de tipo mixto clasificación y regresión para desarrollar esta tarea.

[*] En caso que un separador lineal bastase, el separador no lineal construido por la nueva máquina adoptara una fisonomía quasilineal en la zona cercana a los patrones de entrenamiento.

[16] Ibid., p.4

[17] Ibid., p.9



(a) Conjunto de entrenamiento T1. (b) Conjunto de entrenamiento T2.

Figura 13. Conjuntos de entrenamiento linealmente separables.

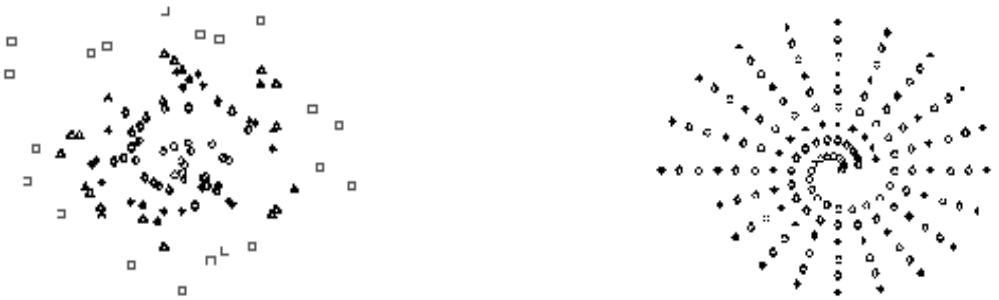
4.2.5 Problemas Artificiales

"Para el caso linealmente separable, se han construido dos tipos de conjunto de entrenamiento, *T1* y *T2*, sobre el plano, R^2, formados por *150* patrones repartidos en 3 clases, con igual número de representantes cada una, generados mediante distribuciones gaussianas de varianza unitaria, *Figura 13*. En el caso de la *Figura 13(a)* los centros de cada distribución han sido situados en línea, mientras que en la *Figura 13(b)* los centros forman un triangulo. El uso de ejemplos similares puede ser observado en *[Kressel, 1999]* para mostrar la eficacia de la clasificación multiclase "por parejas". Con la intención de reducir el nivel de restricción del problema *QP* a resolver en la creación de la máquina $K-SVCR$ también se han creado los conjuntos de entrenamiento *T3* y *T4* tomando de forma aleatoria *15* patrones de los *45* de cada una de las tres clases. El caso no separable será tratado sobre los conjuntos de entrenamiento *T5* y *T6* representados en la *Figura 14(a)* y *la Figura 14(b),* respectivamente. Nuevamente se trata de conjuntos sobre el plano, R^2, con igual número de representantes para cada clase. El conjunto *T5*

está formado por *100* patrones repartidos en *5* clases formando una distribución gaussiana y han sido separados en función del radio al centro de esta distribución. Para el conjunto *T6* los *150* puntos de entrenamiento han sido distribuidos formando una espiral"[18].

4.2.6 Factor de insensitividad d

"Las restricciones impuestas sobre los *3* patrones correspondientes a la etiqueta *0* en la definición de la máquina $K-SVCR$, tanto para el caso separable como para el no separable provocan que la fisonomía de la función de decisión ternaria dependa de forma muy importante de la definición del parámetro de insensitividad δ"[19]. En esta Subsección se realizan una serie de experimentos sobre los problemas artificiales definidos anteriormente que permitirían constatar la inuencia de este parámetro sobre el resultado final de la clasificación.



(a) Conjunto de entrenamiento T5. (b) Conjunto de entrenamiento T6.

Figura 14. Conjuntos de entrenamiento no linealmente separables.

[18] Ibid., p.10

[19] Angulo C.(Abril, 2001) aprendizaje con máquinas núcleo en entornos de multiclasificación. tdx.cesca. p. 10 Obtenido de la red mundial el 25 Abril del 2005: http:// www.tdx.cesca.es/tesis_upc/available/tdx-0628101-41150/05capitulo4.pdf

Inicialmente se utiliza el conjunto de entrenamiento *T1* para observar el comportamiento de la máquina de aprendizaje en función del factor de insensitividad en un caso separable. Para ello se utilizará una función polinomial de grado *4* como núcleo.

En la *Figura 15* aparece la evolución de la función de decisión según aumenta el nivel de insensitividad. Las figuras han sido dispuestas en columnas según la elección que se ha determinado de los patrones con etiqueta *0*, mientras que a cada fila le corresponde un mismo valor *d*. Las cantidades que aparecen debajo de cada subfigura corresponden al factor de insensitividad utilizado, el tiempo de computación y el número de vectores de soporte necesarios en la expansión de la función de decisión. Tal como ya fue apuntado, cuanto menor sea el parámetro *d*, menor es la generalización del espacio para los patrones con etiqueta *0*, mayor es el tiempo de computación y mayor es el numero de vectores soporte, los cuales han sido marcados sobre las graficas.

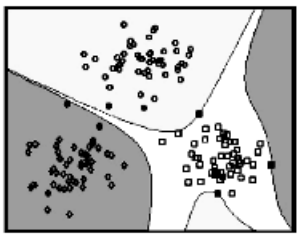
4.2.7 Comparativa con la SVMR

Utilizando el conjunto de entrenamiento de clases linealmente separables *T2*, es entrenada una $K-SVCR$ y una máquina *SVMR* estándar con salidas $\{-1,0,+1\}$ asignadas a cada clase en cualquiera de sus combinaciones, por lo que se dispondría de tres

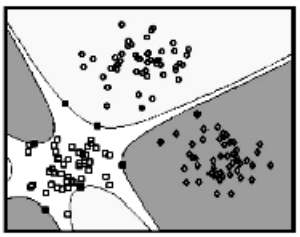
(a) e=0.050, t=114.8s, nsv=13

(b) e=0.050, t =121.0s, nsv=14

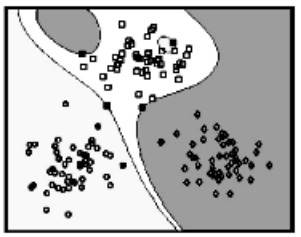
(c) e=0.050, t=138.2s, nsv=14



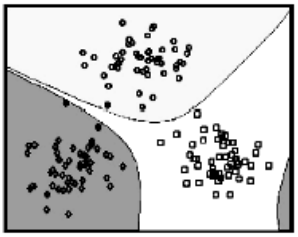
(d) e=0.250, t= 92.2s, nsv=7



(e) e=0.250, t=101.3s, nsv=9



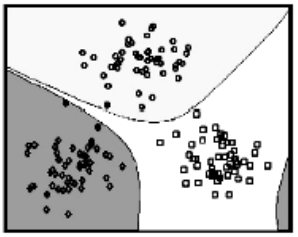
(f) e=0.250, t=110.8s, nsv=7



(g) e=0.500, t=86.0s, nsv=3

(h) e=0.500, t=98.1s, nsv=9

(i) e=0.500, t=96.2 s, nsv=6

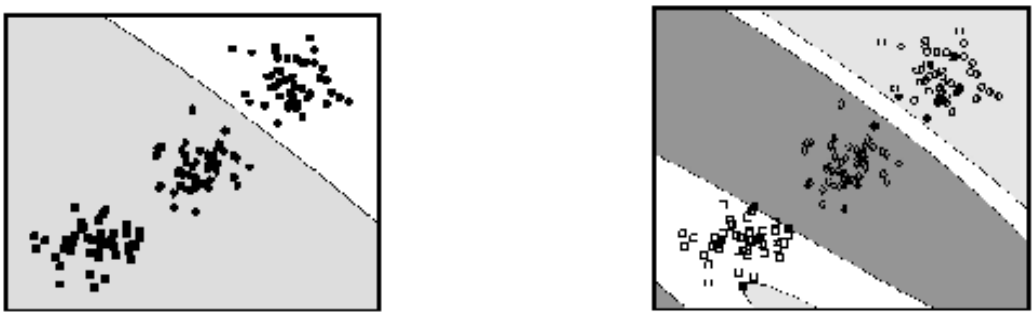


(j) e=0.999, t=89.1s, nsv=3

(k) e=0.999, t=87.9s, nsv=5

(l) e=0.999, t=96.8s, nsv =5

Figura 15. Resultados para diferentes niveles de insensitividad del entrenamiento sobre el conjunto T1. Las cantidades que acompañan cada subfigura representan el nivel de insensitividad, el tiempo de entrenamiento y el número de vectores de soporte utilizados.



(a)Máquina K-SVCR entrenamiento conjunto T2. (b) Máquina SVMR entrenamiento conjunto T2.

Figura 16. Resultados del entrenamiento sobre el conjunto T2 utilizando funciones núcleo polinomiales de grado n = 3.

Máquinas para cada tipo de arquitectura. Para el entrenamiento es utilizado un factor $C = a$ porque las clases son separables y no se hace necesario el uso de variables artificiales y una insensitividad de nivel medio, $d = 0.5$.

Si se utiliza como núcleo una función polinomial de grado *n = 3*

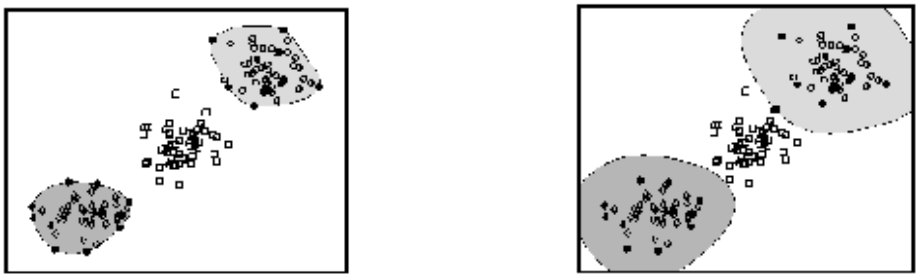
$$k(x, y) = (1 + x \cdot y)^n, \qquad (29)$$

Se obtienen los siguientes resultados: las máquinas $K - SVCR$ emplean 9*3.0, 86.3* y *92.2* segundos en ser entrenadas y la función solución se expande sobre *6,4* y *5* vectores soporte respectivamente, mientras que la *SVMR* tras *375.8, 365.9* y *367.6* segundos de entrenamiento, aunque la función solución se expanda sobre los 150 patrones de entrenamiento no consigue clasificar bien como puede apreciarse en las dos graficas de la *Figura 16*.

"El motivo del malfuncionamiento de la *SVMR* esta en la poca amplitud o capacidad del espacio de *Hilbert* generado por el núcleo polinomio de grado *3* espacio de características *F*. En el caso $K - SVCR$ el espacio *F* resulta suficiente

para conseguir separar las clases, pero la mayor exigencia impuesta por las restricciones de la *SVMR* hacen que el nuevo espacio no sea lo bastante amplio como para permitir la clasificación correcta por esta máquina. Esta experimentación ha permitido ilustrar como la exigencia de cubrir todos los patrones de etiqueta *0* por una zona insensitiva entorno a la función de decisión no resulta un requerimiento demasiado extremo, pues incluso una máquina *SVMR* tradicional implica mayores restricciones"[20].

Para conseguir aumentar el espacio de características sobre el conjunto de patrones de entrenamiento existen dos posibilidades, o bien se usa un núcleo que genere un espacio de *Hilbert* de mayor dimensión o bien se reduce el numero de patrones de entrenamiento.



(a)Máquina K-SVCR entrenamiento conjunto T2. (b) Máquina SVMR entrenamiento conjunto T2.

Figura 17. Resultados del entrenamiento sobre el conjunto T2 utilizando funciones núcleo gaussianas de varianza 0.5.

