A mis padres y mi hermana

AGRADECIMIENTOS

Me gustaría para comenzar este estudio, agradecer a todas las personas que han hecho posible que estos 3 años de estudio hayan valido la pena. Me gustaría empezar por mi familia, mis padres Miguel y Ana que tanto me han dado y que han esto apoyándome de manera incansable, moral y económicamente, gracias a ellos estoy donde estoy ahora. No me puedo olvidar de mi hermana, Ana María, muchas veces ella me ha servido de inspiración y de fuerza de empuje.No me olvidare jamas de mis amigos y compañeros de piso, con los que tanto me he divertido y que han sido los que al fin y al cabo han estado mas cerca en los momentos difíciles. Entre otros Adrián S., Pau, Jean Paul, Adrián B. ,Javi, etc.

Jamas podría olvidarme de una personita, Lucia, que en este ultimo año, ha sido, mas que un pilar de apoyo en los buenos y en los malos momentos, sin ella, esto probablemente no habría sido posible, a ti te digo, muchas gracias!

Tampoco podrían faltar los profesores que tanto me han enseñado y de los que tanto he podido aprender, desde el primer año Dr. Anna Puig, Dr. Maite López, Dr. Maria Salamó, pasando por el segundo, Dr. Manel Puig, Dr. Xavier Jarque y finalmente el tercero, Dr. Oriol Pujol, Dr. Petia Radeva y un largo etcétera que no acabaría en esta pagina.

Para finalizar dar muchas gracias a mis directores de proyecto Dr. Sergio Escalera y Dr. Xavier Baró, han sabido motivarme como al que mas y sacar lo mejor de mi. He recibido un trato envidiable por su parte, solo espero demostrarles mi agradecimiento con el trabajo que se ha llevado a cabo.

A todos gracias,

Miguel Angel Bautista Martín

Resumen

En este proyecto se estudian las alternativas del estado del arte en los problemas de multi-clasificacion y se proponen nuevas estrategias para tratar dichos problemas, las cuales posteriormente serán aplicables a problemas de alta dimensionalidad. Se introducen nuevas estrategias de optimización de clasificadores y de codificacion de ECOC, las cuales son probadas con resultados ampliamente satisfactorios. Finalmente se concluye con líneas de investigación que podrían resultar interesantes como futura aplicación de este proyecto.

Resum

En aquest projecte s'estudien alternatives a l'estat de l'art en problemes de multi-classificació i es proposen noves estrategies per a adreçar-se a aquest tipus de problemes, les quals posteriorment també podran ser útils en problemes d'alta dimensionalitat. Noves estratègies d’optmització de classificadors i codificació ECOC son introduïdes, obtenint resultats àmpliament satisfactoris. Finalment, l'estudi es concluït amb línies d'investigació que podrien ser interessants com futures aplicacions d’aquest projecte.

Abstract

In this project we analyze alternatives to the state-of-the-art on multi-class classification problems and novel strategies to address those problems are proposed. These strategies will be applied over high dimensional data. Novel strategies on classifier optimization and ECOC codification are also introduced, obtaining very satisfying results. Finally, this study is concluded with the discussion of research lines that could be useful for future applications.

INDICE

1.Introducción

1.1 Problema y motivación

A lo largo de la historia, los problemas propuestos a resolver por los métodos de automatización se han centrado principalmente en los llamados “problemas de clasificación”. El término “clasificación” se ha utilizado para denotar el reconocimiento y agrupación de objetos que pueden estar representados de distinta forma dentro de un determinado sistema. Dichos objetos suelen estar definidos a través de un conjunto de clases o categorías, sobre las cuales se debe hacer un previo aprendizaje para proceder a su posterior reconocimiento.

En un principio podríamos pensar que estos sistemas podrían moverse en un espacio con un conjunto binario de clases (dos categorías a distinguir). De aquí surge toda la teoría y estudio de la clasificación binaria. A simple vista podemos ver que se necesitaría alguna herramienta para trabajar con una clasificación que no fuese simplemente binaria, ya que el mundo tiene inherentemente un funcionamiento multi-clase. De esta manera pasaríamos a clasificar un objeto entre N posibles clases definidas en nuestro sistema.

