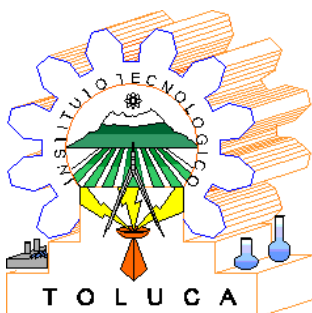




**S.E.P.      S.E.S.T.      D.G.E.S.T.**

---

***INSTITUTO TECNOLÓGICO DE***

**División de Estudios de Postgrado e Investigación**

**Título del Programa Académico:**

***Implementación de Opción de Rechazo y  
Aprendizaje Incremental, Empleando  
Support Vector Machine y Redes  
Neuronales Modulares  
(Reporte Final)***

**Que presenta:**

**Dr. Eduardo Gasca Alvarez**

Metepec, Estado de México.

Febrero 28 del 2011.

**Reporte Final del Año Sabático  
Presentado por el Dr. Eduardo Gasca Alvarez  
Instituto Tecnológico de Toluca**

Durante el período comprendido entre el 28 de enero de 2008 y el 27 de enero de 2009 llevé a cabo las siguientes actividades.

**Procedimiento de Opción de Rechazo.**

Diseñé el procedimiento de la Opción de Rechazo, descrito en el artículo Anexo A, y realicé los experimentos para estimar el error del Perceptron Multicapa (Backpropagation) con la Opción de Rechazo implementada. En el procedimiento de rechazo se evalúa la posición de los patrones con relación a dos hiperplanos que simulan la frontera de decisión de las dos clases más cercanas. En una primera aproximación se consideraron hiperplanos paralelos, sin embargo, en rigor éstos no son paralelos, lo cual dio origen a una modificación del procedimiento original. Comparé los resultados obtenidos con la red neuronal artificial Perceptron Multicapa sin la Opción de Rechazo, el Perceptron Multicapa con Opción de Rechazo sin cruce de hiperplanos, Perceptron Multicapa con Opción de Rechazo y cruce de hiperplanos, y las dos formas de calcular la distancia de un patrón a una clase.

El poder de generalización se calculó mediante validación cruzada, con 10 repeticiones, para doce diferentes bases de datos. El resumen y análisis están contenidos en el artículo del Anexo A.

El artículo se presentó a evaluación y será publicado en Lecture Notes in Computer Science 6593, pp. 51-60, 2011, "Adaptive and Natural Computing Algorithms".

**Procedimiento de Aprendizaje Incremental.**

Diseñé el procedimiento del Aprendizaje Incremental descrito en el documento del Anexo B y realicé los experimentos para estimar el error del clasificador híbrido que se formó con el Perceptron Multicapa y Máquinas de Soporte Vectorial.

Cabe mencionar que las Máquinas de Soporte Vectorial (SVM por sus siglas en idioma Inglés), son clasificadores cuyo comportamiento es óptimo cuando la tarea de clasificación contiene dos clases, linealmente separables, ya que para simular la frontera de decisión emplean hiperplanos. Sin embargo, las bases de datos usadas en los experimentos contienen más de dos clases, y en la mayoría de los casos son no linealmente separables. Esto generó la necesidad de modificar las bases de datos originales en dos sentidos:

- a) Para realizar el aprendizaje incremental se eliminaron todos los patrones de una o dos clases, posteriormente éstos se emplearon para realizar el aprendizaje incremental.
- b) La simulación de las clases linealmente separables se llevó a cabo entrenando un Perceptron Multicapa, los patrones bien clasificados fueron separados para integrar las muestras de entrenamiento para las Máquinas de Soporte Vectorial.

A grandes rasgos el algoritmo de Aprendizaje Incremental es el siguiente:

1. Se crean tantas SVMs como clases tenga la tarea de aprendizaje.

2. La muestra de entrenamiento para cada SVM debe contener dos clases, la primera se integra con los patrones de una de las clases, la segunda se forma con los elementos de las clases restantes. Se deben emplear los patrones que simulan las clases linealmente separables. Este paso pretende crear máquinas expertas en identificar una de las clases en particular.
3. Una vez entrenadas las máquinas de soporte vectorial se crea un patrón de transición 1 con los dos valores de salida de cada una de las SVM, esto es la cantidad de características será igual a dos veces la cantidad de máquinas de soporte vectorial
4. El patrón de transición 1 se emplea para entrenar un Perceptron Multicapa. Cuya capa de entrada tiene el número de nodos igual a dos veces la cantidad de clases, la capa de salida tiene tantos nodos como el número de clases.
5. Para integrar una nueva clase (aprendizaje incremental) se adiciona una máquina de soporte vectorial la cual se entrena con los patrones desconocidos y una muestra editada de los patrones ya conocidos por el sistema.
6. Las salidas de la maquina de soporte vectorial se agregan a las salidas del Perceptron Multicapa para crear el patrón de transición 2, el cual sirve para entrenar un Perceptron Multicapa 2 cuya capa de salida contendrá un nodo adicional que integra la nueva clase.
7. Para adicionar otra clase se repiten los pasos 5-6, tantas veces como sea necesario.

El poder de generalización se calculó mediante validación cruzada, con 10 repeticiones, para diez diferentes bases de datos, con una cantidad de clases mayor o igual a 2. Cabe mencionar que los resultados obtenidos mostraron que el porcentaje de eficiencia del aprendizaje incremental no fue mayor al 50% del obtenido entrenando un Perceptron Multicapa con todas las clases.

Dada la baja eficiencia del procedimiento de Aprendizaje Incremental descrito, no ha sido posible someterlo a evaluación para su publicación. Sigo trabajando en la mejora del algoritmo

Atentamente

Dr. Eduardo Gasca Alvarez  
Profesor-Investigador

# Anexo A

# **Anexo B**

# **Aprendizaje Incremental Empleando Perceptron Multicapa y Máquinas de Soporte Vectorial**

Eduardo Gasca Alvarez

Instituto Tecnológico de Toluca

## **Resumen**

Cómo adquirir nuevo conocimiento de la integración de datos de entrenamiento, adquiridos recientemente, conservando los conocimientos aprendidos con anterioridad es un problema importante para el aprendizaje incremental. Para manejar este problema, proponemos un algoritmo nuevo que integra el funcionamiento de la red neuronal artificial Perceptron Multicapa y las Máquinas de Soporte Vectorial.

El Perceptron Multicapa es una de las redes neuronales más estudiadas, con una amplia gama de aplicación, sin embargo salvo que se realice un reentrenamiento, carece de un procedimiento para adquirir nuevo conocimiento. Las Máquinas de Soporte Vectorial (SVM por sus siglas en inglés) se han convertido en una herramienta popular para el aprendizaje de grandes cantidades de datos con alta dimensionalidad. No obstante, bajo ciertas circunstancias puede ser preferible aprender incrementalmente a partir de los resultados de SVM previos, como por ejemplo; cuando el cálculo de las SVM sea muy costoso en cuanto al tiempo y consumo de memoria, o porque el SVM puede ser usado para realizar aprendizaje en línea.

Se pretende que la combinación del Perceptron Multicapa y las Máquinas de Soporte Vectorial tenga una mayor eficiencia y permita integrar nuevos datos, incluyendo muestras que correspondan a clases no vistas con anterioridad, conservando los conocimientos adquiridos. El algoritmo propuesto entrena diferentes máquinas de soporte vectorial, para dos clases, incluidos los datos de entrenamiento del grupo incremental. Las salidas de todas las máquinas de soporte vectorial son combinadas empleando un Perceptron Multicapa para determinar la etiqueta de clase de los patrones. Los experimentos son llevados a cabo con bases de datos del mundo real y múltiples clases.

