Construcción de un árbol balanceado de subclasificadores para SVM multi-clase

Waldo Hasperué [1,2], Laura Lanzarini[1],

[1] III-LIDI, Facultad de Informática, UNLP
[2] Becario CONICET
{whasperue, laural}@lidi.info.unlp.edu.ar

Resumen. Las Máquinas de Vectores de Soporte han demostrado ser una herramienta sumamente útil para resolver problemas de clasificación. Su aplicación en situaciones con más de una clase generalmente implica la combinación de varios subclasificadores. Este trabajo propone un nuevo método para la construcción de una estructura que organice el entrenamiento y uso de estos subclasificadores logrando de esta forma una reducción en el tiempo de respuesta en una clasificación multi-clase. Se trata de un árbol balanceado generado a partir de la estructura de una red neuronal competitiva dinámica. En base a la información de la red se identifican las neuronas relevantes para la separación de clases y se las utiliza para generar el árbol buscado. Cada nodo del árbol permite etiquetar los datos de entrada brindando la información necesaria para entrenar cada subclasificador. Los resultados obtenidos al comparar el método propuesto con otras soluciones existentes han sido satisfactorios.

Palabras Claves. Máquinas de Vectores de Soporte multi-clase, Aprendizaje supervisado, Mapas Auto-Organizativos Dinámicos.

1 Introduction

Las Máquinas de Vectores de Soporte (SVM - Support vector machines) ofrecen un método de clasificación supervisado que produce muy buenos resultados en tareas de clasificación y de reconocimiento de patrones [1]. El algoritmo de entrenamiento de las SVM es un método de aprendizaje supervisado con capacidad para realizar la separación lineal de dos clases. Lamentablemente, los problemas de clasificación del mundo real, por lo general, no son linealmente separables. Por tal motivo, como forma de resolver este aspecto, las SVM utilizan una función kernel que proyecta la información de entrada a un espacio de mayor dimensión donde la separación lineal sea posible [2].

Aunque originalmente las SVM fueron diseñadas para la clasificación entre dos clases, existen varias propuestas para extender dicha clasificación a más de dos clases. El enfoque más utilizado consiste en combinar varios clasificadores SVM binarios [3] [4] [5].

Los clasificadores SVM multi-clase más conocidos y usados son: one-against-one que genera un clasificador para cada par de clases que se quiera separar,

one-against-rest donde se genera un clasificador para cada clase donde cada una de las clases es comparada con las restantes, los que organizan los clasificadores mediante un grafo dirigido acíclico y los que lo hacen utilizando un árbol de decisión [6].

Los dos primeros son los esquemas más simples de implementar pero ambos tienen la desventaja de dejar zonas del espacio de entrada sin clasificación, aunque para resolver este problema se han propuesto algoritmos basados en lógica difusa [7].

De las soluciones comentadas, la implementada por Chaobin et al [6] es una de las mejores ya que propone ir clasificando de a dos clases y de esa manera ir formando un árbol binario de subclasificadores. Si bien esta solución no deja zonas sin clasificar presenta un árbol binario no balanceado donde la profundidad del árbol es igual a la cantidad de clases.

En este trabajo se propone la construcción de un árbol balanceado de subclasificadores. Para esto se utiliza una red neuronal competitiva dinámica. Este tipo de redes posee la característica de detectar las similitudes entre las clases del espacio de entrada y organizar sus elementos de forma que neuronas cercanas dentro de la red representan clases similares.

A partir de la estructura final de la red neuronal y luego de una etapa de "poda", donde se elimina información inútil, se construye un árbol de expansión mínima (minimum spanning tree – MST). Este árbol hallado servirá de base para armar el árbol balanceado de subclasificadores que será usado luego para predecir futuros patrones. La generación de este tipo de árbol a partir del resultado del entrenamiento de una red neuronal con estas características ya fue realizada en [8] con muy buenos resultados lo que demuestra su capacidad para resolver este tipo de problemas.

El aporte central de este artículo radica en la obtención de una estructura de subclasificadores que permita la reducción del tiempo requerido por la SVM multi-clase para efectuar la clasificación. Además, al igual que en el trabajo propuesto por [6], no quedarán zonas del espacio de entrada sin clasificar.