Además, con el gran incremento de la información experimentado en los últimos años y el trepidante aumento del tráfico en la red, nos estamos acercando a la clasificación de miles de clases. Estos problemas de alta dimensional de datos son los llamados “Large Escale Problems”, problemas en los cuales el conjunto de datos a manejar es enormemente amplio y donde la búsqueda exhaustiva o analítica de una solución sería computacionalmente inviable. Es aquí donde entra la potencia de la inteligencia artificial y en concreto del aprendizaje automático y de la computación evolutiva.

En concreto, dentro del aprendizaje automático, para el desarrollo de este proyecto nos hemos centrado en el aprendizaje supervisado, el cual se basa en obtener una función que, para cada tipo de

entrada (características inherentes de los objetos) se obtiene un tipo o etiqueta diferente en la salida. Es así como el aprendizaje automático nos ofrece una poderosa arma para poder obtener soluciones a los problemas de multi-clasificación.

La evolución y la genética por otra parte llevan desde que se estableció la vida en este planeta resolviendo el problema de la adaptación de los seres vivos al medio en el que se desenvuelven [1]. La evolución se podría ver desde un punto de vista computacional como una manera de abordar el proceso resolutivo de un problema, de una manera no es exhaustiva. Esta manera de abordar el problema consiste en la proposición que tiempo atrás se postuló en [1], en la que aseguraba que cada generación de la especie estaba mejor adaptada al medio que la generación anterior y peor adaptada que la generación siguiente. De esta manera podríamos concebir nuestro proceso resolutivo como un proceso que parte con un conjunto de soluciones aleatorias válidas, que con el paso de generaciones y su consiguiente apareamiento y mutación, producen a cada generación soluciones mejores (mas adaptadas al medio).

En este proyecto proponemos una aproximación a la solución de los problemas de multi-clasificación con múltiples datos y que en futuros estudios nos servirá para atacar problemas de muy alta dimensionalidad, como los llamados “Large Escale Problems”.

1.2 Estado del arte

Como se puede imaginar, el problema propuesto ha sido objeto de estudio por varios investigadores en los últimos años, sin lograr una solución clara y viable. En este apartado vamos a revisar los trabajos que abordan este tipo de problemas.

En un principio se optó por utilizar diversos “frameworks” para la resolución del problema de la multi-clasificación: agentes inteligentes [2], redes neuronales [3], etc. Finalmente estudios recientes

han demostrado que el "framework" de los "Error Correcting Output Codes" (en adelante ECOC) representa significantes mejores cuando se trabaja con diseños dependientes del problema.

Este sistema ha sido ampliamente aplicado a problemas como reconocimiento facial [4], reconocimiento de texto [5], etc.

Este marco de trabajo consiste en dos etapas básicas: en la primera, a cada clase se le asigna una palabra codificada, esta palabra codificada distingue cada clase unívocamente; en la segunda etapa, denominada decodificación, cada ejemplo a clasificar (cada nuevo objeto en el espacio de nuestro sistema) se compara con cada palabra codificada previamente, para dar una predicción de clasificación.

Una de las codificaciones más conocidas se denomina *One vs. All*, a priori una de las más eficientes para resolver este tipo de problemas y que consiste en distinguir una clase del resto de clases posibles.

Este sistema codifica N clasificadores para N clases. Aun así, trabajando con dicho número de clasificadores los resultados obtenidos muchas veces no han pasado de lo que se denomina como "Random classification level" que se conoce como $1/N$. Es decir, el margen que transciende de la mera clasificación aleatoria a la clasificación dada como aprobada es $1/N$ donde N es el número de clases.