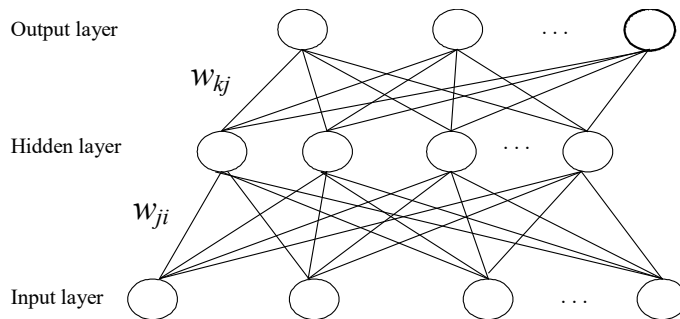
## **1 Introducción**

El cerebro de los seres humanos tiene la poderosa habilidad de incrementar su conocimiento. Por consiguiente, el cómo implementar esta característica es uno de los retos en la investigación del aprendizaje de máquina. La habilidad de aprender incrementalmente, de grupos de datos, es una característica importante que hace a los algoritmos de aprendizaje más adaptables a los problemas del mundo real. Existen tres escenarios que requieren aprendizaje incremental: no todos los datos de entrenamiento pueden ser obtenidos a un tiempo por el costo de coleccionarlos, los datos son adquiridos por grupos; algunas aplicaciones del mundo real necesitan aprendizaje inmediato en cuanto los datos de entrenamiento son obtenidos; si la muestra de entrenamiento es muy grande no todos los patrones pueden ser cargados en la memoria de la computadora.

Para Jantke [16], el aprendizaje incremental es construir una nueva hipótesis usando solamente la hipótesis anterior y la información reciente que se tiene a la mano. En este sentido, Polikar et al. [33] definieron algunos atributos deseables de los algoritmos de aprendizaje incremental, a saber:

1. Debe ser capaz de aprender la información adicional de los nuevos datos.
2. No debe requerir acceder a los datos originales usados para entrenar al Clasificador existente.
3. Debe conservar el conocimiento previamente adquirido (es decir no debe sufrir un olvido catastrófico).
4. Debe ser capaz de integrar nuevas clases introducidas con nuevos datos.

Zhou y Chen [47] distinguen tres clases de tareas de aprendizaje incremental: aprendizaje elemento-incremental (E-IL, por sus siglas en inglés); aprendizaje clase-incremental (C-IL, por sus siglas en inglés); y aprendizaje atributo-incremental (A-IL, por sus siglas en inglés). Sin embargo, C-IL y A-IL no han recibido mucha atención hasta ahora. Syed et al. [42] introdujeron dos tipos de métodos de aprendizaje incremental: aprendizaje de elemento, que usa un patrón a la vez, y



bloque para un grupo de aprendizaje, el cual usa un subconjunto de muestras de tamaño apropiado a la vez.

Un aspecto importante del aprendizaje incremental es, si la meta u objetivo cambia o es constante durante el proceso de aprendizaje. El primer caso es conocido como la persistencia de la meta, el segundo caso según Rüping [35] es verdaderamente un aprendizaje incremental. La diferencia práctica entre ambos aprendizajes es, que en el ajuste de la persistencia de la meta los patrones viejos pueden ser descartados, ya que ellos son ejemplos de una meta vieja, la cual puede ser muy diferente de la meta que se está tratando de aprender. En el caso del aprendizaje incremental, todos los patrones contienen la misma información sobre la meta (más específico: no se puede calificar la información del patrón en función de su edad). Como consecuencia, sólo puede juzgarse el rendimiento de un algoritmo de aprendizaje incremental comparando sus resultados con los resultados del algoritmo de aprendizaje entrenado con todos los datos simultáneamente.

Actualmente, la esencia de los algoritmos de aprendizaje para tareas con múltiples clase es un proceso de optimización que aspira a asegurar la habilidad de generalización con base en el ambiente de aprendizaje en curso. Por lo tanto, todos los algoritmos de aprendizaje de máquina, en su naturaleza, no aplican el aprendizaje incremental. La no adaptación se entiende en el sentido de que el modelo computacional carece de la habilidad de conseguir nuevos conocimientos o no puede conservar los conocimientos aprendidos [14]. El entrenamiento de las redes neuronales artificiales es un proceso de gradiente descendente, y por lo tanto la modificación de los pesos de conexión dañará los conocimientos aprendidos. El entrenamiento de las SVMs es un proceso de optimización global sobre la base de todos datos de entrenamiento. Por consiguiente, la adición de nuevos datos de entrenamiento genera cambios en los vectores soporte. Se han realizado una gran cantidad de trabajos para manejar este problema de no

Figura 1. Red neuronal Perceptron Multicapa de tres capas.

adaptación. Las redes neuronales artificiales adquieren nuevo conocimiento y reducen la pérdida de memoria adaptando únicamente algunos pesos o modificando el número de unidades en la capa oculta [12], sin embargo, la habilidad para aprender y la capacidad para almacenar conocimientos en las redes neuronales están limitadas. Cuando hay un corrimiento de meta en los datos de entrenamiento no vistos, el algoritmo propuesto por Syed et al. pierde su confiabilidad [35].

La combinación de Clasificadores es un método útil para máquina de aprendizaje [24] [47]. El combinar Clasificadores para el aprendizaje incluye el aprendizaje de múltiples clasificadores, el aprendizaje modular, y el meta aprendizaje. Muchos investigadores han aplicado técnicas para el aprendizaje incremental donde se combinan diferentes clasificadores. Polikar et al. propuso Learn ++ sobre la base del algoritmo AdaBoost [33]. Learn ++ satisface los criterios de aprendizaje incremental. Sin embargo, cuando se presenta un grupo de datos de entrenamiento incremental, Learn ++ debe entrenar muchos clasificadores, lo cual es una desventaja. Lu e Ichikawa propusieron un modelo de aprendizaje incremental con base en la descomposición de la tarea de aprendizaje en sub-módulos linealmente independientes [23]. Este modelo computacional usa dos reglas para realizar la descomposición y la integración de la solución [24], sin embargo, todas las muestras aprendidas deben conservarse. Macek propuso algoritmos de aprendizaje incremental

sobre la base de bagging y boosting, los aplicó con éxito a la clasificación de datos de EEG [25]. Wang et al. empleó un conjunto de clasificadores ponderados para buscar el flujo de datos con corrimiento de meta [45].

## 2 Perceptron Multicapa

La figura 1 muestra un Perceptron Multicapa formado por tres capas: entrada, oculta, y salida. Por ser una RNA totalmente conectada, los nodos de la capa oculta se encuentran conectados a todos los nodos tanto de entrada como de salida. Cada conexión tiene asociado un valor, el cual se modifica durante el entrenamiento. Existen diversos algoritmos para realizar el entrenamiento del PM, uno de ellos es el conocido como algoritmo de Backpropagation (BP). El objetivo de BP es determinar los pesos que minimicen, a partir de la ME, la función de error. Ésta se representa por el error cuadrático medio entre la respuesta de la red neuronal y su valor real. El algoritmo de BP realiza la modificación de los pesos mediante la expresión (1)

$$\Delta w(t) = -\eta \nabla E_p + \alpha \Delta w(t-1) \quad (1)$$

donde:  $\Delta w(t)$  y  $\Delta w(t-1)$  representan la variación de los pesos en la iteración  $t$  y  $t-1$  respectivamente,  $\nabla E_p$  es el gradiente de la función de error,  $\eta$  la razón de aprendizaje, y  $\alpha$  el momento. Estos dos últimos parámetros se introducen para controlar la magnitud del ajuste de los pesos a lo largo de la dirección descendente de la función de error y evitar oscilaciones alrededor del mínimo global, sus valores se determinan empíricamente,. Durante el desarrollo de este trabajo se utilizaron los valores de 0.9 y 0.7 para la razón de aprendizaje y momento respectivamente, ya que en diferentes investigaciones [7, 8, 28, 31] han mostrado un comportamiento adecuado en la disminución del error.