[20] Ibid., p.13

Para ilustrar la primera posibilidad se ha optado por elegir como funciones núcleo a gaussianas de varianza *0.5.* El espacio *F* generado por las gaussianas tiene dimensión *VC* infinita por lo que siempre ha de existir una solución, pero el precio a pagar es un mayor coste en tiempo computacional — no es equivalente evaluar un polinomio que una función exponencial — y la imposibilidad de asegurar generalización. Tras realizar la experimentación se obtienen tres máquinas de aprendizaje $K-SVCR$ que se expanden sobre *12, 13* y *10* vectores soporte y que han necesitado de *102.5, 105.4* y *102.0* segundos respectivamente para ser entrenadas. Nuevamente dos de las *SVMRs* entrenadas emplean un mayor tiempo computacional en este proceso que la nueva máquina, *495.6* y *480.8* segundos con *12* vectores soporte en ambos casos, mientras que para la tercera de ellas su entrenamiento fue detenido tras mas de *14* horas de computación sin haber sido obtenida la solución. En la *Figura 17* pueden observarse los resultados de dos de las máquinas entrenadas con los dos tipos de arquitectura.

La segunda opción para reducir la demanda en las restricciones de las máquinas de aprendizaje consiste en reducir el número de patrones. Para el siguiente experimento se han seleccionado solo *15* de los *50* patrones que componen cada clase del conjunto de aprendizaje *T1*, conjunto *T3,* y se ha entrenado cada tipo de máquina. Como era de esperar, el tiempo de computación es mucho menor y se ha obtenido como resultado *2.2, 1.7* y *1.8* segundos de entrenamiento y *5, 3* y *4* vectores soporte para la nueva máquina, y 6.3, 5.6 y 9.8 segundos para entrenar la máquina *SVMR* para obtener *7, 2* y *45* vectores soporte. Como puede deducirse de

los datos y apreciarse en la *Figura 18,* la tercera de las elecciones de etiquetado vuelve a causar problemas a la máquina *SVMR* pues incluso expandiendo la solución sobre todos los vectores soporte se cometen errores en el entrenamiento. Los tres experimentos anteriores con núcleo polinomial de grado *3* sobre todo el conjunto de entrenamiento, sobre una selección aleatoria de *45* patrones en total, y con núcleo de funciones gaussianas de varianza *0.5* sobre todo el conjunto de aprendizaje —han sido también realizados sobre el conjunto *T1* con resultados similares a los obtenidos sobre *T2,* como puede observarse en la *Tabla 1.*

(a)Máquina K-SVCR entrenamiento conjunto T3. (b)Máquina SVMR entrenamiento conjunto T3.

Figura 18. Resultados del entrenamiento sobre el conjunto T3 utilizando funciones núcleo polinomiales de grado n =3.

4.2.8 Problemas Artificiales No Linealmente Separables

"Para ilustrar el funcionamiento de la máquina $K-SVCR$ sobre problemas no linealmente separables se han utilizado los conjuntos de entrenamiento artificiales *T5* y *T6* ya descritos anteriormente. En el caso del conjunto *T5*, se dispone de $K=5$ clases a ser separadas y de un número muy reducido de patrones para cada clase. Para este problema se ha utilizado como núcleo una función gaussiana que asegure la existencia de solución. El parámetro de

insensitividad d es de *0.75,* mayor que el propuesto para los casos linealmente separables, puesto que la zona de *0* etiquetado corresponde ahora a *3* clases y en principio corresponderán con una zona geométrica mas amplia que la representada por las dos clases a ser separadas de forma individual, *±1*. Además, el amplio de las campanas de Gauss se ha establecido en $s = 0.45$ intentando que el numero de vectores soporte que aparezcan sea mas bien elevado en comparación con el numero de patrones ya que se dispone de muy pocos de estos últimos"[21].

En la *Figura 19* se muestran algunas de las máquinas $K-SVCR$ obtenidas en función de las clases a ser separadas y a ser etiquetadas con *0*. Los resultados globales han sido sumariados en la *Tabla 2*, donde se muestran el tiempo de ejecución del entrenamiento en segundos y el número de vectores soporte.

núcleo	parám.	#pat	máquina		1-2-3	1-3-2	2-3-1
polin.	3	150	K-SVCR	tiempo	89.5	96.9	90.5
				#SV	4	8	5
			SVMR	tiempo	366.1	364.4	361.5
				#SV	150	150	150
gauss.	0.5	150	K-SVCR	tiempo	109.1	99.9	102.7
				#SV	15	14	15
			SVMR	tiempo	495.0	470.0	$\geqslant$ 5400
				#SV	15	14	?
polin.	3	45	K-SVCR	tiempo	2.3	1.7	2.0
				#SV	4	5	5
			SVMR	tiempo	6.4	5.9	7.2
				#SV	5	5	6

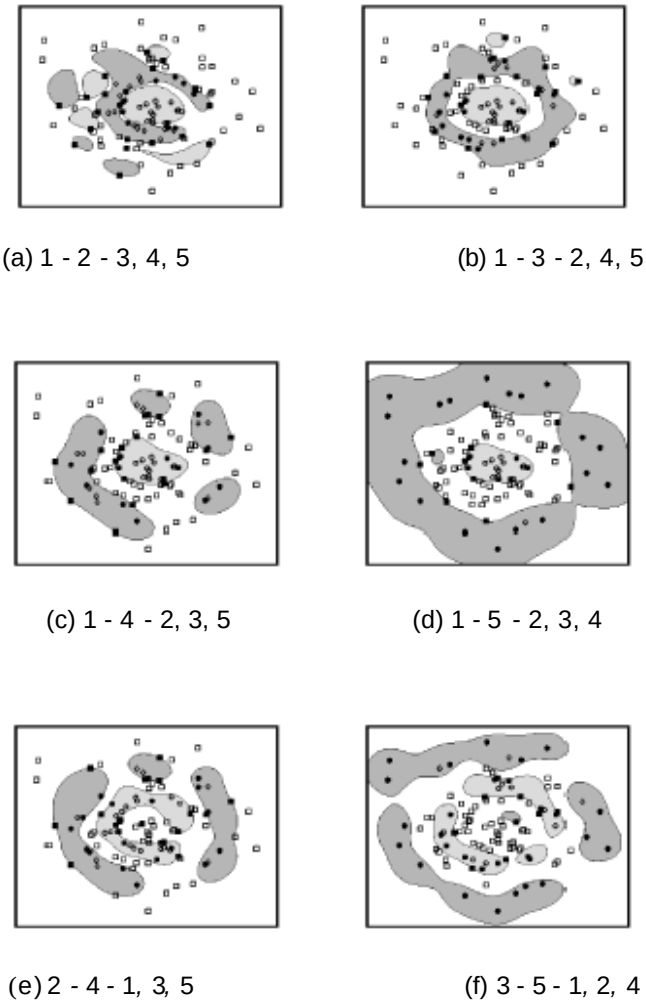
Tabla 1. Resultados sobre el conjunto de entrenamiento T1

[21] Ibid., p.15

etiquetas	1-2-3,4,5	1-3-2,4,5	1-4-2,3,5	1-5-2,3,4	2-3-1,4,5
tiempo	76.1	47.6	46.2	47.7	63.8
nsv	30	30	31	35	45

etiquetas	2-4-1,3,5	2-5-1,3,4	3-4-1,2,5	3-5-1,3,4	4-5-1,2,3
tiempo	54.9	58.0	118.1	58.3	60.5
nsv	36	44	48	38	38

Tabla 2. Resultados sobre el conjunto de entrenamiento T5 utilizando núcleos gaussianos.



(a) 1 - 2 - 3, 4, 5 (b) 1 - 3 - 2, 4, 5

(c) 1 - 4 - 2, 3, 5 (d) 1 - 5 - 2, 3, 4

(e) 2 - 4 - 1, 3, 5 (f) 3 - 5 - 1, 2, 4

Figura 19 Resultados del entrenamiento sobre el conjunto T5 utilizando núcleos gaussianos para diferentes elecciones de etiqueta de clase.

etiquetas	1-2-3,4,5	1-3-2,4,5	1-4-2,3,5	1-5-2,3,4	2-3-1,4,5
tiempo	181.8	79.1	101.5	71.9	114.7
nsv	64	50	48	33	64

etiquetas	2-4-1,3,5	2-5-1,3,4	3-4-1,2,5	3-5-1,3,4	4-5-1,2,3
tiempo	70.5	31.6	78.3	65.4	54.1
nsv	44	100	48	31	20

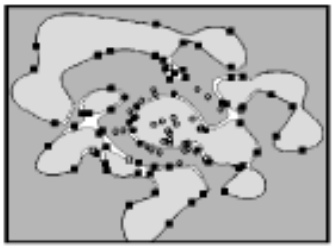
Tabla 3. Resultados sobre el conjunto de entrenamiento T5 utilizando núcleos polinomiales.

etiquetas	1-2-3	1-3-2	2-3-1
tiempo	244.8	222.6	316.3
nsv	98	79	95

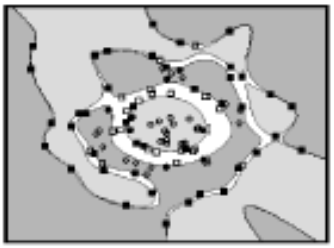
Tabla 4. Resultados sobre el conjunto de entrenamiento T6 utilizando núcleos gaussianos.

Debe resaltarse la especial dificultad que le comporta a la máquina realizar la separación cuando las clases no etiquetadas nulas se hallan contiguas.

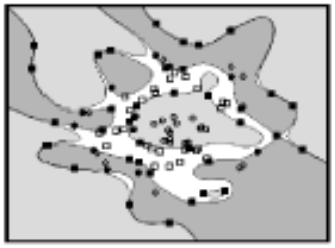
También ha sido utilizado un entrenamiento sobre núcleos polinomiales de grado 10 con igual factor de insensitividad que en el caso anterior, $d = 0.75$, con la intención de mostrar la eficiencia de la nueva máquina sobre un espacio de dimensión *VC* finita. Los resultados han sido reflejados en las graficas de la *Figura 20* y la *Tabla 3*. Para el conjunto de entrenamiento *T6* se vuelve a tener *3* clases con los patrones repartidos equitativamente formando espirales anidadas. Utilizando núcleos gaussianos con los parámetros habituales $s = 0.5$ y $d = 0.5$ se obtienen los resultados mostrados en la *Figura 21* y recogidos en la *Tabla 4*.



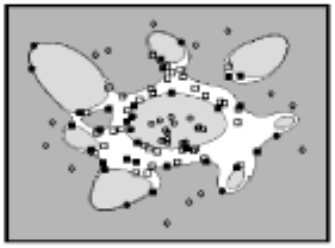
a) 1 - 2 - 3, 4, 5



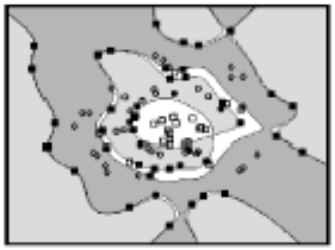
b) 1 - 3 - 2, 4, 5



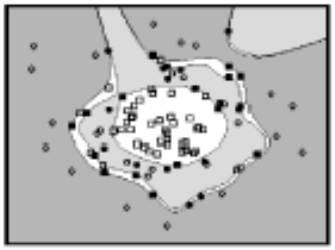
c) 1 - 4 - 2, 3, 5



d) 1 - 5 - 2, 3, 4



e) 2 - 4 - 1, 3, 5



f) 3 - 5 - 1, 2, 4

Figura 20. Resultados del entrenamiento sobre el conjunto T5 utilizando núcleos polinomiales para diferentes elecciones de etiqueta de clase.

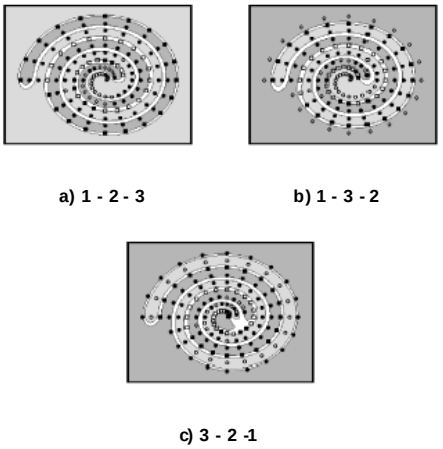


Figura 21. Resultados del entrenamiento sobre el conjunto T6 utilizando núcleos gaussianos para diferentes elecciones de etiqueta de clase

Se ha desarrollado la formulación matemática del problema *QP* asociado tanto para el caso linealmente separable como para el no linealmente separable. Este problema de programación cuadrática podría ser interpretado como un intermedio entre el método *SVMC* y el método *SVMR* de entrenamiento sobre vectores soporte para estimación de regresiones. Finalmente, se ha mostrado el modo de funcionamiento de las $K-SVCRs$ aplicadas a problemas artificiales que permitan su fácil visualización. Así, se ha mostrado su eficiencia respecto a una *SVMR* con tres salidas gracias a su menor demanda en las restricciones del problema *QP* asociado; el nuevo factor de insensitividad ligado a aquel definido por *Vapnik* para crear las *SVMRs* resulta critico en la definición de la tercera clase, su amplitud y el numero de vectores soporte; la formulación permite trabajar con eficiencia con cualquier numero de clases tal como se ha podido observar con un problema no linealmente separable de $K=5$ clases.