Un árbol balanceado resulta útil en aplicaciones donde hay cientos o miles de clases y el tiempo de respuesta para la predicción debe ser rápido, como por ejemplo las aplicaciones de reconocimiento de locutor, donde la cantidad de clases (personas distintas a reconocer) pueden ser varios cientos e incluso miles y luego la tarea de reconocer a un determinado locutor debe ser lo más rápida posible [9].

El resto del trabajo está organizado como sigue, en la sección 2 se mencionan algunos métodos clásicos de algoritmos SVM multi-clase. En la sección 3 se presenta el método propuesto en este trabajo, en la sección 4 se presentan los resultados obtenidos y en la sección 5 las conclusiones y trabajos futuros.

2 Clasificadores SVM multi-clase

Varios métodos de clasificadores SVM se han propuesto a lo largo del tiempo para extender el buen desempeño del SVM tradicional.

One-against-all construye *n* subclasificadores, donde *n* es el número de clases del conjunto de *p* patrones de entrenamiento. El *i*-ésimo subclasificador se entrena con los patrones de la clase *i* con valor 1 y el resto de patrones del resto de las clases con -1. Este algoritmo entrena *n* SVM utilizando en cada entrenamiento los *p* patrones de

entrenamiento. La gran desventaja de esta opción de clasificación es que pueden quedar zonas sin clasificación.

One-against-one construye [*n**(*n*-1)]/2 subclasificadores, donde *n* es el número de clases. Este algoritmo utiliza un método de votación al momento de clasificar a que clase pertenece una muestra. La gran cantidad de subclasificadores que se crean y el hecho de dejar zonas sin clasificación son las desventajas de este método.

El método que utiliza un grafo dirigido acíclico (directed acyclic graph) construye [*n**(*n*-1)]/2 sub-clasificadores donde *n* es el número de clases. Este método si bien no deja zonas sin clasificar necesita evaluar *n*-1 subclasificadores para determinar a que clase pertenece una muestra determinada.

El método propuesto en [6] va dividiendo las clases de a una contra las restantes, construyendo para esto un árbol binario. Así, el nodo raíz divide una clase *i* del resto de clases, con las cuales se forma otro subárbol binario, este a su vez separa una segunda clase *j* de las restantes, las cuales a su vez se utilizan para formar otro subárbol binario; de esa manera se va separando clase por clase hasta llegar al nodo *n*-1 que separa las últimas dos clases. La ventaja de este método es que construye solo *n*-1 subclasificadores cubriendo todo el espacio de entrada y de esa manera no quedan zonas sin clasificación. En el mejor de los casos evaluando un solo subclasificador (el nodo raíz del árbol de subclasificadores) es suficiente para determinar la clase de una muestra dada y en el peor de los casos se evalúan los *n*-1 clasificadores. En promedio son necesarios (*n*-1)/2 subclasificadores para determinar la clase de una muestra.

Este y otros métodos basados en árboles tienen la desventaja de acumular errores en los nodos superiores, por eso que es mejor hacer las primeras separaciones entre aquellas clases que son “más claramente separables”. De esta manera se reducen los errores en los nodos superiores y se mejora la eficacia en la clasificación.

3 Construcción del árbol balanceado de subclasificadores

En este trabajo se propone utilizar una red neuronal competitiva dinámica para encontrar la separación de clases y descubrir como es la relación de vecindad entre ellas en el espacio de entrada. A partir de esta relación de vecindad se construye un árbol balanceado para los subclasificadores.

Las redes neuronales competitivas se caracterizan por estar formadas por neuronas artificiales que, como su nombre lo indica, “compiten” entre sí por la representación de los patrones de entrada. Para ello, cada neurona tiene asociado un vector de pesos contra el cual se compara cada uno de los datos de entrada utilizando una medida de similitud dependiente del problema. Estos vectores de pesos son aprendidos por la red a través de un proceso de entrenamiento no supervisado.

Una de las redes neuronales competitivas más famosas es el Self Organizing Map (SOM) definido por Kohonen [10]. Esta red, si bien ha dado resultados exitosos en numerosas situaciones, tiene como desventaja el uso de una arquitectura estática ya que la cantidad de neuronas que forman la arquitectura así como su modo de conexión deben ser definidos a priori, antes de comenzar con el entrenamiento, condicionando de esta manera la eficiencia y eficacia de la red.