### 2.1 Topología de la red.

La habilidad del PM para afrontar con éxito tareas de reconocimiento de patrones, donde las clases no son linealmente separables, se debe a la inclusión de nodos entre las capas de entrada y salida. Tales nodos pueden agruparse en una o varias capas ocultas. Trabajos previos [6, 13, 15, 20, 23] han mostrado las ventajas de emplear perceptrones de tres capas, figura 1, en lugar de redes con un número mayor. Esto, debido a su menor carga computacional cuando se compara con redes de más de una capa oculta, y a su capacidad para formar aproximaciones arbitrariamente cercanas a cualquier relación entre los patrones de entrada y las clases [46]. Así como por su habilidad para separar regiones no convexas y desconectadas [26] en el espacio de observación. Además, Shah y Poon [36] han demostrado que las unidades de la capa oculta producen salidas linealmente independientes capaces de formar una base para la generación de las salidas de la red. Tal situación se presenta cuando la activación de los nodos de las capas oculta y de salida se determinan mediante el empleo de la función sigmoideal y lineal respectivamente. Con base en lo anterior, durante el desarrollo de este trabajo se utilizó un Perceptron Multicapa formado por tres capas: capa de entrada conteniendo un número de nodos igual  $N_a$  (cantidad de atributos en los patrones); capa oculta con  $N_a + 1$  nodos; y capa de salida formada por tantos nodos como clases tenga la tarea ( $N_c$ ). Se eligió la función sigmoideal como función de activación, ecuación (2).

$$f(net_j) = 1 / (1 + e^{-net_j}) \quad \text{con} \quad net_j = \sum_i w_{ji} \cdot s_i \quad (2)$$

donde:  $net_j$  es la acumulación ponderada de las salidas de los nodos de una capa anterior,  $w_{ji}$  es el peso de la conexión entre el nodo  $j$  y el nodo  $i$  de una capa

anterior,  $s_i$  es la salida del nodo  $i$ .

## 2.2 Aprendizaje Incremental con Perceptron Multicapa

El entrenamiento de las redes neuronales artificiales puede ser pasivo, activo o incremental. El aprendizaje pasivo es un proceso donde el entrenamiento está dirigido por todos los datos disponibles. Por contraste, una estrategia activa admite la interacción entre aprendiz y profesor. Por ejemplo, un proceso activo puede concentrarse en regiones específicas del dominio de entrada. El aprendizaje incremental es diferente de los aprendizajes pasivo y activo con respecto a la suposición de que todos los datos de entrada están disponibles a priori. Las razones principales para querer aprendizaje activo e incremental en las redes neuronales son: 1) admitir el entrenamiento por partes sobre el total del conjunto de datos, 2) permitir el aprendizaje de nuevo conocimiento, 3) adaptarse a la evolución del conocimiento, y 4) reforzar conocimientos en curso. Sin embargo, para conseguir aprendizaje incremental, es necesario que los parámetros libres en la red puedan cambiar con el tiempo, esto se traduce a añadir tanto nuevas neuronas como nuevas conexiones para desarrollar la red neural.

Es conocido que el Perceptron Multicapa (PM) tiene muchos defectos. Por ejemplo, es casi inútil en el contexto del aprendizaje incremental ya que olvida muy rápidamente lo aprendido cuando se le presentan los nuevos datos. Esto es conocido como el corrimiento de meta: ya que las neuronas sobre una capa no se comunican con alguna, cada neurona determina por separado qué parte del problema de clasificación enfrentará [49]. Otra desventaja del PM es su incapacidad de generar superficies de separación cerradas en el espacio de observación. Ese comportamiento inadecuado ha sido demostrado matemáticamente por M. Gori y F. Scarselli [49], quienes concluyen como una consecuencia directa que el PM es inadecuado para aplicaciones donde se requiera un criterio de rechazo.

Para superar tanto el problema de corrimiento de meta, como el de superficies de separación abiertas, se ha concebido especializar cada neurona oculta restringiéndola a actuar solamente dentro de una región (un grupo) localizado en el espacio de observación. Las redes de función de base de radial (RBF) [49, 49] fueron concebidas bajo estas ideas.

## 3 Máquinas de Soporte Vectorial

La idea básica que impulsó el desarrollo inicial de SVMs consiste en, dada una tarea de aprendizaje en particular, con una cantidad finita de datos de entrenamiento, se obtiene el mejor rendimiento en la generalización si se alcanza el balance correcto entre la exactitud alcanzada sobre los datos de entrenamiento, y la "capacidad" de la máquina, es decir la habilidad de la máquina para aprender cualquier conjunto de entrenamiento sin error. Una máquina con demasiada capacidad es como un botánico que tiene memoria fotográfica, al presentarle un nuevo árbol, llega a la conclusión de que no es un árbol porque tiene un número diferente de hojas a los árboles que ha visto con anterioridad; una máquina con muy poca capacidad es como el hermano flojo del botánico, que declara si es verde, entonces es un árbol [1]. Ninguno de los dos puede generalizar bien. La exploración y la formalización de estos conceptos han dado como resultado uno de los máximos logros de la teoría de aprendizaje estadístico [44].

Esta teoría pretende dar respuesta a la pregunta, cómo puede encontrarse una función  $f$  de una clase de funciones  $(f_\alpha)_{\alpha \in A}$ , que minimice el riesgo esperado

$$R[f] = \iint L(y, f(x)) dP(y | x) dP(x) \quad (3)$$

con respecto a una de función de pérdida  $L$ , cuando son desconocidas las distribuciones de los patrones  $P(x)$  y su clasificación  $P(y|x)$  y tienen que ser calculadas de un conjunto finito de patrones  $(x_i, y_i)_{i \in I}$ .

El algoritmo SVM soluciona este problema minimizando el riesgo regularizado  $R_{reg}[f]$ , que es la suma ponderada del riesgo empírico  $R_{emp}[f]$  con respecto a los datos  $(x_i, y_i)_{i \in 1..n}$  y un término de complejidad  $\|w\|^2$

$$R_{reg}[f] = R_{emp}[f] + \lambda \|w\|^2 \quad (4)$$

En su formulación básica, las SVMs encuentran una función de decisión lineal  $y = f(x) = \text{sign}(w \cdot x + b)$  que minimiza la predicción del error sobre el conjunto de entrenamiento y promete el mejor rendimiento en la generalización. Dados los ejemplos  $(x_1, y_1), \dots, (x_n, y_n)$  se busca la solución al siguiente problema de optimización:

$$\begin{aligned} \Psi(w, \xi, \xi^*) &= \frac{1}{2} (w^T w) + C \sum_{i=1}^n \xi_i \\ \rightarrow \min \end{aligned} \quad (5)$$

sujeto a

$$y_i (w^T x_i + b) \leq 1 - \xi_i, i = 1, \dots, n \quad (6)$$

$$\xi_i \geq 0, i = 1, \dots, n \quad (7)$$

El hiperplano  $w$  tiene una representación en términos de los patrones de entrenamiento  $(x_i, y_i)_{i \in I}$  y sus multiplicadores de Lagrange  $(\alpha_i)_{i \in I}$ , que son calculados durante el proceso de optimización:

$$w = \sum_{i \in I} \alpha_i y_i x_i$$

Los casos con el multiplicador de Lagrange diferente de cero son llamados Vectores Soporte (SV por sus siglas en inglés). Una propiedad clave de las Máquinas de Soporte Vectorial es, el entrenamiento de las SVM sobre los vectores de soporte da el mismo resultado que el entrenamiento sobre el total de patrones.