4.3 Comparación SVM "K-SVCR" con otras máquinas de aprendizaje

4.3.1 SVM "K-SVCR" vs. Algoritmos genéticos "AG"

- ✓ **Diversidad de soluciones**

AG: Son algoritmos estocásticos. Dos ejecuciones distintas pueden dar dos soluciones distintas sobre un mismo conjunto de elementos. Son algoritmos de búsqueda múltiple, luego dan varias soluciones. Aunque habitualmente las energías de los individuos de la población final es similar, los individuos suelen ser distintos entre si. Necesitan de programador expertos para seleccionar la mejor solución.

SVM: Típicamente hay muchos hiperplanos infinitos semejantes, obtenidos por las variaciones pequeñas de una solución dada sobre un mismo conjunto de elementos. Facilitan la elección a un programador del hiperplano ideal para la solución, el cual se encuentra el medio de los hiperplanos paralelos que cortan las muestras más cercanas del punto de convergencia (Clasificación).

- ✓ **Convergencia y resultado final**

AG: A diferencia de los otros algoritmos cuya convergencia y resultado final son fuertemente dependientes de la posición inicial, en los algoritmos genéticos -salvo en los que el operador de mutación va a tener mucho trabajo- la convergencia del algoritmo es poco sensible a la población inicial si esta se escoge de forma aleatoria y es lo suficientemente grande. Por su grado de penetración casi nulo, la curva de convergencia asociada al algoritmo presenta una convergencia excepcionalmente rápida al principio aunque puede que no converja.

SVM: El resultado final dependerá de cuan tan estrictas son la restricciones que se le propongan a máquina y al numero de vectores de soporte que se utilicen, se ha logrado mediante algunas modificaciones que estas máquinas siempre converjan.

- ✓ **Conformación de la máquina**

AG: La máquina estructuralmente se define conformada por cromosomas artificiales.

SVM: La máquina estructuralmente se define formando por módulos que realizan su trabajo por medio de vectores de soporte.

- ✓ **Eficacia**

AG: Se debe escoger realmente mal los parámetros del algoritmo para que no converja.

SVM: Siempre converge (Luego de las modificaciones)

- ✓ **Optimización**

AG: La optimización es función de la representación de los datos. Es una búsqueda paramétricamente robusta. Los algoritmos genéticos son internamente módulos paralelos.

SVM: La optimización se realiza por métodos de minimización de riesgos y seleccionando bien el espacio de aproximaciones.

4.3.2 SVM "K-SVCR" vs. Redes neuronales "RN"

Las redes neuronales poseen más afinidad con las *SVM* por lo cual enfocaremos la comparación de una manera diferente. La estructura es bastante similar tanto de las *RNs* como de las *SVMs,* básicamente difieren en la forma del aprendizaje, en la complejidad de algoritmos y el tiempo de adiestramiento de una *SVM* es relativamente mas corto que el de una *RN*.

En los problemas cuando los hiperplanos de decisión lineales no es posible alargarlos, un espacio de la entrada se traza en un espacio del rasgo (la capa oculta en modelos *RN*), produce un "clasificador no lineal''. Por el contrario las *SVMs* siempre que entrenan encuentran un mínimo global.

Al contrario de la estadística convencional y los métodos de la red neuronal, la *SVM* no intenta aproximar la complejidad del modelo de control guardando el número de rasgos pequeño. Los Kernel Gaussian mas frecuentes son usados, cuando el resultado *SVM* corresponde a una red de *RBF* con función de base radial Gaussiana. Cuando los *SVM* se aproximan "automáticamente" resuelve el problema de complejidad de red, el tamaño de la capa oculta se obtiene como el resultado de *QP*. Las neuronas ocultas y vectores de soportes corresponden a nosotros, para que también se resuelven los problemas del centro de la red de *RBF*, cuando los vectores de apoyo sirven como los centros" de función de base. Mientras el término de decaimiento de peso es un aspecto importante por obtener una buena generalización en el contexto de redes neuronales; para la regresión, el margen juega un papel algo similar en los problemas de la clasificación.

Comparado con redes neuronales de perceptron de multicapas tradicionales estas padecen la existencia de soluciones de los mínimos locales múltiples, la convexidad es una propiedad importante e interesante de la no linealidad de los clasificadores de *SVM*.

5 APLICACIONES Y SOFTWARE DE LAS SVMs

5.1 Aplicaciones

Entre las aplicaciones más comunes de las *SVMs* tenemos:

SVMs para el reconocimiento de rostros, sistemas de reconocimiento facial para el control de acceso automático, categorización de texto, reconocimiento de pupila, reconocimiento de imágenes, recuperación de información, predicción de la bolsa de valores, modelos *SVM* difusos, identificación de *TS* (sistemas de transporte) en sistemas *MIMO*, sistemas de transporte inteligente *ITS*, sistemas de asistencia al manejo, reconocimiento automático de voz y sistema de visión integrado para reconocimiento humano, categorización de texto, reconocimiento del habla.

5.1.1 Reconocimiento automático de voz

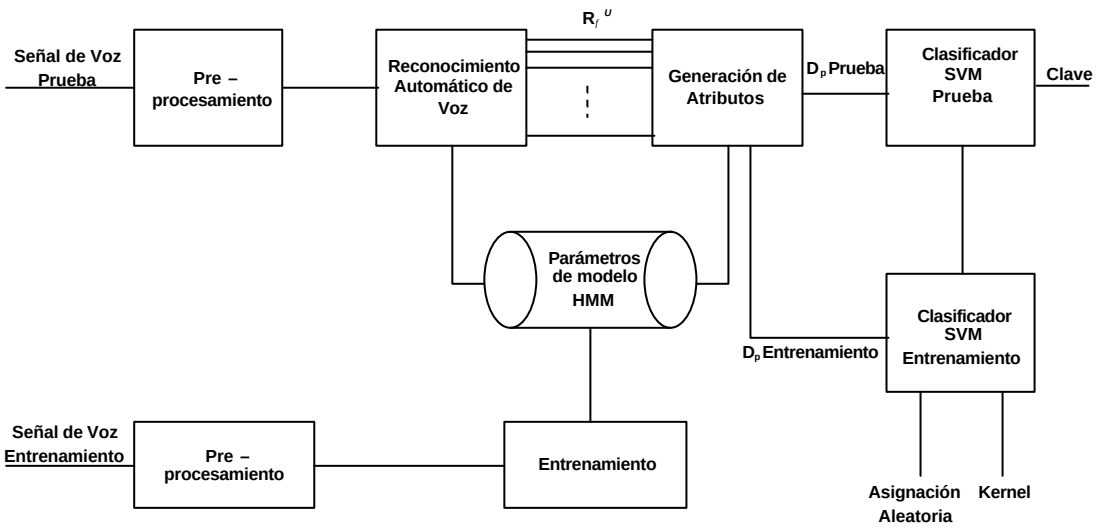


Figura 22. Diagrama de bloques para reconocimiento automático de voz

"Para la generación de la clave se exploran las técnicas de Reconocimiento Automático de Voz (*ASR*, Automatic Speech Recognition), de Máquinas de Vectores de Soporte (*SVM*, Support Vector Machine, un método de clasificación de patrones), y de un bloque intermedio (generador de atributos) que permite hacer una conexión entre las dos técnicas anteriores, como se muestra en la *Figura 22*"[22].

"El objetivo principal del *ASR* en esta investigación es encontrar la trascripción de lo que dice cada usuario y los inicios y finales de cada fonema en cada articulación. Esta parte se divide en dos fases: entrenamiento y prueba. Las señales de voz de varios usuarios contenidas en una base de datos se dividen en dos grupos: *60%* del total son utilizadas por la fase de entrenamiento y el *40%* restante, por la fase de prueba. En el pre-procesamiento, la señal de voz es representada por una serie de coeficientes (*MFCC*, siglas en inglés de Mel Frequency Cepstral Coefficients-Coeficientes Cepstrales de Frecuencia Mel, representación de voz de manera matemática). Posteriormente, se crea un modelo acústico utilizando las ocultas cadenas de Markov (*HMM*, Hidden Markov Model). Este modelo es un método bien conocido y se utiliza para caracterizar la señal de voz como un proceso aleatorio paramétrico (considerando que los parámetros del proceso estocástico pueden ser estimados de manera precisa). Los parámetros están formados por las medias y varianzas de cada uno de los fonemas. En la

[22] García P., Mex C., Nolazco J. (Enero, 2005). Claves criptográficas basadas en la señal de voz. Revista. p.1. . Obtenido de la red mundial el 23 Abril del 2005 :http://Revista - Transferencia.htm

fase de prueba o reconocimiento, nuevamente se realiza el pre-procesamiento de cada articulación. Posteriormente, se obtienen los inicios y finales de cada uno de los fonemas utilizando el modelo mencionado."[23]

"A continuación, las medias del modelo y los inicios y finales de los fonemas se procesan en el bloque intermedio, el cual tiene como finalidad crear subconjuntos de vectores de fonemas iguales. Así, de cada usuario del sistema de seguridad informática se extraen los *MFCC* pertenecientes a cada fonema, se obtiene su promedio y se le resta la media correspondiente contenida en el modelo. Posteriormente, se forman subconjuntos: un 80% de estos subconjuntos servirá de entrada a la etapa de entrenamiento del *SVM* y el restante *20%* formará parte de la fase de prueba."[24]

"El método *SVM* es ampliamente usado para clasificación basado en la utilización del kernel, una función que transforma los datos a una dimensión superior, de tal manera que sean más fáciles de separar (separación lineal). El objetivo primordial del *SVM* básico es la clasificación de vectores en una de dos clases. Para nuestra investigación se requiere que cada uno de los conjuntos de atributos divididos en fonemas puedan ser particionados de manera que a algunos usuarios les corresponda la clase *1* y al resto, la clase *-1*. Nuevamente, nos encontramos con dos fases: entrenamiento y prueba. En la fase de entrenamiento se toman los subconjuntos generados por el bloque intermedio y se asocia a cada uno de los

[23] Ibid., p.1
[24] Ibid., p.1

vectores de cada fonema con una etiqueta con los valores que puede tener (1 *ó* -1), siendo éstas últimas las claves (anteriormente llamadas clases) de cada usuario. Las etiquetas se producen por medio de un generador pseudo-aleatorio, lo que brinda un mayor control sobre la clave y se reduce la posibilidad de que una persona no autorizada suponga que hay claves "preferidas", ya que se usan todas por igual. En el entrenamiento se requiere escoger un kernel y realizar una sintonización de sus parámetros, i.e., encontrar los valores óptimos de cada una de las variables de cada parámetro"[25]. Para esta investigación los diseñadores utilizaron los siguientes tipos de kernel: lineal, polinomial y *RBF* (siglas Radial Basis Function-Función de Base Radial). Al concluir el entrenamiento se obtiene un nuevo modelo (vectores de soporte) con el cual se pueden generar las claves finales utilizando los vectores de prueba.

5.1.2 Un sistema de visión integrado para el reconocimiento humano

"Para investigar computacionalmente los modelos eficaces para el reconocimiento del tiempo real de actividades humanas (*QMW*) Grupo de Investigación de Máquina Visuales, se ha desarrollado una plataforma de sistema integrado - la Interacción Visual basado en los Gestos y nuestra conducta"[26].