Otra manera de abordar dicho problema por parte de algunos investigadores ha sido utilizar un tipo de clasificadores conocidos como "Online Classifiers". Este tipo de clasificación aprende un ejemplo a la vez y tiene como punto a favor que implementa un tipo de retroalimentación que hace que el propio sistema re-adapte su comportamiento con nuevos ejemplos. Así, podríamos decir que los "Online Classifiers" tienes tres pasos básicos: el primero sería la recepción por parte del algoritmo de clasificación del ejemplo a clasificar; el segundo paso sería la predicción de la etiqueta

de dicho ejemplo a clasificar; y el tercero seria el innovador en el que el algoritmo recibe la etiqueta real de dicho ejemplo y utiliza este "feedback" para minimizar el error en futuras pruebas. La contra de dicho sistema es que se le ha de proporcionar la etiqueta real de cada ejemplo a clasificar en un tiempo no muy largo, y actualmente es difícil diseñar un sistema que aprendiendo de forma online de un rendimiento similar a los clasificadores offline. Esto sugiere que podría ser un buen algoritmo para situaciones de evolución continua y sistemas en tiempo real en los que en un corto espacio de tiempo se puede saber cuál va a ser la clasificación del ejemplo que se le ha dado al algoritmo. Por ejemplo, en [6] se demostró la aplicación satisfactoria de este sistema en la predicción de la dirección del movimiento del mercado financiero.

Otra rama que muchos investigadores han explorado para mejorar los problemas el resultado de los problemas de multi-clasificación han sido la redes neuronales. Uno de los sistemas de redes neuronales que se utilizan en problemas de multi-clasificación es el MLP (Multi-layer Perceptron). Este sistema se basa en el principio de las redes neuronales básicas, en las que existen tres capas: la capa de entrada, la capa oculta y la capa salida. En este sistema la salida de cada neurona de la capa i es entrada de todas las neuronas de la capa i+1.

Destacar que cada capa tiene su tarea en este sistema: la capa de entrada se encarga únicamente de recibir los patrones (ejemplos a clasificar) que se procesaran en el sistema, la segunda capa (la capa oculta) es la capa donde se lleva a cabo todo el procesamiento de dichos patrones y la capa de salida es la capa donde se obtiene las predicciones para los patrones o ejemplos a clasificar. El algoritmo utilizado para el entrenamiento de estas redes se denomina propagación hacia atrás (del inglés "backpropagation"). Este algoritmo trata de minimizar un error, comúnmente cuadrático, utilizando el algoritmo del gradiente descendiente. De aquí obtenemos uno de los primeros problemas al utilizar este sistema, y es que el gradiente descendiente puede estancarse fácilmente en mínimos

relativos y no conseguir los mejores resultados posibles. No obstante, pese a obtener resultados robustos en ciertos problemas de clasificación [7], este método sufre de problemas de escalabilidad.

1.3 Propuesta de resolución

Finalmente, después de comprender y estudiar las anteriores alternativas analizadas en este campo, nos declinamos por un tipo de solución, con la que se espera llegar a obtener una serie de resultados que mejoren ampliamente lo existente.

Principalmente el sistema está basado en la potencia de la clasificación binaria que nos ofrece el aprendizaje automático, y más en concreto las máquinas de vectores de soporte (en adelante SVM, del inglés “Support Vector Machines”). Este mecanismo nos ofrece un alto rendimiento en lo que a clasificación binaria se refiere.

Como habíamos comentado anteriormente, para abordar problemas de multi-clasificación, el framework del los ECOC es uno de los mejores en términos de combinación de clasificadores binarios y de rapidez de cómputo [8]. En este sentido hemos diseñado un nuevo ECOC que permite definir un número reducido de clasificadores binarios para abordar problemas “large-scale”. El ECOC-Sublineal contiene $\log_2 N$ clasificadores para distinguir N clases de forma unívoca.

Finamente, el último pilar de nuestro sistema será la computación evolutiva, y en concreto los algoritmos Genéticos, que nos ofrecen una poderosa arma en lo que a proceso de búsqueda de soluciones se refiere.

Así, finalmente denominaríamos a nuestro sistema como “SE-ECOC” del inglés “Sublinear Evolutive Error Correcting Output Codes”.

Para demostrar la fiabilidad y la viabilidad del método, se ha decidido realizar una batería de test sobre algunos datasets del UCI Machine Learning Repository. En concreto han sido seleccionados 4

datasets, descritos en el apartado de resultados. Los resultados obtenidos serán comentados y analizados en capítulos posteriores.