La constante  $C$  óptima para el problema de aprendizaje es usualmente determinado por la técnica de selección del modelo, por ejemplo validación-cruzada [35].

En la ecuación (6), el riesgo empírico de la solución de SVM es medido con respecto a una función de pérdida lineal. Es importante mencionar, la teoría del algoritmo de Vectores Soporte no está restringida al caso de las funciones de pérdida lineal, puede ser extendida a otro tipo de funciones de pérdida [38].

### 3.1 Límite sobre el rendimiento de la generalización de una Máquina de Aprendizaje de Reconocimiento de Patrones

Hay una serie de límites que gobiernan la relación entre la capacidad de una máquina de aprendizaje y su rendimiento. Para su determinación teórica se debe dar repuesta a las siguientes consideraciones, si el número de puntos de datos aumenta, bajo qué circunstancias y qué tan rápidamente, la media de una cantidad empírica converge uniformemente a la media verdadera (que se calcula con una cantidad infinita de datos) [44]. Empecemos con uno de estos límites.

Suponiendo que se tienen  $l$  observaciones. Cada observación consiste de un par: un vector  $x_i \in R^n$ ,  $i = 1, 2, \dots, l$  y la etiqueta asociada  $y_i$ , dada por un experto. Además, se supone que

existe una distribución de probabilidades  $P(x, y)$  desconocida, de la cual provienen los datos, i.e., se presume que los datos son "iid" (por ser independientes y de forma idénticamente distribuidos). (Se usará  $P$  para las distribuciones de probabilidad, y  $p$  para su densidad). Note que esta suposición es más general que el relacionar una  $y$  a cada  $x$ : esto permite que haya una distribución de  $y$  para una  $x$  dada. En tal caso, el experto asignaría etiquetas  $y_i$  de acuerdo con una distribución fija, condicionada a  $x$ . Esta situación es más general que en el caso particular de Reconocimiento de Patrones donde se tiene una  $y$  fija para una  $x$  dada.

Suponga que tenemos una máquina cuya tarea es aprender la correspondencia de  $x_i \mapsto y_i$ . En realidad, la máquina será definida por un conjunto de correspondencias posibles  $x \mapsto f(x, \alpha)$ , donde la función  $f(x, \alpha)$  es regulada por el parámetro  $\alpha$ . La máquina se asume como determinista: para una entrada  $x$ , y una elección de  $\alpha$ , siempre dará la misma salida  $f(x, \alpha)$ . Una elección particular de  $\alpha$  genera lo que se conoce como "Máquina Entrenada". Por ejemplo, una red neuronal artificial con su arquitectura fija es una máquina de aprendizaje, si  $\alpha$  representa a los pesos incluidos los bias.

El error esperado para una máquina entrenada es:

$$R(\alpha) = \int \frac{1}{2} |y - f(x, \alpha)| dP(x, y) \quad (8)$$

Note que, cuando existe una función de densidad  $p(x, y)$ ,  $dP(x, y)$  puede sustituirse por  $p(x, y) dx dy$ . Ésta es una forma elegante de escribir el error medio verdadero, pero a menos que se tenga un cálculo aproximado de  $P(x, y)$ , no es muy útil.

La cantidad  $R(\alpha)$  se denomina riesgo esperado, o simplemente riesgo. Como esta cantidad es la que se desea determinar será llamada riesgo verdadero. Por otro lado, el riesgo empírico  $R_{emp}(\alpha)$  estima el error medio sobre el conjunto de entrenamiento, para un número fijo y finito de observaciones:

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^l |y_i - f(x_i, \alpha)| \quad (9)$$

$R_{emp}(\alpha)$  es una cantidad fija para una elección de  $\alpha$  y un conjunto de entrenamiento  $\{x_i, y_i\}$ , además en su cálculo no aparece ninguna distribución de probabilidades.

La cantidad  $\frac{1}{2} |y_i - f(x_i, \alpha)|$  es conocida como pérdida. Para el caso descrito aquí, ésta sólo puede tomar los valores 0 y 1. Ahora bien, si se escoge una  $\eta$  tal que  $0 \leq \eta \leq 1$ . Entonces para pérdidas que toman esos valores, con probabilidad  $1 - \eta$ , tienen el siguiente límite [42]:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left( \frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l} \right)} \quad (10)$$

donde  $h$  es un entero no-negativo denominado dimensión Vapnik Chervonenkis (VC), y es una medida de la capacidad de la máquina de aprendizaje. Al miembro del lado derecho de la Ec. (10) se le conoce como riesgo límite. El segundo término del lado derecho se denomina confianza VC.

Se pueden mencionar tres puntos clave sobre este límite. Primero, es independiente de  $P(x, y)$ . Se asume que tanto los datos de entrenamiento como los datos de prueba son obtenidos por

separado de acuerdo a alguna  $P(x, y)$ . Segundo, generalmente no es posible calcular el lado izquierdo. En tercer lugar, si conocemos  $h$ , fácilmente se calcula el lado derecho. Por lo tanto, dadas diferentes máquinas de aprendizaje, y eligiendo un valor fijo y suficientemente pequeño de  $\eta$ , se tomará aquella máquina que minimice el lado derecho, i. e. se escoge la máquina que dé el menor límite superior del riesgo verdadero. Esto proporciona, en principio, un método para escoger una máquina de aprendizaje para una tarea en particular.

### 3.2 Dimensión VC

La dimensión VC es una propiedad de un conjunto de funciones  $\{f(\alpha)\}$  (como se mencionó,  $\alpha$  se emplea como un conjunto genérico de parámetros: una elección de  $\alpha$  especifica una función en particular), y puede ser definido para varias clases de funciones  $f$ . Aquí se considera solamente las funciones que corresponden al caso de reconocimiento de patrones para dos clases, tal que  $f(x, \alpha) \in \{-1, 1\} \forall x, \alpha$ . Dado un conjunto  $l$  de puntos, éstos pueden ser etiquetados en  $2^l$  maneras posibles, si para cada manera de etiquetar, se puede encontrar un miembro del conjunto  $\{f(\alpha)\}$  que asigna la etiqueta correctamente, se dice que ese conjunto de puntos puede separarse en grupos con una sola etiqueta por el conjunto de las funciones. La dimensión VC para el juego de las funciones  $\{f(\alpha)\}$  se define como el número máximo de puntos de entrenamiento que pueden ser separados en grupos por  $\{f(\alpha)\}$ . Note que, si la dimensión VC es  $h$ , entonces existe al menos un conjunto de  $h$  puntos que pueden ser agrupado, sin embargo no todos los conjuntos de  $h$  puntos pueden ser agrupados.

### 3.3 Puntos representados con hiperplanos orientados en $R^n$

Sea el espacio de los datos  $R^2$ , y el conjunto  $\{f(\alpha)\}$  formado por líneas rectas orientadas, entonces para una línea en particular, los puntos colocados en uno de sus lados son asignados a la clase 1, el resto del otro lado a la clase -1. La orientación es mostrada en la Figure 2 por una flecha, que especifica el lado donde los puntos son asignados con la etiqueta 1. Mientras es posible encontrar tres puntos que pueden ser separados en grupos de elementos con una sola etiqueta por este conjunto de funciones, no es posible encontrar cuatro. La dimensión VC del conjunto de líneas orientadas en  $R^2$  es tres.