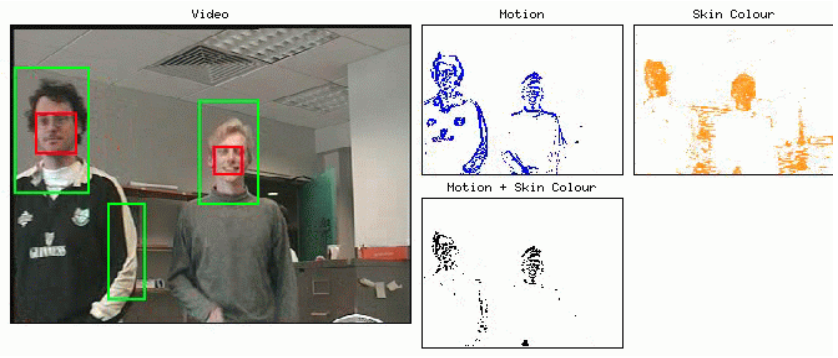
[25] Ibid., p.1
[26] Sherrah J., Gong S.(septiembre, 2000). VIGOUR: An Integrated Vision System for Research in Human Recognition. dcs.qmw p.1. Obtenido de la red mundial el 23 Abril del 2005: http://www.dcs.qmw.ac.uk/research/vision/projects/VIGOUR/

- ✓ **Integración perceptora**

"Las señales y los módulos perceptores integrados por VIGOUR son:

Color de la piel, movimiento, detección del rostro, modelando el cuerpo humano, extracción de rasgos, reconocimiento del gesto"[27]. En *figura 23(a)* se muestra el reconocimiento de rostros, color de la piel, movimiento y movimiento mas el color de la piel. En la *figura 23(b)* se tienen los mismos parámetros pero con el brazo y el rostro.

(a)



(b)

Figura 23. Reconocimiento humano

[27] Ibid., p.1

5.1.3 Detección de pupilas

"La *figura 24* da una apreciación global del sistema. El efecto del ojo rojo es una propiedad fisiológica del ojo y la primera parte interesante del uso para rastrear robustamente a las pupilas. Una vez las posiciones de las pupilas son conocidas, aquéllos se usan para normalizar las imágenes y parámetros del extracto que describen los rasgos faciales y pueden usarse para reconocer las acciones faciales. Finalmente, las unidades de acción faciales superiores que usan las Máquinas de soporte vectorial se reconocen. Una separación del capturador kernel está especializado para cada unidad de acción facial. Los parámetros extraídos se usan como los rasgos de la entrada al vector soporte que se mecaniza para descubrir ocurrencia de acciones faciales. Desde que se esta utilizando un separador que descubra ocurrencia de acciones faciales"[28].

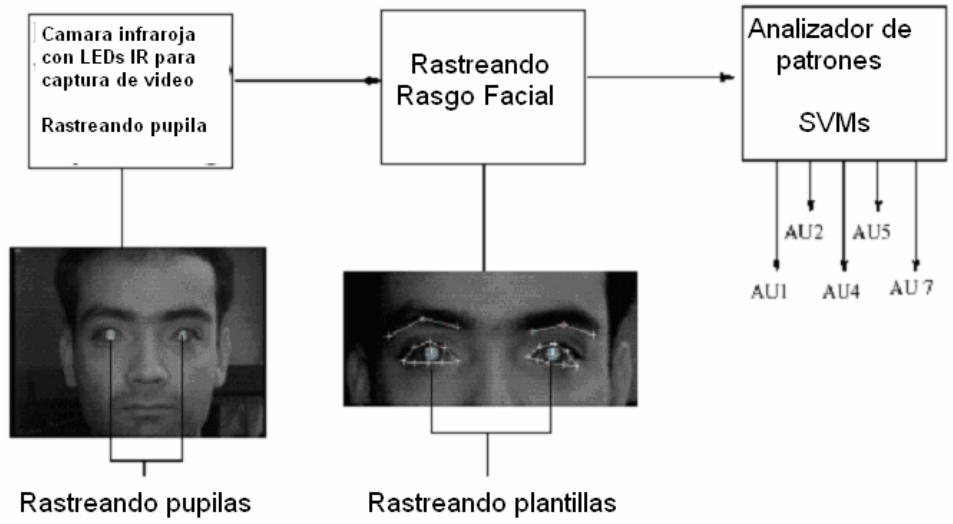


Figura 24. Sistema global

[28] Kapoor A., Qi Y., Picard R. (Octubre, 2003). Fully Automatic Upper Facial Action Recognition. Vismod. p.2. Obtenido de la red mundial el 20 Abril del 2005 : http://vismod.media.mit.edu/pub/tech-reports/TR-550.pdf

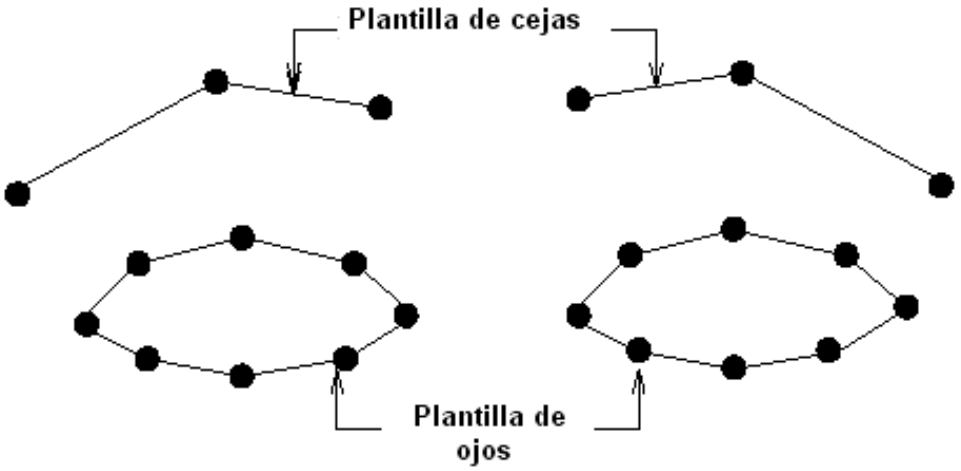


Figura 25. Plantillas de ojos y cejas

En la *figura 25* se muestra la estrategia de la simplificación de la aproximación usada por *[Covell et al].* La naturaleza no reiterativa de la aproximación lo hace ideal para ser usado en un sistema de tiempo real.

"El sistema de detección de pupila detecta las pupilas utilizando el efecto (Ojo-Rojo). La robustez del sistema a las oclusiones y realización de movimientos de cabeza es ideal para ser usado para el análisis de acción facial automática. Como las posiciones de la pupila pueden recuperarse muy eficazmente y robustamente, elimina la necesidad de etiquetado del manual o pre-procesado de las imágenes, un paso que opaca varios acercamientos anteriores que lo requerían"[29].

[29] Ibid., p.3

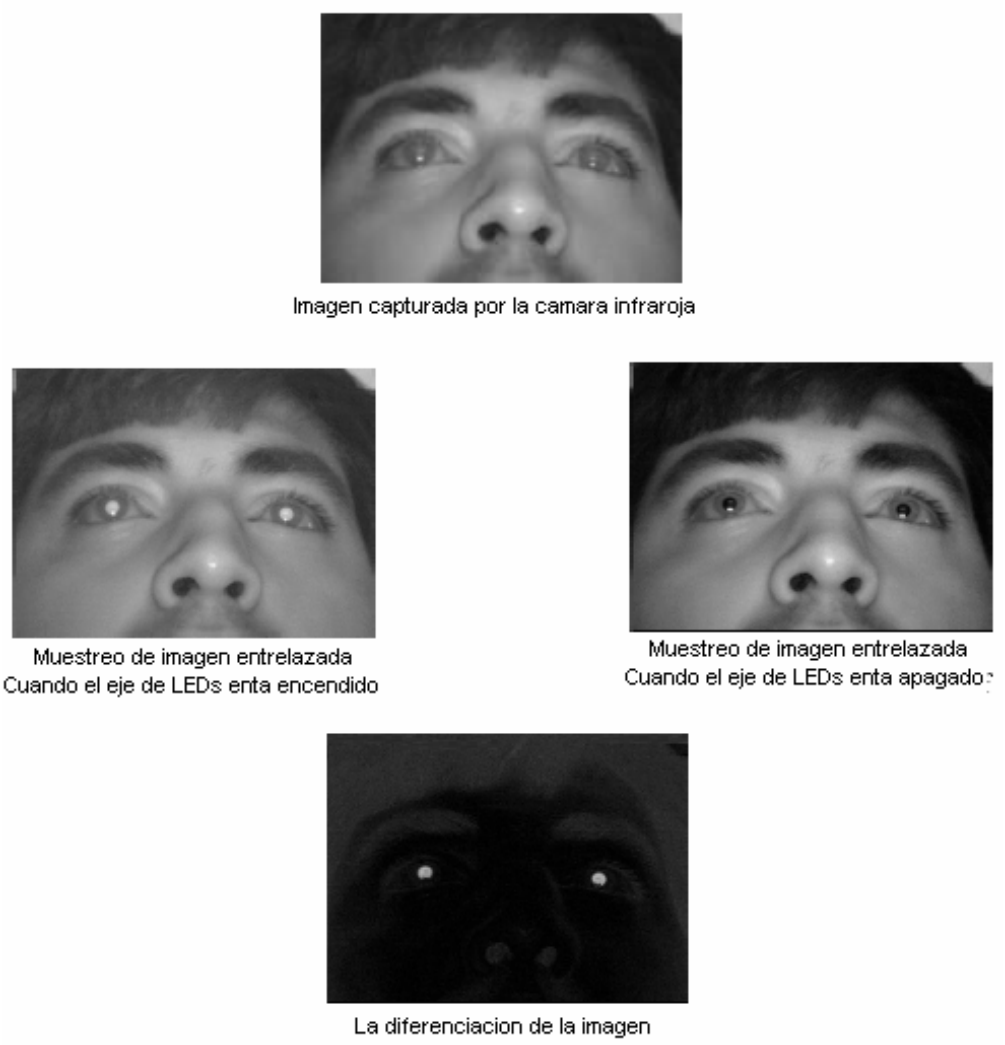


Figura 26. Rastreando pupila con cámara infrarroja

"Las pupilas son detectadas y rastreadas utilizando la diferenciación de imagen, que es la cuota de ruido que resulta al procesar la diferenciar las dos imagenes[*]. También, los objetos como las gafas y aretes pueden presentarse como manchas luminosas en la diferenciación de imagen debido a sus cualidades"[30].

[*] Las dos imagines capturadas son la que contiene el resultado del efecto ojo rojo y una en su estado normal

[30] Ibid., p.3

5.1.4 Autenticación de rostros

"Cualquier proceso de la autenticación involucra dos fases computacionales básicas. En la primera fase una representación conveniente se deriva con el objetivo múltiple de hacer el subsiguiente, la fase de decidir-hacer, computacionalmente factible, inmune a los cambios medioambientales durante la adquisición de datos biometricos, y eficaz sólo proporcionándole la información pertinente a la tarea de la autenticación. El propósito de la segunda fase es aceptar o rechazar la demanda de identidad que corresponde a una medida de prueba biométrica. Éste es básicamente uno de dos clases de problema de reconocimiento de patrones"[31].