1.4 Organización

Este estudio se organiza por capítulos: en el primer capítulo se hace una pequeña introducción historia y se presentan tanto los problemas a tratar como la solución diseñada.

En el segundo apartado se tratan profundamente las cuestiones técnicas de desarrollo analizando las propuestas del estado del arte en cada tópico y haciendo especial hincapié en la razón de buscar nuevas estrategias cuando es necesario. El apartado concluye con diagramas, planificación y costes relativos al proyecto.

En el tercer apartado se describe la batería de pruebas desarrollada para mostrar la eficiencia del sistema. Se muestran y analizan los resultados obtenidos. Finalmente en el último apartado se concluye el estudio sopesando cuales son los puntos fuertes del proyecto y nombrando algunas líneas de investigación futura que serian interesantes. En los apéndices se incluye un CD con el código del proyecto junto con la memoria.

2.ECOC-SVM Sublinear Evolutivo

En este apartado se presenta el esquema de clasificación evolutivo basado en un nuevo diseño de los códigos correctores de errores.

2.1 ECOC "Error Correcting Output Codes"

Como se ha comentado en el apartado anterior muchos estudios han concluido que el framework de los ECOC representa una forma robusta de combinar clasificadores débiles para formar una clasificador multi-clase robusto [9].

En los problemas de multi-clasificación tal y como los hemos descrito anteriormente, el objetivo es agrupar cada nuevo ejemplo que recibe el sistema en su clase asociada. Inicialmente se puede pensar que bastaría con una función (en adelante denominada "clasificador") que supiese distinguir entre las *N* clases que previamente a entrenado el sistema [10]. No obstante, se observa que esta manera de atacar el problema no es eficiente cuando el espacio del sistema tiene una dimensionalidad elevada.

Por otra parte lo que se propone en el esquema de trabajo de los ECOC es utilizar clasificadores binarios, es decir, que aprenden a distinguir entre dos clases. A simple vista se puede ver que para un problema de multi-clasificación, un simple clasificador binario no será suficiente, sino que se tendrá que utilizar un conjunto de clasificadores binarios.

Así, se puede concebir un ECOC como una matriz, en la cual las filas pertenecerán a las clases que aprende a distinguir el sistema y las columnas a los distintos clasificadores binarios que distinguen entre (y aquí está la clave) dos conjuntos de clases.

En conclusión, el marco de los ECOC nos proporciona una potente arma para atacar el problema de multi-clasificación combinando clasificadores binarios. Por ejemplo, en la tabla inferior se codifica un ECOC con 9 clases y 9 clasificadores binarios, donde cada columna C_i es un clasificador.

Clase	C1	C2	C3	C4	C5	C6	C7	C8	C9
1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0	0
5	0	0	0	0	1	0	0	0	0
6	0	0	0	1	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0

Tabla 1,ECOC de 9 clases con 9 clasificadores.

El marco de los ECOC se divide en dos etapas. Dichas etapas, codificación y decodificación, determinan la eficacia de un diseño ECOC. A continuación se revisa el estado del arte de estas dos etapas para justificar la selección de nuestro nuevo diseño.

2.1.1 Codificación de los ECOC

La primera etapa es la de codificación, en la que a cada clase se le asocia una palabra clave de bits que distingue cada clase unívocamente. Así pues, como se comenta en [11], una buena codificación para un problema de *N* clases deberá satisfacer estas dos propiedades:

- Cada palabra clave deberá estar correctamente separada en términos de distancia Hamming

(diferencia en los valores de los bits) con el resto de palabras clave.

- No deben existir dos clasificadores binarios los cuales aprendan a discernir entre la misma agrupación de clases.

En lo que a etapa de codificación se refiere, se conocen varias alternativas a la hora de obtener las palabras clave de cada clase.

Las alternativas estándar en la etapa de codificación son : “OneVsAll” (Uno contra todos) y “Dense Random” (Codificación Aleatoria).

La estrategia OneVsAll es la más utilizada. Consiste en entrenar cada clasificador para distinguir una clase del resto de las clases. Dadas *N* clases, esta estrategia establece un numero de *N* clasificadores, es decir, tiene una longitud de palabra clave de *N* bits. Como se puede observar para problemas de alta dimensionalidad se han de entrenar el mismo número de clasificadores como clases ha de aprender a distinguir. Así, en [12] se demuestra que la computación exhaustiva de la matriz de codificacion es NP-Hard con el número de clases.