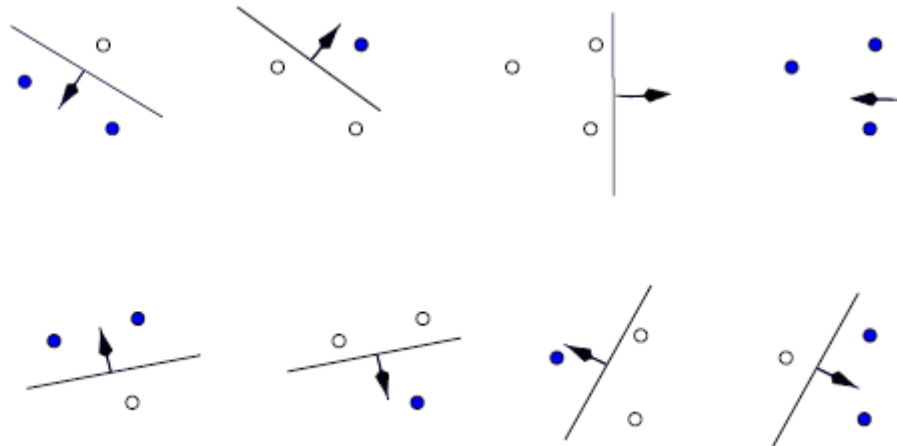


Figure 2. Tres puntos en  $R^2$ , separados en grupos de elementos con una sola etiqueta empleando líneas orientadas.

Para un conjunto de hiperplanos en  $R^n$ , se puede demostrar:

**Teorema:** Dado un conjunto de  $m$  puntos en  $R^n$ . Elegir uno de ellos como el origen. Los  $m$  puntos pueden ser separados en grupos, cuyos elementos tienen la misma etiqueta, por hiperplanos orientados si y sólo si los vectores de posición de los puntos restantes son linealmente independientes.

**Corolario:** La dimensión VC del conjunto de hiperplanos orientados en  $R^n$  es  $n+1$  ya que siempre podemos escoger  $n+1$  puntos, marcar uno de los puntos como el origen, de tal forma que los vectores de los  $n$  puntos restantes son linealmente independientes, pero nunca se puede escoger  $n+2$ , debido a que no hay  $n+1$  vectores en  $R^n$  linealmente independientes.

### 3.4 Vectores Soporte

Para un problema de clasificación, donde contamos con  $l$  puntos  $\{(x_1, y_1), \dots, (x_l, y_l)\}$  donde  $x_i$  es el patrón  $i$ -ésimo que tiene asociada la etiqueta de clase  $y_i$ , entrenar una Máquina de Soporte Vectorial (SVMs) involucra resolver un problema cuadrático cuya solución genera una función de decisión del siguiente tipo

$$f(x) = \text{sgn}\left[\sum_{i=1}^l y_i \alpha_i (x - x_i) + b\right] \quad (11)$$

Si se desea una superficie de decisión diferente a un hiperplano, primero se realiza una transformación no lineal de los patrones de entrenamiento de entrada  $x_1, \dots, x_l$  a un espacio característico de mayor dimensión empleando una correspondencia de  $\Phi(x_i) \mapsto z_i$ , y realizar ahí una separación lineal. Esto lleva a una función de decisión de la forma:

$$\begin{aligned} f(x) &= \text{sgn}\left[\sum_{i=1}^l y_i \alpha_i (\Phi(x) \cdot \Phi(x_i)) + b\right] \\ &= \text{sgn}\left[\sum_{i=1}^l y_i \alpha_i K(x, x_i) + b\right] \end{aligned} \quad (12)$$

donde  $K(x, x_i) = (\Phi(x) \cdot \Phi(x_i))$  es conocido como el Kernel. Nótese que ahora el procedimiento de entrenamiento y clasificación requiere calcular un producto punto de la forma  $(\Phi(x) \cdot \Phi(x_i))$  en un espacio de mayor dimensión. Para evitar estas operaciones computacionalmente costosas sólo se consideran aquellas clases de función  $\Phi$  que reducen al kernel  $K$  a una función simple.

Frecuentemente, sólo una pequeña fracción de los coeficientes  $\alpha_i$  son diferente de cero. Los correspondientes pares de patrones de entrada  $x_i$  (conocidos como vectores soporte) y su etiqueta de salida  $y_i$  definen totalmente la función de decisión [41]. Junto con los coeficientes  $\alpha_i$ , son conservados para usarse en el procedimiento de clasificación. El resto de patrones de entrenamiento que no contribuyen a la función de decisión pueden considerarse como redundantes.

### 3.5 Interpretación de los Vectores Soporte

Se argumenta, que los Vectores Soporte suministran una representación suficiente de los patrones para una tarea de clasificación en particular, esto porque al entrenar una SVM con los Vectores Soporte da la misma función de decisión que al entrenar con el conjunto total de datos. Por supuesto, esto solamente quiere decir que los Vectores Soporte son una representación suficiente de la función de decisión generada con los patrones, no son éstos en si mismo.

La diferencia importante es, en términos del principio de Aprendizaje Estadístico formalizado en la ecuación 1, Los Vectores Soporte proveen una estimación de  $P(y|x)$ , pero no de  $P(x)$ . En la Figura 3, ambos conjuntos A y B tienen la misma  $P(y|x)$ , pero diferente  $P(x)$ . Si los Vectores Soporte son empleados para representar el conjunto de datos completo, como en el algoritmo de aprendizaje incremental, esto tiene que ser tomado en cuenta.

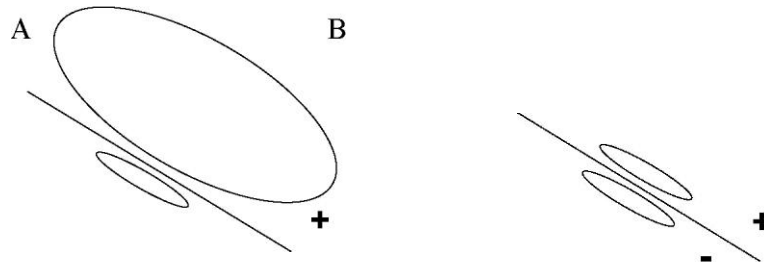


Figura 3. Ejemplos de dos conjuntos A y B con igual  $P(y|x)$  pero diferente  $P(x)$ .

### 3.6 Aprendizaje Incremental con Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial (SVMs) han sido exitosamente empleadas para aprender conjuntos de datos grandes con alta dimensionalidad. Esto es debido al hecho de que el poder de generalización de las SVMs no depende de contar con todos los datos de entrenamiento, sino solamente de un subconjunto de ellos, los denominados vectores de soporte. Desafortunadamente, el entrenar las máquinas de soporte vectorial es un proceso que consume mucho tiempo, en especial cuando se tienen datos con ruido.

Como el número de vectores de soporte es pequeño comparado con la cantidad de patrones de entrenamiento, los SVMs se muestran como una herramienta eficaz para el aprendizaje incremental ya que representan los datos de los grupos de entrenamiento a través de los vectores de soporte.

Syed et al. iniciaron los trabajos de aprendizaje incremental en las (SVMs) [41]. Su algoritmo conserva solamente los Vectores Soporte en cada paso incremental y los añade al entrenamiento para el próximo paso. El razonamiento detrás de esto es: la función de decisión resultante de una SVM solamente depende de sus Vectores Soporte, i.e. entrenar una SVM sólo sobre los Vectores Soporte da como resultado la misma función de decisión como si se entrenará con el conjunto de datos total. Por esto, se puede esperar conseguir un resultado incremental que es igual al resultado non-incremental, si el último conjunto de entrenamiento contiene todos los patrones que son Vectores Soporte en el caso non-incremental.

Pero cómo decidir si un patrón terminará como Vector Soporte en el caso no-incremental sin verdaderamente entrenar con todos los datos? Si un conjunto de datos es una buena muestra, i.e. si las propiedades estadísticas del conjunto y el conjunto de datos total no difieren mucho, se puede esperar que la función de decisión resultante es burdamente similar a la función de decisión final. Por lo tanto, un Vector Soporte en el archivo final es probablemente que sea un Vector Soporte también en las previas iteraciones.