Como forma de resolver esto se han propuesto soluciones alternativas denominadas Mapas Auto-organizativos Dinámicos, también conocidas como DSOM (Dynamic Self Organizing Maps) los cuales conservan la capacidad de preservar adecuadamente la topología de los datos permitiendo la incorporación y/o eliminación de elementos durante el entrenamiento. De esta forma no es necesario indicar a priori la cantidad de neuronas a utilizar ya que la arquitectura es de dimensión variable.

El método propuesto en este artículo propone entrenar un SOM dinámico denominado AVGSOM [11] para descubrir la relación de vecindad entre las distintas clases en el espacio de entrada y además determinar su grado de separación. Luego del entrenamiento se utilizará la estructura interna resultante como punto de partida para construir el árbol de subclasificadores balanceado. El método AVGSOM fue utilizado con resultados interesantes en varios tipos de problemas [8][9][12] ya que su estructura comienza con solo cuatro neuronas y luego va creciendo en la dirección que necesite acomodándose a la topología de los datos de entrada.

El método propuesto en este trabajo consiste en cuatro etapas:

Etapa 1: Se realiza el entrenamiento del SOM dinámico.

Etapa 2: Se poda la estructura del SOM entrenado eliminando la información (neuronas y conexiones entre neuronas) innecesaria para la detección de vecindad de clases en el espacio de entrada.

Etapa 3: Se construye el Arbol de Expansión Mínima con las neuronas resultantes de la etapa 2.

Etapa 4: Preparar la estructura del árbol de subclasificadores.

Finalmente existe una quinta etapa que consiste en entrenar los subclasificadores, pero que no es objetivo de este trabajo.

En la figura 1.a) se muestra un ejemplo hipotético con una imagen bi-dimensional la cual posee puntos que pertenecen a 10 clases distintas. Este ejemplo se usará para ilustrar las distintas etapas del método propuesto.

3.1 Entrenamiento del SOM dinámico

En primer lugar se entrena un SOM dinámico utilizando el conjunto completo de patrones de entrada. Si bien este algoritmo de aprendizaje es no supervisado utiliza de manera implícita la información y topología de los patrones en el espacio de entrada para acomodar su estructura interna a una buena representación de los mismos. Es de esperar que, si existen dos cúmulos de patrones correspondientes a dos clases separadas, también hay dos grupos bien identificados de neuronas que mapean a cada una de las clases. De esa manera, las neuronas que son vecinas directas dentro de un mismo cúmulo estarán mucho más cerca entre si (midiendo distancia euclídea por ejemplo) que aquellas neuronas vecinas directas que mapean a distintas clases. La figura 1.b) muestra el entrenamiento final del ejemplo de la figura 1.a).