Por otra parte, tenemos la alternativa “Dense Random”, en la que se generan un alto número de matrices de codificación de longitud *n*, donde los valores binarios (1 y 0) que veíamos en la tabla anterior (1 y -1 en adelante) tienen la misma probabilidad de aparecer. Estudios sobre el rendimiento de dichas matrices de codificación han sugerido una longitud de $n = 10 \log N$, donde *N* es el número de clases. Así, mediante esta técnica de codificación se consigue una notable mejora en lo que a número de clasificadores se refiere y en lo que repercute en el tiempo de cómputo global. Para el conjunto generado de matrices, la matriz óptima es la que maximiza la distancia Hamming, teniendo en cuenta que cada columna ha de tener los dos valores (1 y -1).

Recientemente también han surgido estudios en los cuales se ha incorporado un tercer símbolo en el diseño de los ECOC, el símbolo cero, que implica que implica que determinadas clases no son

consideradas en las agrupaciones para ser entrenadas por un clasificador. No obstante, independientemente del rendimiento, el uso de tres símbolos requiere incrementar la longitud de los códigos para obtener resultados robustos. Dado que uno de los objetivos de este proyecto es reducir la longitud de los códigos para conseguir trabajar con grandes cantidades de clases, en este estudio no revisaremos al caso "ternario" de los ECOC.

Vistos ya los estudios hasta el momento, en próximos apartados se discurre sobre el tipo de codificación a utilizar por el sistema, analizando a fondo las mejoras que nos pueden ofrecer sobre lo ya existente.

2.1.2 Decodificación de los ECOC

En esta etapa se obtiene las predicciones hechas por parte de los clasificadores binarios y se procede a su posterior clasificación. En este sentido, basándose en la reglas generales de la etapa de decodificación, estas técnicas o estrategias se pueden dividir en dos grupos, dependientes de como se ataca el problema. El primer grupo trataría de resolver el problema desde la perspectiva del cálculo de la distancia entre la predicción llevada a cabo por los clasificadores y la palabra clave destinada a ese ejemplo. El segundo grupo, atacaría el problema desde la perspectiva de la probabilidad de cada ejemplo de pertenecer a una clase en concreto.

El primer intento de decodificación de un ECOC ha sido la decodificación Hamming. Otra de las más conocidas es la decodificación Euclidiana. En [13] se introduce el estudio de la codificación Hamming invertida, y se introduce la decodificación centrada para diseños de ECOC binarios.

El proceso de decodificación es simple, para cada ejemplo de test en el conjunto de ejemplos, cada clasificador binario entrenado para discernir entre los grupos de clases designados por el ECOC, predice un valor para dicho ejemplo. Una vez obtenidas todas la predicciones del conjunto de

clasificadores, se forma una palabra clave predicha por dicho conjunto de funciones de clasificación y después cada palabra clave obtenida se asigna la clase con una palabra clave mas “parecida”.

Como se comento con anterioridad, uno de los objetivos de este estudio es utilizar un número reducido de clasificadores, con lo cual la longitud de la palabra clave también será reducida. Consecuentemente deberemos tener especial cuidado a la hora de escoger la alternativa de codificación y decodificación.

Analizamos pues tres alternativas en la etapa de decodificación, del grupo que trata el problema desde la perspectiva de la distancia entre la palabra predicha y la palabra clave de cada clase. Así pues, procederemos al análisis de: la decodificación Hamming, la decodificación Hamming Inversa y la decodificación Euclidiana.

- **Decodificación Hamming (HD)**

  La propuesta de decodificación es la decodificación Hamming, en la que se calcula la distancia Hamming entre la palabra predicha por los clasificadores del ECOC y las distintas palabras clave de las clases para encontrar la palabra clave de clase con una distancia mínima a la predicha. Esta estrategia de decodificación está basada en los principios de los principios de los correctores de error bajo la asunción que una tarea de aprendizaje se puede modelar como un problema de comunicación donde la información de cada clase es transmitida por un canal:

$$HD(x, y_i) = \sum_{j=1}^{n} \left(1 - sign\left(x^j \cdot y_i^j\right)\right)/2$$

Donde *x* e *y* son los códigos de test y de una fila de la matriz de codificación, respectivamente.