El problema con este enfoque es la suposición de que el grupo de los datos sea una muestra apropiada. Esto puede ser probable si los datos son colectados de antemano y presentados a la SVM en conjuntos aleatorios, no hay manera para decir que éste es el caso si los patrones son generados en línea durante el entrenamiento.

Como se estableció arriba, el algoritmo incremental-SV sufre del problema, que los Vectores Soporte no describen el conjunto total de los datos sino solamente la función de decisión producida por ellos. Para tratar con este problema Rüping [35] hace el error sobre los Vectores Soporte viejos (que representa el conjunto de aprendizaje viejo) más costoso que el error sobre un nuevo

ejemplo. Afortunadamente, esto puede ser realizado fácilmente en el algoritmo de Vector Soporte. Sea  $(x_i, y_i)_{i \in S}$  los Vectores Soporte viejos y  $(x_i, y_i)_{i \in I}$  los nuevos patrones. Reemplazando la definición (5) por

$$\frac{1}{2}(w^T w) + C(\sum_{i \in I} \xi_i + L \sum_{i \in S} \xi_i)$$

Esta modificación del problema de SVM puede ser vista como entrenar el SVM con respecto a una nueva función de pérdida.

Una elección natural para L es asignarla como el número de patrones en el grupo previo dividido por el número de Vectores Soporte.

$$L = \frac{\#ejemplos}{\#SVMs}$$

Esto viene de la idea de aproximar el error medio de una función de decisión arbitraria de todos los patrones por el error medio sobre únicamente los Vectores Soporte. El resultado obtenido con este procedimiento está mucho más cerca del generado con la muestra completa, que el de Syed et al [41].

Con base en el trabajo de Syed et al [41]., Dominiconi y Gunopulos hicieron una investigación adicional [10] donde analizaron cuatro técnicas diferentes para el aprendizaje incremental de las SVMs: la técnica manejada por error, técnica de partición corregida, técnica de margen excedido, y técnica de margen excedido + errores.

Cauwenberghs y Poggio [2] propusieron un algoritmo de aprendizaje en línea para SVMs. En su algoritmo, los incrementos adiabáticos preservan la condición de Kuhn-Tucker sobre todos los datos de entrenamiento conocidos, al integrar un nuevo elemento. Diehl y Cauwenberghs ampliaron este algoritmo para incrementalmente acomodar varios elementos a la vez [9]. Sin embargo, tienen como desventaja que deben conservar todos los datos de entrenamiento empleados. Ralaivola y su colega propusieron una estrategia local para el aprendizaje en línea de las SVMs [34]. Liu et al. [22] exploraron el aprendizaje incremental en grupo con SVMs, mostraron que este aprendizaje requiere dar solución a un sistema cuadrático convexo, igual que el algoritmo SVMs estándar.

Como se comentó en la introducción, para Rüping el verdadero aprendizaje incremental en las Máquinas de Soporte Vectorial se da cuando hay un cambio en la meta y se descartan patrones del entrenamiento original. Esto ha sido estudiado en [41], donde se ha validado experimentalmente que los SVMs manejan bien los corrimientos de la meta con respecto a los criterios de estabilidad de los resultados durante el entrenamiento, la mejora de la exactitud del pronóstico durante el avance del entrenamiento, y la recuperación de los errores resultado del corrimiento del modelo. Otro enfoque para este manejo ha sido presentado en [18], donde un estimador de rendimiento [17] ha sido usado para detectar si un corrimiento subyacente ocurrió, en tal punto los datos viejos fueron descartados y el entrenamiento tuvo lugar solamente sobre los nuevos datos.

En [35] el ajuste del aprendizaje incremental se realiza presentando los datos al algoritmo en varios grupos, de tal forma que éste debe generar un resultado preliminar después de cada paso de entrenamiento. Para mantener bajo el consumo de tiempo y espacio no se permite usar todos los datos directamente en el último paso de entrenamiento. Sin embargo, se vigila la idea del corrimiento del modelo, porque en los problemas de la vida real puede ocurrir un cambio en la meta entre diferentes grupos de datos, lo cual es simplemente un artificio de la manera en que los datos son presentado al algoritmo de aprendizaje.

Cuando los patrones de aprendizaje son generados para un experimento controlado, el investigador puede encontrar favorable seguir un plan de prueba que cambia los parámetros del proceso sucesivamente. Si estos ejemplos son usados para el aprendizaje incremental, los grupos iniciales contienen patrones de regiones muy diferentes del espacio de parámetros que los últimos

grupos. También puede observarse cambios en los datos de entrenamiento que no tienen correspondencia con los parámetros controlables del experimento, por ejemplo, en aplicaciones en ingeniería la calidad de una máquina se deteriora durante el transcurso de su ciclo de vida, o en aplicaciones médicas si el algoritmo aprende sobre los datos de un paciente a la vez. En ambas situaciones, se está interesado en un resultado que sea óptimo para todos los datos (todos los parámetros/ momentos/pacientes), esto es aunque hay cambios observables e importantes en los datos de un grupo a otro, para estos casos no debe presentarse ningún corrimiento en el modelo. Por supuesto, todos estos problemas pueden solucionarse de manera eficaz realizando muestreos de datos aleatoriamente.

Las técnicas incrementales han sido desarrolladas para facilitar el aprendizaje por lotes de las SVM sobre un gran conjunto de datos, y ha encontrado un uso muy amplio en la comunidad SVM, e.g. [30],[32], [11], [43]. El aprendizaje incremental SVM es particularmente atractivo para un ajuste en línea, y para aprendizaje activo [4]. Muchas de estas técnicas son aproximaciones y requieren varias pasadas de los datos para alcanzar la convergencia. Un procedimiento para un aprendizaje incremental “adiabático” exacto para un clasificador SVM, en un solo paso de los datos, fue introducido en [2] y extendido a una SVM para regresión [27] e incrementos con conjuntos grandes [37].

Las investigaciones en aprendizaje incremental han sido enfocadas en gran parte en aprendizaje en línea en contraposición al problema de selección de modelo grande. La adaptación dinámica del Kernel es presentada en [5], y procedimientos aproximados para la selección del modelo, usando la aproximación leave-one-out (LOO) y límites, son dados en [19] y [3]. El procedimiento de aprendizaje incremental para SVM puede ser revertido realizando un proceso que permita olvidar lo aprendido para una selección de modelo exacto con base en LOO [2].

En [9] se extiende el paradigma del aprendizaje incremental para SVM de [2] a un marco general de aprendizaje incremental, adaptación y optimización que permita aprender u olvidar uno o múltiples patrones, adaptar los actuales SVM cambiando los parámetros del kernel y la regularización, y evaluar el desempeño de la generalización a través de la estimación exacta del error en leave-one-out.

#### **4 Un nuevo algoritmo para el aprendizaje incremental**

Cabe mencionar que las Máquinas de Soporte Vectorial son clasificadores cuyo comportamiento es óptimo cuando la tarea de clasificación contiene dos clases, linealmente separables, ya que para simular la frontera de decisión emplean hiperplanos, sin embargo en este trabajo, se pretende diseñar un procedimiento para bases de datos con más de dos clases, que en la mayoría de los casos son no linealmente separables

El nuevo algoritmo de Aprendizaje Incremental combina una red neuronal artificial Perceptron Multicapa y Máquinas de Soporte Vectorial. Inicialmente se emplea un Perceptron Multicapa para simular clases linealmente separables. Para esto se entrena un PM y se eligen los patrones de cada clase que son bien clasificados por el PM. Al igual que las SVMs el PM emplea hiperplanos para separar el espacio de observación, en regiones donde se tienen patrones de una sola clase, lo cual puede ser similar en ambos casos. Dada la simulación de las clases linealmente separables se especializa una Máquina de Soporte Vectorial por clase, lo cual se realiza entrenando ésta con una muestra de entrenamiento integrada por los patrones de la clase y el resto de patrones en una única clase.