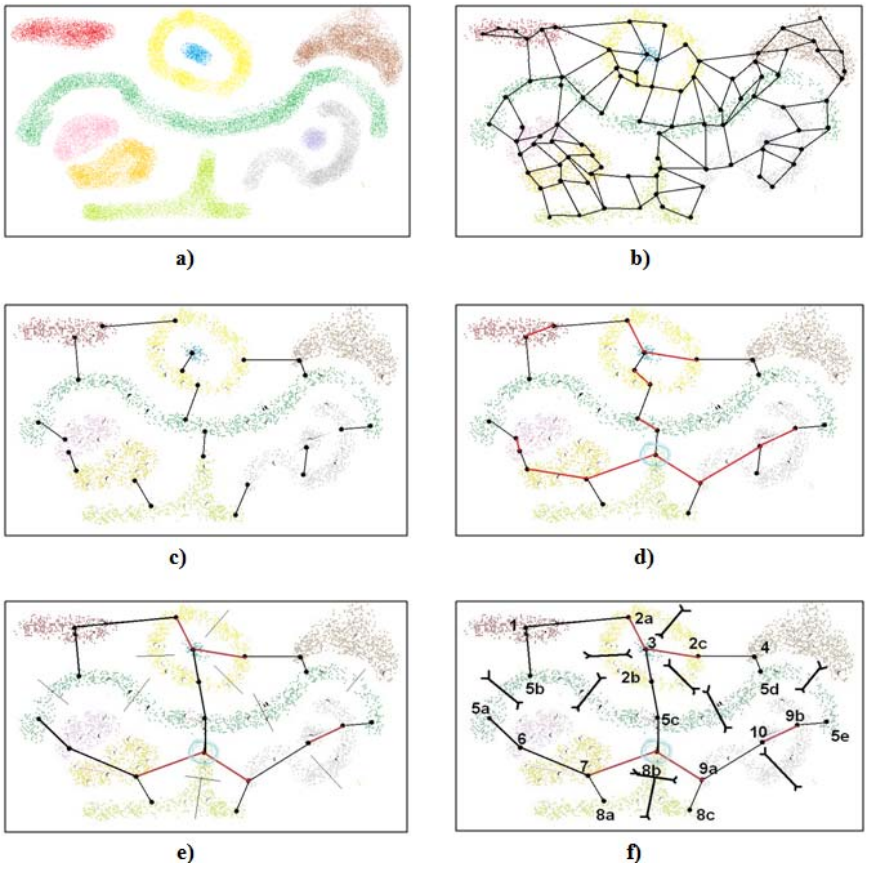

Fig. 1. a) Ejemplo de estudio de una imagen bi-dimensional con 10 clases. b) Red AVGSOM entrenada con el conjunto de patrones de entrada, c) Las neuronas y aristas (con menor D_{ij}) que unen más de una clase. d) El árbol de expansión mínima. El círculo identifica el nodo que hace que el árbol quede con un óptimo balanceo, e) El árbol final luego de la eliminación de 6 nodos y 6 aristas, f) La separación de subclases y lo que corresponde a cada neurona. Así la clase 2 se divide en 3 subclases, la 5 en 5, la 8 en 3 y la 9 en 2.

3.2 Poda de la estructura del SOM dinámico

Finalizado el entrenamiento del SOM se usa la información de la vecindad entre neuronas para detectar la separación de clases. Para cada neurona *i* se calcula el Q_{ic}, la cantidad de patrones de la clase *c* que caen dentro de la región de Voronoi de esa neurona. Así para cada neurona y para cada clase habrá un valor Q_{ic} donde vale cero cuando para esa neurona no hay patrones de la clase *c* dentro de su región de Voronoi.

Si dos neuronas *i* y *j* vecinas entre si tienen al menos un patrón de clases distintas significa que dichas neuronas están mapeando la separación de dos o más clases. Esto

ocurre cuando $Q_{ic} > 0$ y $Q_{jd} > 0$ donde c y d son clases distintas. Detectados estos casos se realiza lo siguiente, entre todos los patrones mapeados por las neuronas i y j se calcula la distancia euclídea mínima D_{ijcd} entre los patrones representados por i y los representados por j y que a su vez sean de clases distintas c y d; a ese par de neuronas (la arista) se le asigna el valor D_{ijcd}.

Así, se arma para cada par de neuronas vecinas y para cada clase que estén mapeando el valor de separación de clase. Aquellos pares de neuronas donde no tengan valores de separación de clases se eliminan; $D_{ijcd} = 0$ para todo c y d. Y aquellas neuronas que quedaron sin conexiones también se eliminan. Por último, de los distintos pares de neuronas para la mismas dos clases c y d solo se guarda la arista (relación directa entre dos neuronas vecinas) con el menor valor (Figura 1.c); $\min(D_{ijcd})$ para todo i y j.

3.3 Armado del árbol de expansión mínima

Con las neuronas y aristas resultantes de la etapa 2 se completa un árbol de expansión mínima sin eliminar las aristas guardadas. Luego, para el árbol resultante se busca el nodo n que hace que, si n fuera el nodo raíz, el árbol quede lo más balanceado posible (Figura 1.d). Para ello, durante el armado del árbol de expansión mínima se guarda para cada arista un peso que en realidad es el número de aristas que hay desde ella hasta un nodo terminal. Así el nodo raíz es aquel cuyas aristas tengan el menor peso posible en cada una, esto se hace buscando nodo por nodo aquel que tenga todas las aristas de menor peso.