- **Decodificación Hamming Inversa (IHD)**

  Esta alternativa de decodificación produce una matriz con los valores de proporcionalidad de cada palabra clave de cada clase en la palabra predicha por los clasificadores de los ECOC. En lo que a la práctica se refiere la IHD tiene un comportamiento muy similar al comportamiento de la estrategia HD.

- **Decodificación Euclidiana (ED)**

  En esta estrategia se calcula la distancia Euclidiana de la palabra predicha por los clasificadores del ECOC con las diferentes palabras clave de las clases y se etiqueta a cada ejemplo con la palabra clave de la clase con la distancia Euclidiana mínima:

$$ED(x, y_i) = \sqrt{\sum_{j=1}^{n} \left(x^j - y_i^j\right)^2}$$

2.2 ECOC Sublineal

Así pues, vistos los diferentes tipos de estrategias en las etapas de codificación y decodificación, en este apartado se analiza la alternativa propuesta a la hora de desarrollar el proyecto, entrando en profundidad en las mejoras que obtenemos utilizando este tipo de esquema.

Como habíamos visto con anterioridad, el framework del ECOC se utiliza básicamente con el objetivo de combinar la potencia de los clasificadores binarios.

También se pudo ver como utilizando las estrategias de codificación del estado del arte, el número de clasificadores necesarios para llevar a cabo la tarea es excesivo si pretendemos trabajar con un número elevado de clases, como es el caso de este proyecto.

Así pues, en el desarrollo de este estudio se ha propuesto una nueva técnica de codificación: ECOC Sublineal. Cuando se comenta Sublineal se refiere a que la longitud de palabra a utilizar va a ser la mínima posible con la cual se pueda distinguir unívocamente cada clase: dadas N clases, el número de bits mínimo para distinguirlas coincide exactamente con $\log_2 N$. El algoritmo se describe en la página siguiente.

ALGORITMO GENERACION ECOC SUBLINEAL

entrada: número de clases del problema a tratar (*N*)

generar una permutación aleatoria de números naturales de 1 a *N*

repetir

particionar por el punto central la/s cadena/s de números naturales

hasta que no se puedan particionar en la partición *i* las particiones generadas por la *i* -1

de la estructura de árbol resultante codificar el nivel *i* como la columna *i*, donde el hijo izquierdo tendrá valor 1 y el derecho valor -1

salida : codificación del ECOC Sublineal

Así pues, utilizando este esquema de codificación obtenemos el número mínimo de clasificadores necesarios en nuestro ECOC para poder distinguir las *N* clases que aprenderá a discriminar el sistema.

En un principio se puede pensar que ésta no es una mejora sustancial cuando se trabajo con un número pequeño de clases y estaría en lo correcto dado que la potencia real de esta alternativa de codificación reside cuando *N* incrementa acercándose a los recientemente llamados “Large Escale Problems”, problemas de alta dimensionalidad en los que se trabaja con un número de clases enorme. Es aquí donde la potencia de un ECOC Sublinear se hace notar.

Así pues, proponemos este nuevo sistema como solución al problema de escalabilidad de los ECOC.

En la tabla 2 se puede ver la codificación de un ECOC Sublinear para un problema de 10 clases.

Clase	C1	C2	C3	C4
1	-1	-1	-1	-1
2	-1	-1	-1	1
3	-1	-1	1	-1
4	-1	-1	1	1
5	-1	1	-1	-1
6	-1	1	-1	1
7	-1	1	1	-1
8	-1	1	-1	1
9	1	-1	-1	-1
10	1	-1	1	-1

Tabla 2. ECOC Sublineal para un problema de 10 clases.

En el desarrollo del proyecto, la generación de este tipo de codificación es completamente aleatoria y es independiente del problema a tratar. Como se puede observar, los clasificadores que se utilizan en este tipo de codificación, ya no aprenden a discriminar una clase contra todas las demás sino a distinguir entre grupos balanceados de clases.