Una vez creadas las SVMs por clase, con las salidas de cada una de éstas se integra un patrón de soporte, donde una de sus características es notablemente menor al resto. Con estos elementos se entrena un PM que generará la clasificación final.

Al obtener patrones de una nueva clase se crea una SVM con una muestra de entrenamiento formada por los patrones de la nueva clase y los Vectores Soporte de las SVMs ya creadas. El nuevo conocimiento se integra formando el patrón soporte con las salidas de las SVMs, por clase incluida la nueva, y se realiza el entrenamiento de un nuevo PM. A continuación se presenta el algoritmo:

8. Se crean tantas SVMs como clases tenga la tarea de aprendizaje.
9. La muestra de entrenamiento para cada SVM debe contener dos clases, la primera se integra con los patrones de una de las clases, la segunda se forma con los elementos de las clases restantes. Se deben emplear los patrones que simulan las clases linealmente separables. Este paso pretende crear máquinas expertas en identificar una de las clases en particular.
10. Una vez entrenadas las máquinas de soporte vectorial se crea un patrón de soporte 1 con los dos valores de salida de cada una de las SVM, esto es la cantidad de características será igual a dos veces la cantidad de máquinas de soporte vectorial
11. El patrón de soporte 1 se emplea para entrenar un Perceptron Multicapa. Cuya capa de entrada tiene el número de nodos igual a dos veces la cantidad de clases, la capa de salida tiene tantos nodos como el número de clases.
12. Para integrar una nueva clase (aprendizaje incremental) se adiciona una máquina de soporte vectorial la cual se entrena con los patrones desconocidos y una creada con los vectores de soporte de cada Máquina de Soporte Vectorial.
13. Las salidas de la máquina de soporte vectorial se agregan a los patrones de soporte 1 para crear el patrón de soporte 2, el cual sirve para entrenar un Perceptron Multicapa 2 cuya capa de salida contendrá un nodo adicional que integra la nueva clase.

Para adicionar otra clase se repiten los pasos 5-6, tantas veces como sea necesario.

## 5 Experimentos

Para evaluar el poder de generalización del nuevo procedimiento de aprendizaje incremental se emplearon diferentes bases de datos que representan casos del mundo real. En su mayoría fueron obtenidas del almacén de datos de la Universidad Irving en California [29], y son conocidas como: Breast Cancer (Can), Glass (Gla), Iris, Image Segmentation (Ima), Ionosphere (Ion), Liver Disorders (Liv), Pima (Pim), Sonar (Son), Vehicle (Veh), Vowel (Vow) y Wine (Win).

La tabla 1 presenta la información relevante de las bases de datos.

Tabla 1. Características de las bases de datos.

|                       | Can | Gla | Ima | Ion | Iris | Liv | Son | Veh | Vow | Win |
|-----------------------|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|
| Cantidad de atributos | 9   | 9   | 18  | 34  | 4    | 6   | 60  | 18  | 10  | 13  |
| Cantidad de clases    | 2   | 6   | 7   | 2   | 3    | 2   | 2   | 4   | 11  | 3   |
| No. de patrones       | 683 | 214 | 420 | 351 | 150  | 345 | 208 | 846 | 528 | 178 |

Con relación a la red neuronal artificial, en las pruebas se empleó un Perceptron Multicapa, entrenado con el algoritmo de Backpropagation, integrado por tres capas y función de activación sigmoideal. Los valores para la razón de aprendizaje y momento fueron 0.9 y 0.7, respectivamente. La actualización de los pesos se practicó al introducir cada patrón. Todos los entrenamientos se realizaron con el número de iteraciones máximo, igual a 30,000. Como criterio de convergencia se empleó un error del sistema igual a 0.0001 para todas las muestras.

Antes de realizar los entrenamientos fue necesario modificar las bases de datos originales en dos sentidos:

- c) Para realizar el aprendizaje incremental se eliminaron todos los patrones de una o dos clases, posteriormente éstos se emplearon para realizar el aprendizaje incremental.
- d) La simulación de las clases linealmente separables se llevó a cabo entrenando un Perceptron Multicapa, los patrones bien clasificados fueron separados para integrar las muestras de entrenamiento para las Máquinas de Soporte Vectorial.

Para evaluar el poder de generalización se empleo el método de validación cruzada con 10 repeticiones, en la tabla 2 se presenta el promedio de los errores con la muestra completa y empleando aprendizaje incremental.

Tabla 2. Error promedio del Perceptron Multicapa con la muestra completa (MC), y con aprendizaje incremental (AI).

|                     | Can  | Gla  | Ima  | Ion  | Iris | Liv  | Son  | Veh  | Vow  | Win  |
|---------------------|------|------|------|------|------|------|------|------|------|------|
| Error promedio MC   | 11.3 | 54.8 | 7.5  | 11.1 | 3.3  | 34.8 | 44.4 | 38.1 | 40.2 | 3.9  |
| Desviación estándar | ±4.8 | ±11. | ±5.0 | ±6.9 | ±5.7 | ±8.8 | ±14. | ±13. | ±13. | ±3.8 |
| Error promedio AI   | 7.3  | 69.5 | 15.7 | 32.4 | 4.4  | 45.3 | 53.6 | 49.4 | 77.1 | 16.1 |
| Desviación estándar | ±0.0 | ±6.0 | ±1.3 | ±9.0 | ±1.5 | ±3.8 | ±7.9 | ±3.2 | ±4.9 | ±1.7 |

## 6 Conclusiones

Como puede observarse, los resultados obtenidos en los experimentos, en su mayoría, no presentan una buena aproximación al error obtenido para la muestra completa. Aunque, cabe mencionar que la desviación estándar, es decir la variación entre los resultados de cada repetición, es menor al 10% en la mayoría de resultados.

## Bibliografía

- [1] Burge Christopher J. C., A Tutorial on Support Vector Machines for Pattern Recognition, Kluwer Academic Publishers, pp:1-43. [Burge]
- [2] Cauwenberghs, G. and Poggio, T.: Incremental and Decremental Support Vector Machine Learning. In: Advances in Neural Information Processing Systems, MIT Press, 13 (2001) 409-415. [7 Wan]
- [3] Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. In Machine Learning Kluwer Academic, 2002. [[5] Diehl]
- [4] Colin Campbell, Nello Cristianini, and Alex Smola. Query learning with large margin classifiers. In Proceedings, 17<sup>th</sup> International Conference on Machine Learning, pages 111-118. Morgan Kaufmann, San Francisco, CA, 2000. [[3] Diehl]
- [5] Cristianini, N. , C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines . In NeuroCOLT Technical Report NC-TR-98-017. Royal Holloway College, University of London, UK, 1998. [[6] Diehl]
- [6] Cybenko, G. (1989). "Approximation by superposition of a sigmoid function", Mathematics of Control, Signals and Systems, 2, pp.303-314. [6. ART1-MECIARP04]
- [7] Dai, Hengchang and C. MacBeth (1997). "Effects of learning parameters on learning procedure and performance of a BPNN", Neural Networks, vol.10, no.8, pp.1505-1521. [7. ART1-MECIARP04]
- [8] Demuth, H. and Beale, M. (1993). "Neural networks toolbox for use with MATLAB: user's guide", Natick, MA: The Math Works, Inc. [9. ART1-MECIARP04]
- [9] Diehl, C.P. and Cauwenberghs, G.: SVM Incremental Learning, Adaptation and Optimization. In: Proceedings of IJCNN-03 (2003) 2685-2690 [8 Wan]
- [10] Domeniconi, C. and Gunopulos, D.: Incremental Support Vector Machine Construction. In: Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, California, USA, (2001) 589-592 [6 Wan]
- [11] Friess, T. T., N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In Proceedings, 15<sup>th</sup> International Conference on Machine Learning. Morgan Kaufmann, 1998. [[8] Diehl]
- [12] Fu, L.M., Hsu, H.H., and Principe, J.C.: Incremental Back-propagation Learning Networks. IEEE Transaction on Neural Networks, 7 (1996) 757-761 [12 Wan]
- [13] Funahashi, K. (1989). "On the approximate realization of continuous mapping by neural networks", Neural Networks, vol.2, pp.183-192. [12. ART1-MECIARP04]
- [14] Grossberg, S.: Nonlinear Neural Networks: Principles, Mechanisms and Architectures. Neural Networks, 1 (1988) 17-61 [11 Wan]