Si luego de armado el árbol quedan pares de neuronas unidas y las neuronas representan a la misma clase, se juntan eliminando así un nodo y una arista (Figura 1.e). Con esta acción no se elimina información y además se reduce la complejidad del árbol.

Para las clases que tienen más de una neurona se divide la clase en tantas subclases como neuronas haya y cada subclase tendrá los patrones representados por su respectiva neuronas (Figura 1.f).

3.4 Armado de la estructura final del árbol de subclasificadores

Una vez determinado quien será el nodo raíz del árbol comienzan a construirse los subclasificadores. Si el nodo raíz solo está unido con un único nodo, entonces el subclasificador del nodo raíz del árbol tendrá una rama que será la región del nodo raíz y la otra rama el resto del subárbol. Si el nodo raíz está unido a dos nodos entonces se genera un subclasificador con una rama que será la región del nodo raíz y la otra rama será un segundo subclasificador entre las dos ramas del árbol. Si el nodo raíz está unido a tres nodos (ejemplo de la figura 1.e) entonces se genera un subclasificador que tendrá una rama con un subclasificador entre la región del nodo raíz y una de las tres ramas del árbol, y una segunda rama con un subclasificador entre las otras dos ramas del árbol. La figura 2 muestra un ejemplo de estos casos. Este procedimiento puede generalizarse para casos donde el nodo raíz esté conectado a más de tres nodos.

Luego, con los nodos restantes se continúa de la siguiente manera. Si el nodo une solo dos aristas, una de ellas es la arista que la une con el nodo padre del árbol de clasificadores, entonces se crea un subclasificador entre la región que representa ese nodo y el resto de la rama. Si el nodo se une con tres nodos, donde uno de ellos es la arista que lo une en el árbol de subclasificadores, entonces se crea un subclasificador entre la región del nodo y las otras dos ramas, para estas ramas se genera otro subclasificador (figura 3). Este procedimiento puede ser generalizado fácilmente para casos donde el nodo esté unido a más de tres nodos.

Así en el ejemplo de la figura 1.f el árbol de clasificadores resultante es el mostrado en la figura 3. Como se puede ver la profundidad del árbol es 6. Si al momento de usar el árbol como predictor la clase resultante es, por ejemplo una de las clases 5a, 5b, 5c, 5d o 5e entonces se puede afirmar que el patrón es de la clase 5.

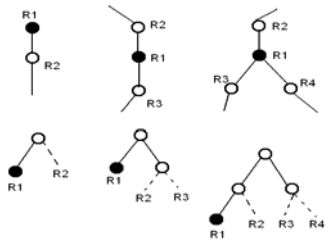

Fig. 2. Tres ejemplos de cómo se transforma desde el nodo raíz (R1) del árbol de regiones y neuronas al nodo raíz del árbol de subclasificadores.

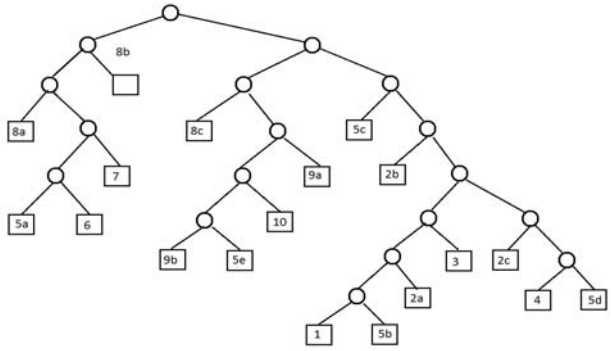

Fig. 3. La estructura final del árbol de subclasificadores del ejemplo de 10 clases de la figura 1.f.

Por último se entrenan por separado cada uno de los subclasificadores (nodos no-hojas del árbol de subclasificadores). Cada subclasificador puede usar una función kernel así como los parámetros que más se adecuen para producir un resultado óptimo. Los subclasificadores de los niveles más bajos solo se entrenan con los patrones de las regiones que representan, por ejemplo en la figura 3, el nodo inferior de la izquierda que tiene como región a 5a y 6 se entrenan únicamente con los patrones de la subclase 5a y la clase 6, el nodo padre de este se entrena con los patrones de la clase 5a, 6 y 7. A medida que se va subiendo niveles en el árbol, los subclasificadores se van entrenando con todos los patrones de las regiones hojas, al entrenar se etiquetan con valores positivos los patrones de la región de una rama y negativos los patrones de las regiones de la otra.