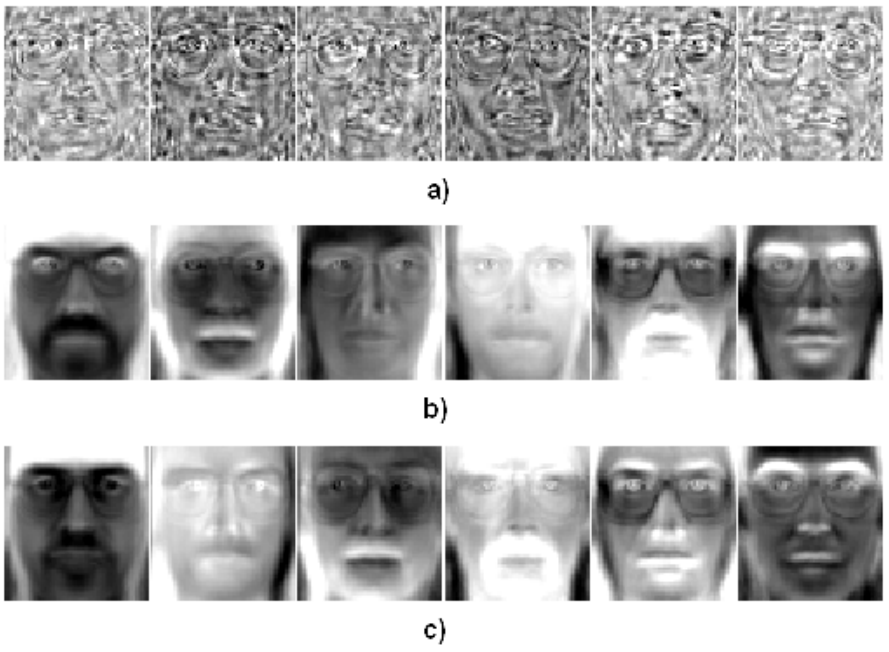


Figure 27. (a) (Captura de datos no normalizados) y (b-c) (los datos no normalizados y normalizados). En todos los tres casos, se muestra los primeros seis vectores de la base.

[31] Jonsson K., Kittler J., Li Y., Matas J. (Enero, 2001). Support Vector Machines for Face Authentication. Bmva. p.3. bmva. Obtenido de la red mundial el 16 Abril del 2005: www.bmva.ac.uk/bmvc/1999/papers/54.pdf

"Para comprender el procedimiento explicado se puede usar una base de datos de imágenes bidimensionales y en escala de grises. Las muestras con las que se va a entrenar la *SVM* corresponden a una base de datos de caras y no caras, que se descargan en la página Web del *MIT CBCL* («Center for Biological and Computational Learning»). En particular, la base de datos que se va a usar contiene *2.429* caras y *4.546* no caras, lo que hace un total de *6.975* muestras para entrenar y testear la *SVM*. Todos estos archivos se encuentran en formato .pgm («Portable Gray Map»). Las imágenes son de *19 x 19* píxeles, por lo que se trabajará con *19 x 19 = 361* dimensiones[32]".

"Estos archivos de imagen en formato *.pgm*, se pueden abrir por ejemplo con el programa «Fine View», que es un potente visualizador de imágenes que permite en particular este formato, a parte de muchos otros"[33].

Unos ejemplos de estas caras y no caras que se utilizan para entrenar la máquina de vectores soporte se muestran a continuación. Se cogen aleatoriamente el *80%* de los datos (*5.580* datos de los *6.975* totales), y utilizando un kernel gaussiano se obtiene el mejor valor de los hiperparámetros que definen este tipo de kernel"[34].

[32] Sánchez R. (12 de Julio, 2004). Sistema de reconocimiento facial. faq-mac. p.1. Obtenido de la red mundial el 18 Abril del 2005: www_faq-mac_com Sistemas de Reconocimiento Facial, por Raúl Sánchez Vítores.htm
[33] Ibid., p1
[34] Ibid., p1.

Figura 28. Rostros de entrenamiento

"Antes también se debe hacer una transformación, ya que las caras pueden ser de distinto tamaño, por lo que si son mayores de *19 x 19* se diezma y si son menores, se interpola (lo más aconsejable es utilizar el método de «interpolación bilineal»). Esto da una idea de la complejidad computacional del proceso. Los datos a probar se obtienen desplazándose píxel a píxel. Después, igual con las muestras de doble fila, hasta el final de su respectiva fila y de la misma forma hasta el final de la fotografía en cuestión"[35].

Como se puede ver en la *figura 29*, una de las caras no se detecta y luego tomando distancias de ojos, nariz y boca tampoco se reconocería comparándola con otra de la misma persona previamente almacenada en una base de datos. Esto está dentro del error de test obtenido y se debe a que la *SVM* no se ha entrenado con caras tan inclinadas, como se puede observar en la *figura 28*, donde todas las caras están de frente"[36].

[35] Ibid., p1.
[36] Ibid., p1.

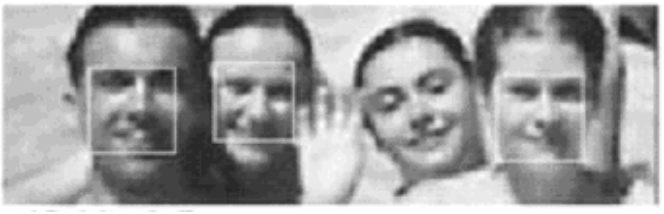


Figura 29. No reconocimiento de rostro inclinado

"Para resolver este tipo de problemas y después los de la fase de reconocimiento, la solución es utilizar la misma herramienta de la *SVM* pero para trabajar en tres dimensiones, como se explica en el apartado siguiente, donde se muestra uno de los mejores productos que están actualmente en el mercado"[37].

Figura 30. Rostros en tercera dimensión

"Neurodynamics ofrece su producto Tridenty, uno de los mejores sistemas de reconocimiento facial que existen en el mercado. Utiliza la información de la luz para poder recrear una imagen en tres dimensiones, con la cual se puede generar la estructura ósea de la cara y se puede independizar así de la sensibilidad a la posición del usuario en cada toma. Una última ventaja es que al construir el sistema una imagen tridimensional, la misma se puede rotar y ver desde distintos ángulos, aunque el usuario nunca haya sido capturado desde ese ángulo"[38].

[37] Ibid., p1.
[38] Ibid., p1.

5.1.5 Reconocimiento del habla

"Los acercamientos híbridos para el reconocimiento del habla proporciona un paradigma flexible para evaluar las nuevas técnicas acústicas modeladas. Estos sistemas no eliminan el armazón de *HMM* completamente porque los modelos de clasificación tal como las *SVMs* no pueden modelar la estructura temporal del habla eficazmente"[39].

"Un problema fundamental en el diseño del clasificador es si los clasificadores deben ser un clasificador contra otro clasificador, que aprende a diferenciar una clase de otra clase o uno clasificador -contra-todos, que aprende a diferenciar una clase de todas las otras clases. Los clasificadores uno contra uno son típicamente más pequeños y menos complejos y pueden estimarse usando menos recursos que los clasificadores uno - contra- todos. Cuando el número clases es *N*, se necesita estimar *N(N - 1)/2* clasificadores uno -contra- uno como comparado los clasificadores uno-contra-todos. En varias tareas de clasificación normal, ha sido probado que los clasificadores uno -contra- uno son marginalmente más exacto que los clasificadores uno-contra-todos. No obstante, para la eficacia computacional, se prefiere usar los clasificadores uno-contra-todos. Un ejemplo de ajuste para un clasificador típico se muestra en *figura 31*"[40].

[39] Ganapathiraju *A.*, Hamaker J., Picote j. (8 de Agosto, 2004). Applications of Support Vector Machines to Speech Recognition. ieeexplore.ieee p. 2. Obtenido de la red mundial el 25 Abril del 2005: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/78291 66/01315952 .pdf

[40] Ibid., p. 3.

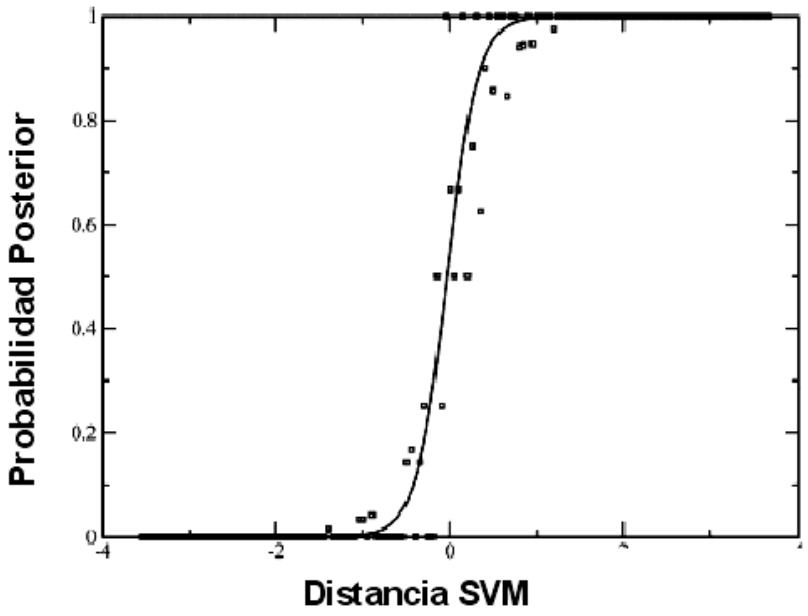


Figura 31. Sigmoidal de ajuste de la distancia SVM.

"En la *figura 32* Se muestra la construcción de un vector compuesto por un segmento telefónico. Los clasificadores *SVM* en nuestro sistema híbrido operan en los llamados vectores compuestos. La composición de los vectores de rasgo del segmento son basados en las alineaciones de un tercer estado básico, el Gaussiano mezclado con el sistema *HMM*[*]. La longitud del vector compuesto es dependiente en el número de secciones en cada segmento y la dimensión de la trama de los vectores del rasgo nivelados. Por ejemplo, con un vector del rasgo dimensional *39* al nivel de la trama"[41].

[*] Hidden Markov Models (Modelo de Cadenas de Markov)

[41] Ibid., p. 3.

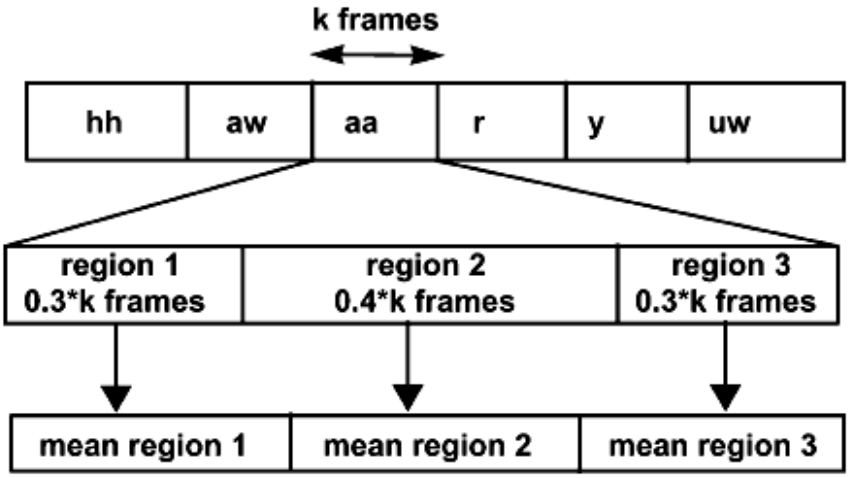


Figura 32. La composición del segmento de vector de rasgo nivelado que asume una proporción 3-4-3 para las tres secciones.

5.1.6 Categorización de texto

"La categorización del texto es el proceso de ordenar los documentos del texto en uno o más categorías predefinidas o documentos de clases similares. Las diferencias en los resultados de la categorización resultan del rasgo escogido, al basarse en la asociación de un documento dado con una categoría dada. Los defensores de la categorización del texto reconocen que el orden de los documentos de texto en las categorías así como los documentos deseados reducen la sobrecarga requerida para la recuperación rápida de documentos semejantes y proporciona dominios más pequeños en que los usuarios pueden explorar los documentos similares"[42].

[42] Basu A., Watters C., Shepherd M. (2003). Support Vector Machines for Text Categorization. hicss.hawaii. p. 1. Obtenido de la red mundial el 25 Abril del 2005: http://www.hicss.hawaii.edu/HICSS36/HICSSpapers/DDDLS03.pdf

"La categorización puede ser basada en el juicio humano, como se hace por Yahoo, el simple agrupamiento de la palabra clave, o el algoritmo de aprendizaje.

La categorización del texto requiere, como una base, la identificación de rasgos dentro de los documentos que pueden usarse al indiscriminar entre los documentos y la asociación de documentos individuales a categorías individuales. Estas categorías pueden determinarse a priori, o por humanos o algorítmicamente, o puede determinarse dinámicamente como requerido"[43]. Este tipo de categorización también es realizado por otros tipos de máquinas de aprendizaje como redes neuronales.