De esta manera se puede pensar que la eficiencia del sistema va a radicar en el rendimiento del clasificador solucionando para partición binaria de clases. Esto implica que para poder sacar el máximo rendimiento a un sistema sublineal, el clasificador base aplicado en los ECOC debe ser capaz de aprender las nuevas distribuciones de los datos. Así pues, necesitamos unos clasificadores que tenga un alto rendimiento de predicción. En los siguientes apartados se comentan las diferentes alternativas del estado del arte de los clasificadores robustos que nos van a permitir entrenar los clasificadores de nuestro ECOC sublineal. En particular nos centraremos en las máquinas de

vectores de soporte (SVM, del Ingles Support Vector Machines), las cuales han demostrado tener los mejores resultado de clasificación que se pueden encontrar en la literatura.

2.3 Support Vector Machines (Maquinas de vectores de Soporte)

2.3.1 Utilización y elección de los SVM

- **Clasificación mediante SVM**

Del conjunto de los posibles clasificadores, los clasificadores binarios son los más utilizados. En este tipo de clasificadores el objetivo es separar las dos clases por una función inducida por los ejemplos de entrenamiento. El objetivo de dicha función es que tenga un rendimiento notable con ejemplos sin previo entrenamiento.

- **Clasificación Lineal**

Véase la ilustración 1, he aquí un ejemplo de unos cuantos clasificadores trabajando linealmente para separar dos clases. Obviamente vemos que todos son válidos (todos separan los datos en 2 clases), no obstante, sólo uno de ellos a parte de ser válido es el que minimiza la distancia entre los puntos más cercanos de las dos clases a separar. Este clasificador será conocido en adelante como hyperplano de separación óptima.

Se puede observar que los clasificadores lineales no siempre van a poder discriminar los datos. En múltiples ocasiones los clasificadores lineales no son buenos, particularmente cuando trabajamos con problemas de alta dimensionalidad.

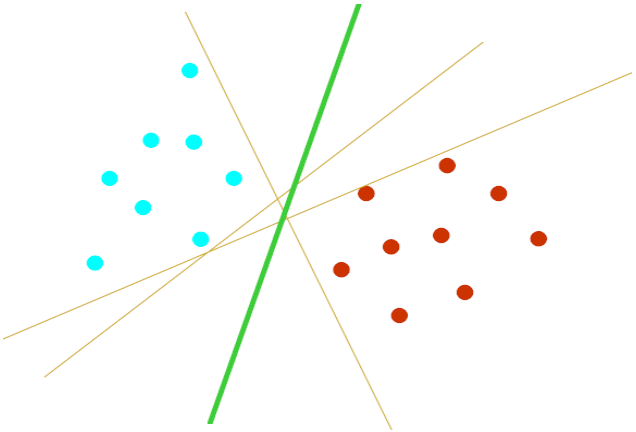


Ilustración 1: Clasificadores Lineales y hyperplano de separación óptima

Así pues, en la mayoría de problemas de clasificación el espacio de entrada será un espacio no clasificable linealmente (Véase ilustración 2).

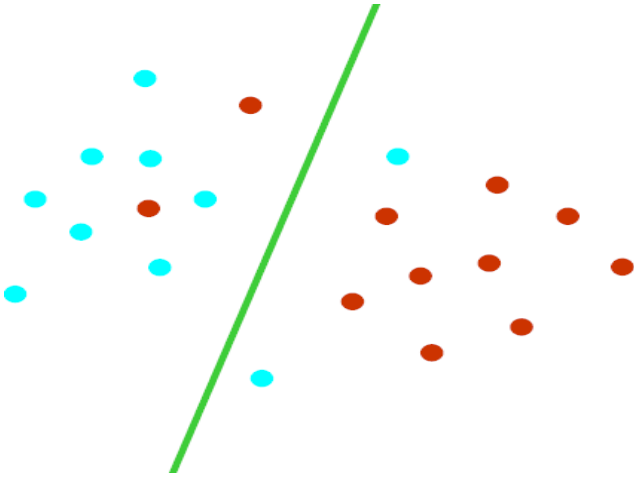


Ilustración 2: Ejemplo de espacio de entrada no clasificable linealmente

Es en este tipo de problemas donde se establece la necesidad de un tipo de clasificador o de función de apoyo que no sea simplemente lineal sino que aporte algún arma para combatir este tipo de problemas. Inicialmente, para la generación del hyperplano de separación óptimo se utilizaba una