4 Resultados

El algoritmo propuesto en este artículo fue evaluado con imágenes bi-dimensionales como el de la figura 1. Para el caso particular de la figura 1.a la red inicial contenía 88 neuronas, de las cuales solo quedaron 19 después de la poda. Hubo 2 clases con una única neurona y la clase 5 tenía 5. La rama de mayor profundidad del árbol es de 7 niveles para las 10 clases.

La mejor estructura que puede obtenerse con este método es un árbol binario completamente balanceado por lo que para predecir la clase de una muestra dada se necesita evaluar $\log_2(n)$ subclasificadores. En el peor de los casos, en donde el árbol de expansión mínima resultante sea una lista se necesitarían evaluar $n/2$ subclasificadores.

En un caso donde haya que entrenar una SVM multi-clase con p patrones pertenecientes a n clases y suponiendo que cada clase tenga k patrones, el método one-against-all crea n subclasificadores. Cada subclasificador es entrenado con los p patrones, k patrones positivos y p-k patrones negativos. Al momento de predecir se necesitan evaluar un subclasificador en el mejor de los casos y n-1 en el peor, con $n/2$ de promedio.

One-against-one crea un subclasificador por cada par de clases, en total $n*(n-1)/2$ subclasificadores, cada subclasificador es entrenado con los patrones de las dos clases involucradas, es decir entrena más subclsificadores que one-against-all pero cada uno es entrenado con menos patrones. Al predecir siempre se evalúan los n subclasificadores, decidiendo por votación la clase de la muestra.

El método basado en un grafo dirigido acíclico (directed acyclic graph) crea $n*(n-1)/2$ subclasificadores, formando un grafo de n-1 niveles. El subclasificador raíz (nivel 0) se entrena con los p patrones, los dos nodos de nivel 1 se entrenan con p-k patrones cada uno, los tres nodos del nivel 2 se entrenan con p-k*2 patrones cada uno, en general un nodo de nivel v se entrena con p-k*v patrones, habiendo en cada nivel v+1 subclasificadores. En total el grafo tiene n-1 niveles. Siempre se evaluan n-1 subclasificadores al momento de predecir una muestra.

El método del árbol binario propuesto en [6] crea n-1 subclasificadores, el nodo raíz (nivel 0) se entrena con los p patrones, el siguiente nodo (nivel 1) con p-k, el de nivel 2 con p-k*2, etc. En general un subclasificador de nivel v se entrena con p-k*v

patrones, habiendo n-1 niveles en el árbol. Al momento de evaluar en el mejor de los casos hace falta solo evaluar un subclasificador y en el peor de los casos n-1, en promedio $n/2$.

El método propuesto en este artículo necesita crear n-1 subclasificadores en el mejor de los casos y (n+l-1) en el peor de los casos, donde l es la cantidad de subclases que se debieron crear. El nodo raíz (nivel 0) se entrena con los p patrones, los dos nodos del nivel 1 con $p/2$ patrones, los nodos del nivel 3 con $p/4$ patrones, etc. En general un nodo de nivel v se entrena con $p/(2^v)$. Al momento de evaluar solo son necesarios $\log_2(n)$ subclasificadores en el mejor caso y $n/2$ en el peor.

Si bien es muy difícil demostrar que un algoritmo se entrena más rápido que otro ya que depende de muchos factores, intuitivamente puede deducirse que el método propuesto lleva menos tiempo de entrenamiento que el resto, tal como lo demuestra la tabla 1. Lo que efectivamente puede afirmarse es que, en el peor de los casos, al momento de predecir una muestra, la cantidad de subclasificadores a evaluar es la misma que en el promedio del método de Chaobin et al. [6] y mucho mejor que en los métodos restantes.

Tabla 1. Tiempos de entrenamiento (min.) comparativos sobre ejemplos bidimensionales entre el método propuesto y tres de los métodos vistos en el trabajo.