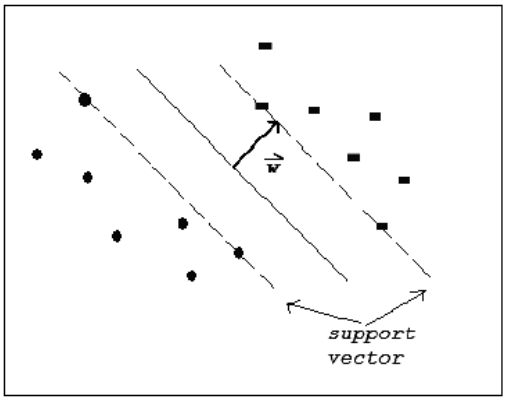


Figura 33. Muestra vectores soporte de un conjunto de categorización cualquiera.

5.1.7 Localización exacta de la falla en la línea de transmisión de Potencia usando aproximación SVM

"Se considerara una línea de transmisión de potencia que conecta dos sistemas *A* y *B* como se muestra en la *figura 34*. Se asume la falla de la línea, como la resistencia que se pone en cortocircuito de valor desconocido, ocurriendo en

[43] Ibid., p. 1.

cualquier parte a una distancia desconocida. Los tipos diferentes de fallas son considerados, incluso la fase a conecte con tierra *(R-g, S-g, T-g)*, fase a fase *(R-S, S-T, R-T),* dos fases a conecte con tierra *(R-S-g, S-T-g, R-T-g)*, y tres fase *(R-S-T)*. El sistema de potencia ha sido modelado usando el programa *ATP-EMTP*. Las fallas han sido simuladas aplicando la estructura del circuito universal (las resistencias e interruptores) como se presenta en la *figura 34*. Todas las resistencias puestas en corto circuito son de valores iguales, las cuales se les puede cambiar el valor en la simulación dependiendo de las suposiciones reales"[44].

"En la figura 35 se presentan tres ejemplos de fase de los voltajes transitorios y corrientes de *200-km 400-kV* de la línea de transmisión transpuesta (modelo de Clark) bajo *R-g* fase a tierra, la falla ocurre en dos lugares diferentes. Estos transitorios se han obtenido usado el programa *ATP-EMTP*. La resistencia puesta en cortocircuito asumida en los experimentos era igual. Ambos sistemas usados en las simulaciones eran débiles. *Figura 35(a)* y *(b)* corresponda a la falla que ocurre a la distancia de *1 km* del sistema *A* y *figura 35(c)* y *(d)—199 km* del sistema *A*. que se ve que el nivel de corriente de fase es malo así como los componentes armónicos de ambas corrientes y cambio brusco de voltaje con la distancia de falla"[45].

[44] Salat R., Osowski S. (2 de Mayo, 2004). Accurate Fault Location in the Power Transmission Line Using Support Vector Machine Approach. ieeexplore.ieee p. 1. Obtenido de la red mundial el 25 Abril del 2005: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/ 78/29166/01315952.pdf

[45] Ibid., p. 1.

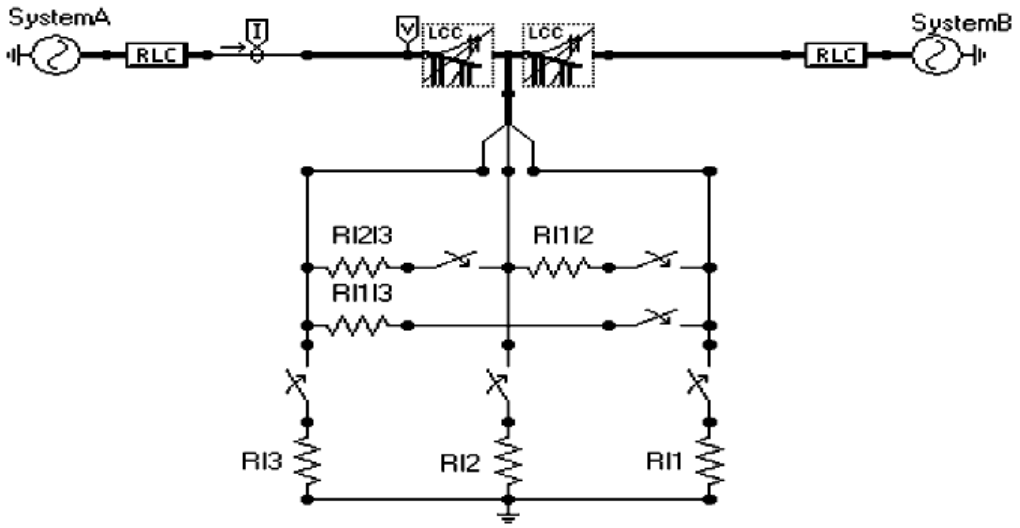


Figura 34. EMTP Modelando la falla de la línea que conecta dos sistemas A y B.

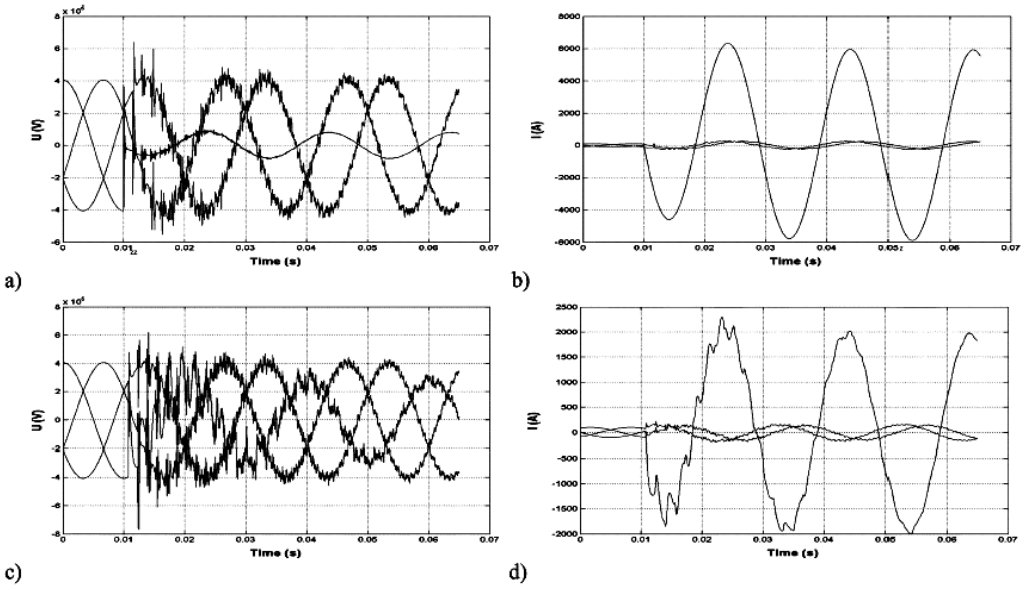


Figura 35. Los voltajes de los transitorios y corrientes midieron al lado de la entrada de la línea a la falla que ocurre en el origen (a y b) y fin (c y d) de la línea.

5.2 Programas para SVM

En los capítulos anteriores se habló sobre los entrenamientos que deben recibir las *SVM,* pues en esta sección se observará como se entrena una *SVM* mediante un programa, de manera didáctica y constructiva para que todo los interesados en estas máquinas puedan comprender la complejidad de estas, teniendo en cuenta que la mayoría de los programas son libres y que se pueden conseguir única y exclusivamente para fines demostrativos y didácticos.

Existen diversos programas para *SVM* entre los cuales tenemos: *svmclassify, svmtrain, svm-km, matlab (toolbox svm), svmstruct, svmdark, winsvm y svmlight* a estos tres últimos se dará una breve descripción y ejemplos.

5.2.1 WinSVM

WinSVM es un software muy censillo con interfase para Windows y programado en *C*. "Este software puede realizar reconocimiento de patrones y regresión, el ejemplo siguiente utiliza la regresión.

A continuación se presenta un ejercicio de aprendizaje dirigido, se entregan los siguientes datos de input/output entradas y salidas. Se dan las entradas futuras, se desea predecir las salidas (desconocidas)"[46].

[46] Sewell M (25 de Junio, 2005). winSVM. cs.ucl.ac.uk. p.1. Obtenido de la red mundial el 25 Abril del 2005 : l, http://www.cs.ucl.ac.uk/staff/M.Sewell/winsvm/

input 1	input 2	output
7	0	7
9	1	10
4	6	10
3	8	11
9	4	13
3	8	11
5	4	9
6	0	6
3	8	11
4	5	9
0	1	1
3	8	11
9	0	9
3	0	3
2	9	11
1	4	5
7	9	16
0	9	9
1	8	9
2	3	5

Tabla 5. Entradas y salida programada para entrenamiento "wins vm"

- "Divida los datos en tres conjuntos (en la proporción *50%, 25%, 25%*).

Guarde el conjunto de entrenamiento como train.txt en el formato siguiente:

@parameters

training set

@examples

format xy

7 0 7

9 1 10

4 6 10

3 8 11

9 4 13

3 8 11

5 4 9

6 0 6

3 8 11

4 5 9

validation set
@examples
format xy
0 1 1
3 8 11
9 0 9
3 0 3
2 9 11"[49]

- "Haga Click en "*Input file*..."" y seleccione *'validation.txt.*'
- Haga Click en "*Predict*".

Figura 39. Muestra la respuesta de winSVM después del proceso de predicción correcto.

- Inspeccione *'validation-pred.txt'*.
- Continúe ajustando los parámetros de Kernel, *'Predict'*, inspeccione 'validación-pred.txt' hasta que los resultados sean más cercanos al actual conjunto de validación. En este caso, se trabaja con un Kernel de punto *"Dot"*. Copie los conjuntos-entrenamiento-con-*alphas* y la prueba colóquelas en un nuevo archivo llamado *'test.txt'*

[49] Ibid., p1.

- Haga clic en "*Input File*... " y seleccione *'train.txt'*
- Entre "*100*" para el número de ejecuciones.
- Haga click "*Optimize*."[47]

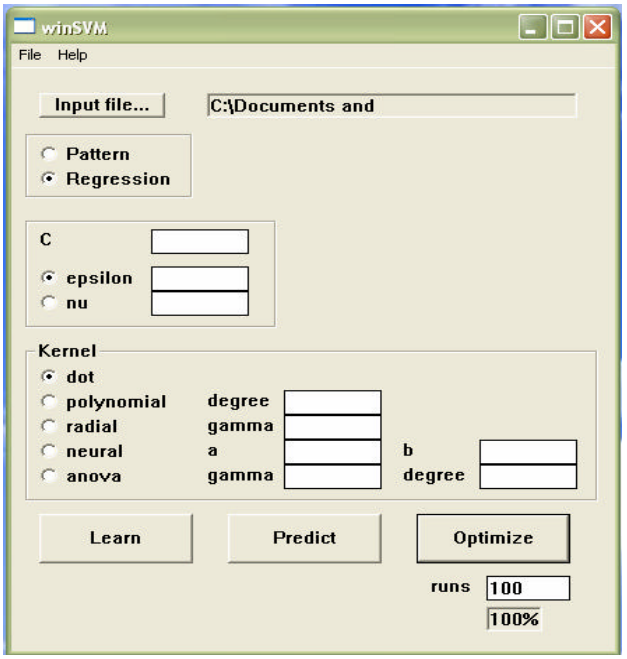


Figura 36. Muestra la ventana winSVM después de realizado correctamente la optimización.

- "Inspeccione *Input-opt.csv*, cual de los parámetros producen una error medio cuadrático bajo (*MSE*).
- Con el ejemplo dado, *epsilon* con un valor bajo y un kernel de tipo *Dot* parecen ser óptimos. Así para este conjunto de datos particulares, entre cualquier valor para *C* pequeño (sea, *1*), un valor bajo para *epsilon* (sea, *0.01*) y selecciona el kernel de tipo *Dot* en la interfaz del usuario"[48].

[47] Ibid., p1.
[48] Ibid., p1.