- [15] Irie, B., and S. Miyake (1988). "Capabilities of three-layered perceptrons", en Proc. of the IEEE Conference on Neural Networks, vol.I, pp.641-6[18. ART1-MECIARP04]
- [16] Jantke, P.: Types of Incremental Learning. AAAI Symposium on Training Issues in Incremental Learning, March 23-25, Stanford CA, 1993[Wan]
- [17] Joachims T. Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference on Machine Learning*, San Francisco, 2000. Morgan Kaufman. [[Joachims, 2000]Ruping]
- [18] Klinkenberg Ralf and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco, 2000. Morgan Kaufmann. [[Klinkenberg and Joachims, 2000]Ruping]
- [19] Lee J. H. and C. J. Lin. Automatic model selection for support vector machines. <http://www.csie.ntu.edu.tw/~cjlin/papers/modelselect.ps.gz>, 2000. [[10] Dielhn]
- [20] Lin, Che-Chern and Amor El-Jaroudi (1998). "An algorithm to determine the feasibilities and weights of two-layer perceptrons for partitioning and classification", *Pattern Recognition*, vol.31, no.11, pp.1613-1625. [22. ART1-MECIARP04]
- [21] Lippmann, R. P. (1987). "An introduction to computing with neural nets", *IEEE Acoust. Speech Signal Process. Mag.*, 4, pp.4-22. [23. ART1-MECIARP04]
- [22] Liu, Y.G., He, Q.M., and Chen, Q.: Incremental Batch Learning with Support Vector Machines. In: *Proceedings of the 5th World Congress on Intelligence Control and Automation*, Hangzhou, P.R. China (2004) 1857-1861[10 Wan]
- [23] Lu, B.L. and Ichikawa, M.: Emergent Online Learning in Min-max Modular Neural Networks. In: *Proceedings of IJCNN'01* (2001) 2650-2655[16 Wan]
- [24] Lu, B.L., and Ito, M.: Task Decomposition and Module Combination Based on Class Relations: a Modular Neural Networks for Pattern Classification. *IEEE Transaction on Neural Networks*, 10 (1999) 1244-1256[14 Wan]
- [25] Macek, J.: Incremental Learning of Ensemble Classifiers on ECG data. In: *Proceedings of CBMS'05* (2005) [17 Wan]
- [26] Makhoul, J., A. El-Jaroudi and R. Schwartz (1991). "Partitioning capabilities of two-layer neural networks", *IEEE Trans. Signal Process.*, vol.39, no.6, pp.436-1440. [24. ART1-MECIARP04]
- [27] Martin M. On-line support vector machines for function approximation. <http://www.lsi.upc.es/dept/techreps/html/R02-11.html>, 2002. [[11] Dielhn]
- [28] McClelland, J. and Rumelhart, D. (1988). "Explorations in parallel distributed processing, a handbook of model, program, and exercises", Cambridge, MA. [26. ART1-MECIARP04]
- [29] Murphy, P. M., and D.W. Aha (1991), "UCI repository of machine learning databases", Irvine, University of California, anonymous FTP:/pub/machine-learning-database in ics.uci.edu. [27. ART1-MECIARP04]
- [30] Osuna E., R. Freund, and F. Girosi. An improved training algorithm for support vector machines . In *Proceedings , 1977 IEEE Workshop on neuroal Networks for Signal Processing*, pages 276-285, 1997. [[12] Dielhn]
- [31] Pao, Y. H. (1989). "Adaptive pattern recognition and neural networks", Ed. Addison-Wesley, MA. [28. ART1-MECIARP04]
- [32] Platt J. C.. Fast training of support vector machines using sequential minimum optimization. In *Advances in Kernel Methods-Support Vector Learning*, pages 185-208. Cambridge MA: MIT Press, 1998. [[13] Dielhn]
- [33] Polikar, R., Udpa, L., Udpa, S.S., and Honavar, V.: Learn++: An Incremental Learning Algorithm for Supervised Neural Networks, *IEEE Transaction on Systems, Man, and Cybernetics*, 31 (2001) 497-508[4 Wan]
- [34] Ralaivola, L. and d'Alché-Buc, F.: Incremental Support Vector Machine Learning: a Local Approach. *Lecture Notes in Computer Science*, 2130 (2001) 322-329[9 Wan]
- [35] Rüping, S.: Incremental Learning with Support Vector Machines. In: *Proceedings of the IEEE International Conference on Data Mining*, San Jose, CA (2001) [ "[Ruping] 13 Wan]
- [36] Shah, Jagesh V. and Chi-Sang Poon (1999), "Linear independence of internal representations in multilayer perceptrons", *IEEE Trans. on Neural Networks*, vol.10, no.1, pp.10-18. [32. ART1-MECIARP04]

- [37] Shai Fine and Katya Scheinberg. Incremental learning and selective sampling via parametric optimization framework for SVM. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002. [[7] Dietlh]
- [38] Smola A. and B. Scholkopf. On a kernelbased method for pattern recognition regression approximation and operator inversion *Algorithmica* to appear [(Smola et al, 1998)Burge]
- [39] Smola, A., B. Scholkopf, and K.-R. Müller. General cost functions for support vector regression. In L. Niklasson, M. Boden, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks, 1998*. [Smola et al, 1998] [Ruping]
- [40] Smola A. B. Scholkopf and K. R. Muller. General cost functions for support vector regression. In *Ninth Australian Congress on Neural Networks* to appear.[Burge]
- [41] Syed, N., H. Liu, and K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, USA, 1999. [[Syed et al, 1999a] Ruping] 3 Wan
- [42] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995. [Vapnik 1995]
- [43] Thorsten Joachims. Making large-scale SVM learning practical. In B. Schölkof, C. Burges, and A. Smola, editors. *Advances in Kernel Methods-Support Vector Learning*. MIT Press, 1999. [[9] Dietlh]
- [44] Vapnik V. Estimation of Dependences Based on Empirical Data in Russian Nauka Moscow. English translation 1979 Springer Verlag New York [ Vapnick, 1979)Burge] [Burge]
- [45] Wang, H.X., Fan, W., Yu, P.S., and Han, J.W.: Mining Concept-drifting DataStreams Using Ensemble Classifiers. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (2003)* [18 Wan]
- [46] Wilson, Charles L., J. L. Blue and O. M. Omidvar (1997). "Training dynamics and neural network performance", *Neural Networks*, vol.10, no.5, pp.907-923. [33. ART1-MECIARP04]
- [47] Zhou, Z.H. and Chen S.F.: Neural Network Ensemble. *Chinese J.Computers (inChinese)*, 25 (2002) 1-8[15 Wan]
- [48] Zhou, Z.H. and Chen, Z.Q.: Hybrid Decisions Tree. *Knowledge-Based System*, 15(2002) 515-528[Wan]
- [49] Z. Gasca, Aprendizaje Incremental y transferencia de hiperplanos, sin publicar.