Imagen	One-against-one	One-against-all	Árbol binario	Este trabajo
10 clases	0.76	12.59	2.00	1.50
20 clases	3.22	86.02	25.86	4.48
50 clases	17.62	584.95	52.43	17.02
100 clases	82.84	3977.71	115.88	39.10
1000 clases	389.37	27048.48	256.10	90.62

5 Discusión y trabajos futuros

El algoritmo propuesto en este artículo resultó muy útil para detectar clases en imágenes bidimensionales reduciendo el número de subclasificadores utilizados al momento de predecir una muestra por la SVM multi-clase. Este tipo de soluciones ofrece una buena alternativa a problemas donde hay muchas clases.

Actualmente los autores de este trabajo están utilizando redes neuronales competitivas dinámicas para identificar personas por su señal de voz [9][13]. Este problema puede llegar a tener numerosas clases por lo que se espera que utilizar una SVM multi-clase, cuyos subclasificadores se organicen a través del árbol descripto en este trabajo, permita reducir considerablemente el tiempo de respuesta del clasificador. Esto puede lograrse ya que la cantidad de subclasificadores necesarios tiende a ser de orden $\log_2(n)$ siendo n el número de clases.

En relación a este mismo problema de identificación de personas a través de su señal de voz, actualmente se está estudiando la manera de detectar grandes “vacios” entre clases para poder mapearlos como zonas sin clasificar y así identificar posibles nuevos locutores. También resulta de interés lograr incorporar nuevas personas con un tiempo de entrenamiento reducido.

Finalmente, otra línea de investigación que resulta de interés es el desarrollo e implementación de un SOM dinámico no convencional de más de dos dimensiones. Esto permitiría que la estructura interna de la red se acomode de manera óptima a las clases en el espacio de entrada, cuando la dimensión es muy alta.

References

1. Vapnik V. N.: The Nature of Statistical Learning Theory. New York. (1995)
2. Vapnik V. N.: An overview of statistical learning theory. IEEE Transaction Neural Networks 10, 88-999 (1999)
3. Jiayang L., Jianhua X.: A Fast Multi-label Classification Algorithm Based on Double Label Support Vector Machine. In: International Conference on Computational Intelligence and Security, pp. 30-35 (2009)
4. Juntao L., Yingmin J., Junping D., Fashan Y.: Adaptive multi-class support vector machine for microarray classification and gene selection. In: ICCAS-SICE. pp. 2658 – 2663 (2009)
5. Guo J., Takahashi N., Wenxin H.: An Efficient Algorithm for Multi-class Support Vector Machines. In: Advanced Computer Theory and Engineering. pp. 327-332 (2008)
6. Chaobin L., Yuexiang Y., Chuan T.: An Improved Method for Multi-class Support Vector Machines. In: International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). pp. 504 – 508 (2010)
7. Zhuoda J.: Support vector machines for multi-class pattern recognition based on improved voting strategy. In: Control and Decision Conference (CCDC). pp. 517 - 520 (2010)
8. Hasperue W., Corbalán L., Lanzarini L., Bría O.: Skeletonization of Sparse Shapes using Dynamic Competitive Neural Networks. Inteligencia Artificial 11, 33-42 (2007)
9. Estrebou C., Hasperué W., Lanzarini L.: Voice Recognition based on vote-SOM. In: XXVIII Jornadas Chilenas de Computación. JCC. pp. 89-93 (2009)
10. Kohonen, T. Self-Organizing Maps. 2nd Edition. Springer. ISSN 0720-678X. 1997.
11. Hasperué W., Lanzarini L.: Dynamic Self-Organizing Maps. A new strategy to upgrade topology preservation. In: XXXI Congreso Latinoamericano de Informática CLEI. pp. 1081-1087 (2005)
12. Hasperué W., Lanzarini L.: A New Clustering Strategy for Continuous Data Datasets Using Hypercubes. In: XXXVI Congreso Latinoamericano de Informática CLEI (2010)
13. Estrebou C., Lanzarini L., Hasperué W.: Voice recognition based on probabilistic SOM. In: XXXVI Congreso Latinoamericano de Informática CLEI (2010).