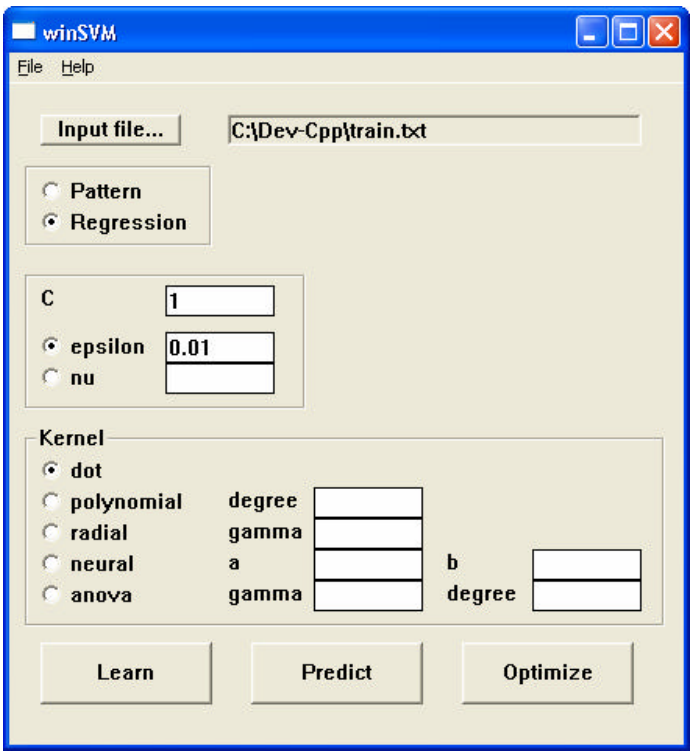


Figura 37. Muestra la ventana de winSVM después de introducir los datos parar el aprendizaje.

- Haga Click en '*Learn*'

Figura 38. Muestra la respuesta de winSVM después del aprendizaje correcto.

- "Abra *'train-svm.txt'*
- Contiene el entrenamiento puesto con los *alpha*-valores. Copie el contenido más el conjunto de validación en un nuevo archivo y guárdelo como *'validation.txt'*:

@parameters

training set

@examples

dimension 2

number 10

b -0.00224081523342617

format xya

7 7.676151381197371e-017 7 -0.1797849206983427

9 1 10 0.06802209641651072

4 6 10 0

3 8 11 0

9 4 13 1

3 8 11 0.0916387839347123

5 4 9 0

6 7.676151381197371e-017 6 -1

3 8 11 0.02012404034711963

4 5 9 0

@parameters
training set
@examples
dimension 2
number 10
b -0.00224081523342617
format xya
7 7.676151381197371e-017 7 -0.1797849206983427
9 1 10 0.06802209641651072
4 6 10 0
3 8 11 0
9 4 13 1
3 8 11 0.0916387839347123
5 4 9 0
6 7.676151381197371e-017 6 -1
3 8 11 0.02012404034711963
4 5 9 0"[50]

"# test set
@examples
format xy
1 4 5
7 9 16
0 9 9
1 8 9
2 3 5

- Haga Click en *"Predict".*
- Inspeccione el archivo *'test-pred.txt':*

[50] Ibid., p1.

examples ys
5.00009
15.9998
9.00001
9
5.00008

- Renombre las salidas actuales:

5
16
9
9
5"[51]

5.2.2 SVM*light*

"*SVM*^{light} es una implementación de Máquinas de soporte Vectorial en *C*.

Las características principales del programa son lo siguiente:

- Algoritmo de optimización rápido
- Selección del conjunto de trabajo en pasos de descenso factible
- "Encogimiento" heurístico
- Escondimiento de evaluaciones de kernel
- Uso de rendimiento en el caso lineal
- Resuelve clasificación y problemas de la regresión. Para multivariable y salidas estructuradas se utiliza *SVM*^{struct}.
- Resuelve los problemas de la clasificación jerárquica.
- *XiAlpha-stimate* computa la tasa de error, la precisión, y la recuperación.
- Eficazmente computa un permiso por salida de las estimaciones de la tasa de error, la precisión, y la recuperación.

[51] Ibid., p1.

- Incluye un algoritmo para aproximaciones de entrenamientos grandes de transducciones *SVMs (TSVMs).*
- Puede entrenar *SVMs* con modelos del coste y ejemplo de costes dependientes.
- Permite reiniciar el vector especificado de variables duales.
- Maneja muchos miles de vectores de apoyo.
- Maneja algunos cientos de miles de ejemplos de entrenamiento
- Soportes estándar de la función de kernel y le permite definir los suyos.
- Utiliza representaciones de esparcimiento de vectores."[52]

"SVM^{light} es una aplicación de *[Vapnik] support vector machine [Vapnik, 1995]* para el problema de reconocimiento de patrones, para el problema de regresión, y para el problema de aprendizaje de una función de clasificación jerárquica. Los algoritmos de optimización usados en *SVM^{light}* se describen en *[Joachims, 2002a]. [Joachims, 1999a].* El algoritmo tiene los requisitos de memoria escalables y puede ocuparse de problemas eficazmente con muchos miles de vectores soporte"[53].

"El software también mantiene los métodos evaluando eficazmente la ejecución de la generalización. Incluye dos métodos de estimación eficaces para la taza de error y precisión / recuperación"[54].

"Lo nuevo en esta versión es un algoritmo para aprendizaje de funciones de clasificación jerárquica *[Joachims, 2002c].* La meta es aprender una función de

[52] Joachims T.(2 de septiembre, 2004). SVM^{light} Support Vector Machine. Svmlight.joachims. p. 1. Obtenido de la red mundial el 14 Abril del 2005 : http://svmlight.joachims.org
[53] Ibid., p.1
[54] Ibid., p.1

los ejemplos de la preferencia, para que pida un nuevo conjunto de objetos tan precisos como sea posible. Los problemas de la clasificación jerárquica ocurren naturalmente en las aplicaciones como los artefactos de la búsqueda y sistemas recomendados"[55].

"*SVM*^{light} también puede entrenar *SVMs* con modelos de coste. El código se ha usado en un rango grande de problemas, incluso la clasificación del texto *[Joachims, 1999c][Joachims, 1998a],* tareas de reconocimiento de imagen, bioinformática y aplicaciones médicas.

Muchas tareas tienen la propiedad de vectores del caso esparcido. Esta aplicación hace uso de esta propiedad que lleva a una representación muy compacta y eficaz"[56].

- ✓ **Datos de entrenamiento:**

"Pueden cargarse los datos de entrenamiento de un archivo o pueden agregarse usando el API de *jSVM.* La estructura de los datos es creada por *jSVM* para guardar los datos de entrenamiento. Cuando se llama el *buildModels (),* los datos de entrenamiento se envían por SVM^{light}. SVM^{light} crea e inicializa la los datos estructurados para los datos de entrenamiento. Entonces SVM^{light} realiza el algoritmo de aprendizaje sobre un conjunto de datos de generalización de clasificador de modelos, los cuales son guardados en SVM^{light}. Después del proceso de aprendizaje, los datos de entrenamiento se eliminan de SVM^{light} *t"*[57].

[55] Ibid., p.1
[56] Ibid., p.1
[57] Joachim T. (26 de Marzo, 1998). Transfer Protocol. cad.eecs berkeley. p.1. Obtenido de la red mundial el 25 de Mayo del

"Se pueden agregar los datos de entrenamiento al *jSVM,* los nuevos datos no serán visibles a SVM*light* hasta llamar los próximos *buildModels ().* Si los modelos ya existen en SVM*light*, ellos deben quitarse antes de que los nuevos modelos puedan construirse. Esto puede hacerse llamando *clearModels ().* Entonces cuando se llama *buildModels ()* , se enviarán los datos de entrenamiento en el *jSVM* a *SVMlight,* y el resto sigue el mismo procedimiento como fue expresado arriba"[58].

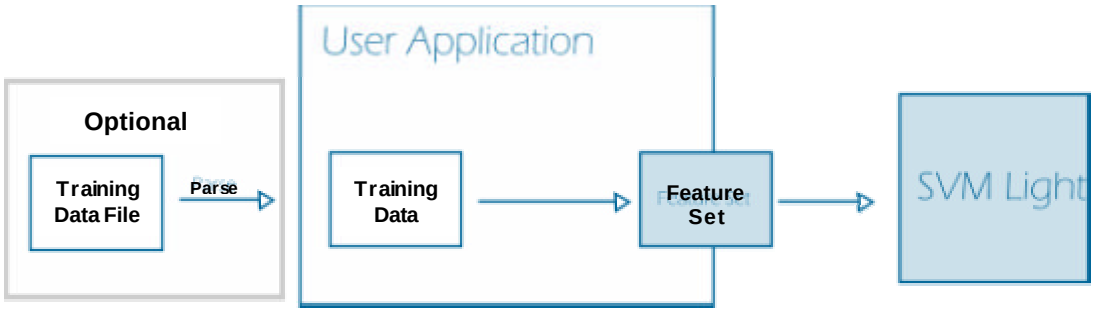


Figura 40. Aprendizaje SVM*light*

- ✓ **Modelos:**

"Después del aprendizaje, SVM*light* produce el modelo de la estructura de los datos. Los modelos también pueden cargarse en un archivo a través del *jSVM*, En cualquier caso, el modelo de información se envía a SVM*light* como un byte de flujo y se analiza entonces y se construye una estructura de datos. *jSVM* no necesita saber de los modelos, porque el cómputo se hace en SVM*light*. Si modelos serán

2005 : http://www-cad.eecs.berkeley.edu/~hwawen/research/projects/jsvm/doc/manual/jsvm-transfer-protocol.html

[58] Ibid., p.1

escritos fuera de un archivo, SVM*light* presentará a los modelos como un byte de flujo, enviándolo por la de *jSVM* que entonces manda el un archivo de salida"[59].

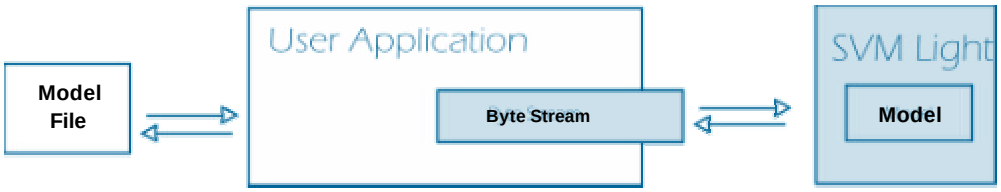


Figura 41. Modelos para *SVMlight*

- ✓ **Casos de Test (Prueba):**

"Los casos de test son manejados similarmente a los datos de entrenamiento. Ellos o pueden cargarse en un archivo o pueden agregarse a través del *API* del *jSVM.* Durante la clasificación, *jSVM* pasa un caso de test en un tiempo a *SVMlight* para ser clasificado. El resultado de la clasificación se envía antes al *jSVM* y se guarda en un del dato de estructura"[60].

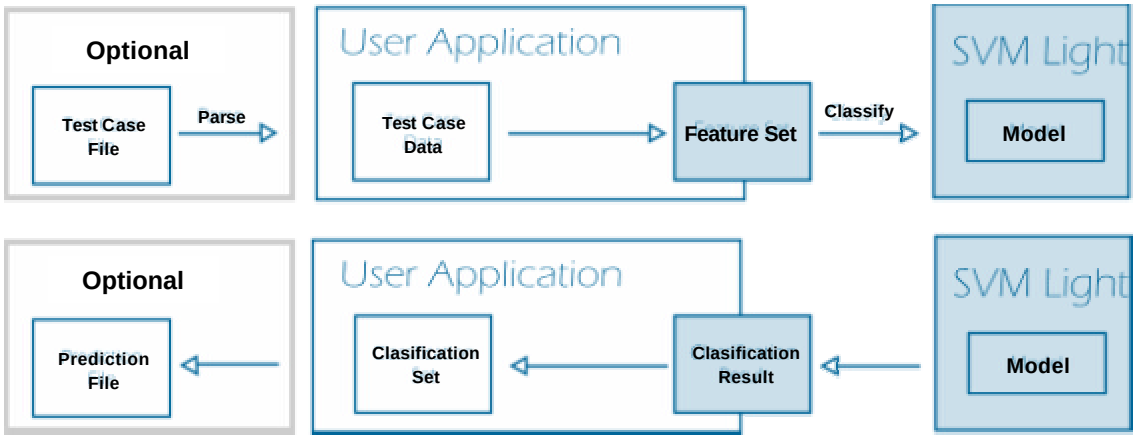


Figura 42. Casos de prueba de *SVMlight*.

[59] Ibid., p.1
[60] Ibid., p.1

5.2.3 SVMdark

El software puede realizar la clasificación y la regresión, El siguiente ejemplo utiliza regresión. Para la clasificación, substituya las salidas por "1 o -1". Cualquier espacio dejado en blanco en la interfaz se tratarán como un ceros. A continuación se presenta un ejercicio de aprendizaje dirigido, se entregan los siguientes datos de input/output entradas y salidas. Se dan las entradas futuras, se desea predecir las salidas (desconocidas)"[61].

input 1	input 2	output
7	0	7
9	1	10
4	6	10
3	8	11
9	4	13
3	8	11
5	4	9
6	0	6
3	8	11
4	5	9
0	1	1
3	8	11
9	0	9
3	0	3
2	9	11
1	4	5
7	9	16
0	9	9
1	8	9
2	3	5

Tabla 6. Entradas y salida programada para entrenamiento "svmdark"

[61] Sewell M (25 de Junio, 2005). winSVM. cs.ucl.ac.uk. p.1. Obtenido de la red mundial el 25 Abril del 2005 : l, http://www.cs.ucl.ac.uk/staff/M.Sewell/svmdark/

"Divida los datos en tres sistemas (en el cociente el 50%, el 25%, el 25%). En cada sistema, haga la primera columna la columna de la salida y etiquete las otras 1:, 2:, etc. Excepto como tres archivos separados del texto, como sigue"[62].

Sistema del entrenamiento (train.txt)

7 1:7 2:0

10 1:9 2:1

10 1:4 2:6

11 1:3 2:8

13 1:9 2:4

11 1:3 2:8

9 1:5 2:4

6 1:6 2:0

11 1:3 2:8

9 1:4 2:5

Sistema de la validación (validation.txt)

1 1:0 2:1

11 1:3 2:8

9 1:9 2:0

3 1:3 2:0

11 1:2 2:9

Pruebe el sistema (test.txt)

5 1:1 2:4

16 1:7 2:9

9 1:0 2:9

9 1:1 2:8

5 1:2 2:3

[62] Ibid., p.1.

Seleccione "Regresión".

”Haga click "el botón del archivo de la prueba..." y seleccione validation.txt.

Complete los otros campos según lo demostrado abajo”[63].

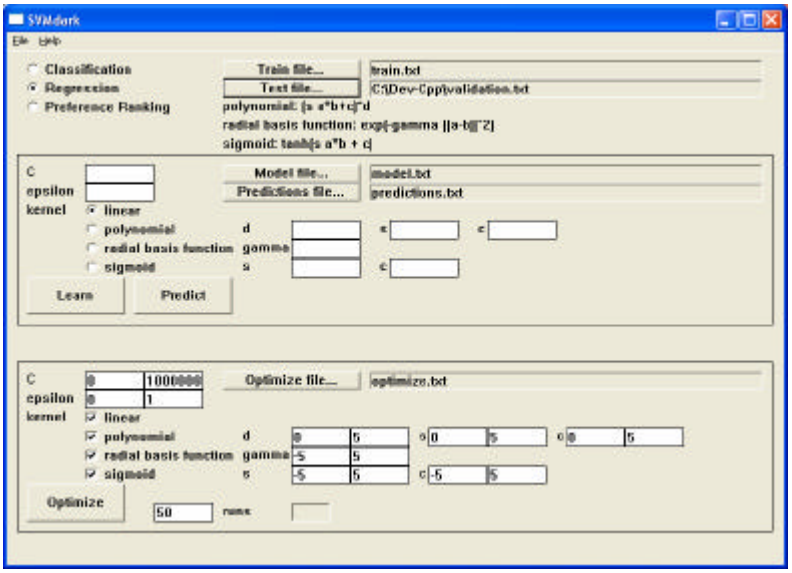


Figura 43. Optimización

Haga click en el botón "optimize".

“Transcurrido el tiempo, abra el archivo "optimize.csv". Observe los parámetros usados que producen un MSE bajo (error medio cuadratico). Reduzca el número de modelos potenciales "enfocando" en los que se realicen bien en el sistema de la validación, repiten en caso de necesidad. En este ejemplo, el núcleo lineal con un valor pequeño para la épsilon se realizó lo mejor posible, así que reducimos nuestra búsqueda por consiguiente:

Una vez más examine "optimize.csv" e identifique el MSE más pequeño. Seleccionamos los parámetros que se realizaron óptimo en el sistema de la

[63] Ibid., p.1.

validación. En este ejemplo, el núcleo lineal con un valor pequeño para épsilon funciona lo mejor posible, así que se reduce la búsqueda por consiguiente:"[64].

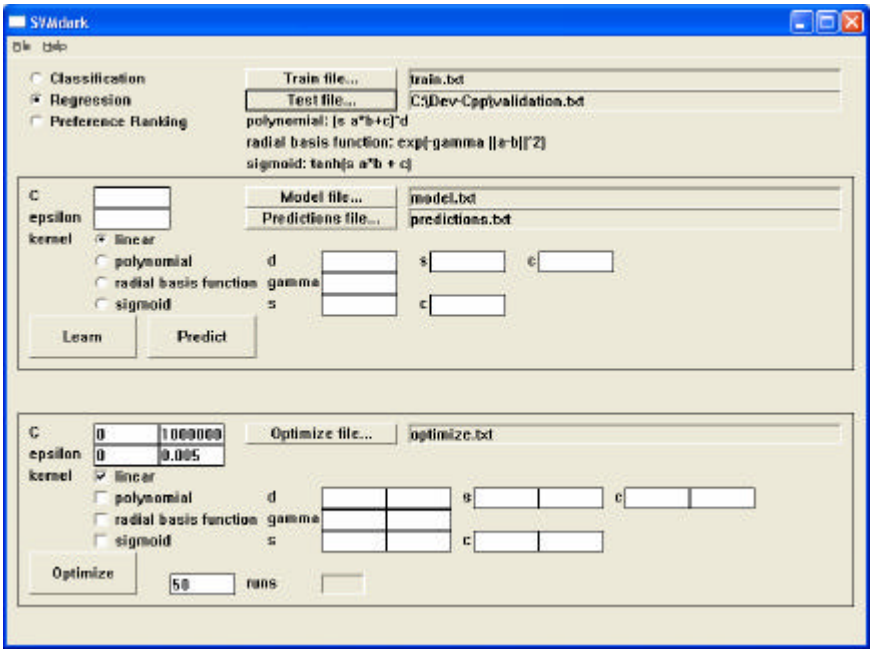


Figura 44. Validación

"Una vez más examine "optimize.csv" e identifique el MSE más pequeño. Seleccionamos los parámetros que se realizaron óptimo en el sistema de la validación. Ahora, seleccionamos la prueba para fijar – Haga click en el boton "Test file..." otra vez y seleccione test.txt"[65].

[64] Ibid., p.1.
[65] Ibid., p.1.

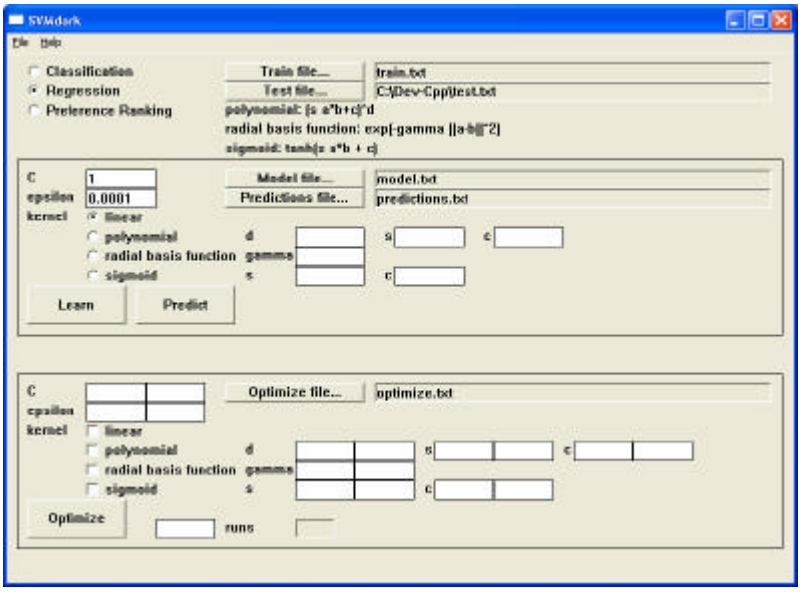


Figura 44. Prueba

Haga click en el botón "Learn".

Haga click en el botón "Predict".

Inspeccione el archivo "predictions.txt".

Predicciones:

5,0006471

15,999537

9,0003896

9,0003437

5,0006011

Recuerde las salidas reales:

5

16

9

9

5

5.2.4 Otros programas direcciones URL

- ✓ ***Svmclassify*:** http://www.mathworks.com/access/helpdesk/help/toolbox/bioinfo/ug/svmclassify.html
- ✓ ***Svmtrain*:** http://www.mathworks.com/access/helpdesk/help/toolbox/bioinfo/ug /svmtrain.html
- ✓ ***Svm-km:*** http://www.rpi.edu/~kunapg/mmld/
- ✓ ***Matlab (toolboxsvm):*** http://www.support-vector.ws/html/downloads.html, http://asi.insa-rouen.fr/~arakotom/toolbox/index
- ✓ ***Svmstruct:*** http://svmlight.joachims.org

CONCLUSIONES

El algoritmo *K-SVCR* permite integrar los esquemas de descomposición *1-v-1* y *1-v-r* superando algunas de sus desventajas iniciales y este es más eficiente que una *SVMR* con tres salidas gracias a su menor demanda en las restricciones del problema *QP* asociado.

Al evaluar las características de la *K-SVCR* con otras máquinas de aprendizaje se pudo observar que los algoritmos utilizados para las *K-SVCR* siempre llegan a un mínimo global que da solución al conjunto de aprendizaje y que son menos complejos, en cuanto a la estadística que utiliza para su desarrollo y la hace mas eficaces y eficientes.

Las *SVM* poseen más afinidad con las redes neuronales que con los algoritmos genéticos por la forma de aprendizaje y su estructura.

Mediante las aplicaciones de las máquinas *SVM* se profundiza sobre como son utilizadas estas en el ámbito científico e industrial para satisfacer las necesidades de las personas.

Con el software winsvm, el cual es muy fácil de manejar, se practicó haciendo prueba para comprender como se optimiza, se enseña y se predice al trabajar con una máquina de soporte vectorial.

RECOMENDACIONES

Debido a que este es un tema nuevo e innovador sugerimos que se abra un grupo de investigación para profundizar más sobre este tema ya que estas máquinas tienen muchas aplicaciones interesantes como las que se trataron en esta monografía las cuales valen la pena investigar y por que no, implementar y así llevar a nuestra institución a otro nivel investigativo.

Siendo este un tema de tanto interés para el programa de ingeniería electrónica se debe fomentar este por medio de la realización de conferencias, foros etc., y dotación de bibliografía a la biblioteca para el estudio del mismo.

Proyectos UTB:

Sistema de reconocimiento de huellas. La identificación de la huella digital es un problema de reconocimiento de patrones multi-clase se propone implementar esta para la presentación de exámenes, préstamo de libros, herramientas de laboratorio y equipos del departamento de audiovisuales, y otras aplicaciones similares que requieran identificar un responsable directo.

Sistema de reconocimiento de patologías para la determinación de enfermedades por medio de *SVM*. Este es un problema de reconocimiento de patrones multi-clase estrechamente ligada con electromedicina y para fines lucrativos, se propone desarrollar estas maquinas para determinar enfermedades que tengan características reflejadas en órganos externos como ojos y piel.