función de penalización que castiga los errores en la clasificación. Este proceso es guiado por la optimización de parámetros del clasificador. Por ejemplo, en el caso lineal la optimización se refiere a la minimización de la penalización obtenida por la función de penalización:

$$\Phi(\omega, \xi) = \frac{1}{2}||\omega||^2 + C\sum_i \xi_i$$

Donde *C* es un parámetro a introducir, denominado, consecuentemente parámetro de penalización del error y que puede hacer variar de manera extraordinaria la clasificación si se introduce un valor apropiado para cada problema.

- **Generalización de la alta dimensión del espacio de entrada**

No obstante, aun con esta mejora hay casos en el que el espacio de entrada no es linealmente clasificable incluso utilizando las funciones de penalización. Estos casos suelen ser los de alta dimensionalidad en los cuales el espacio de entrada comúnmente tiene más de 3 dimensiones. Es aquí donde entra la teoría de la generalización de la alta dimensión del espacio de entrada y donde radica la verdadera potencia de los SVM. Se puede observar que cuando una clasificación lineal es inapropiada, los SVM pueden mapear los datos de entrada de dimensión *x*, en un espacio de alta dimensión *z*.

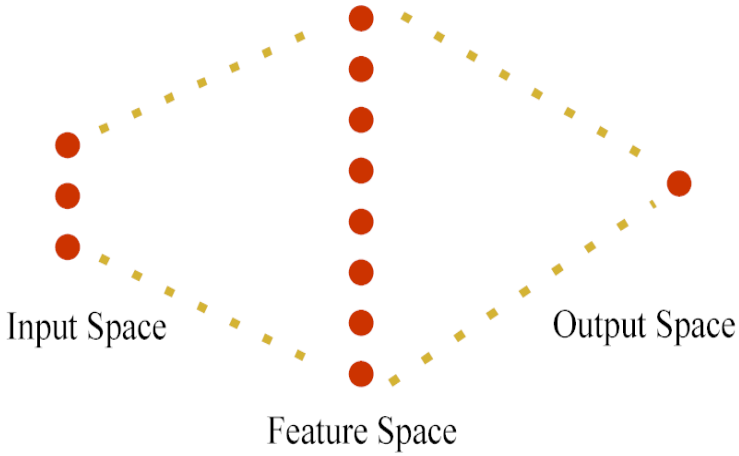


Ilustración 3: Generalización en un espacio de alta dimensión.

Escogiendo un mapeado no lineal a priori el SVM construye un hyperplano de separación óptima en ese espacio de alta dimensión.

Se podría pensar que en el mapeo de los datos de entrada, cualquier función no lineal no sería aceptable, pero sorprendentemente, las funciones utilizadas usualmente son aceptables. Algunos ejemplos son las funciones polinomiales, funciones de radio base o las funciones sigmoideas. En adelante las funciones que realizan el mapeado de los datos de entrada serán denominados “kernels” o núcleos.

- **Funciones de Kernel**

Aunque existen varios tipos de funciones de kernel, vamos a comentar las dos utilizadas en el desarrollo del proyecto, éstos son los kernels polinomiales y los kernels con funciones de radio base (en adelante RBF, del Ingles “Radial Basis Function”).

El kernel polinomial es uno de los más usados para modelar un problema de manera no lineal. Este

se basa en concebir una función polinomial de grado *d* (donde *d* es la dimensión del espacio del kernel) que consiga una clasificación aceptable de los datos de entrada:

$$K(x, x') = (\langle x, x' \rangle + 1)^d$$

Véase en la ilustración 4, como dicho kernel establece una clasificación mucho más aceptable que cualquier clasificador lineal por sí solo. No obstante en la imagen podemos observar que no es la mejor clasificación posible, sino que se puede refinar.

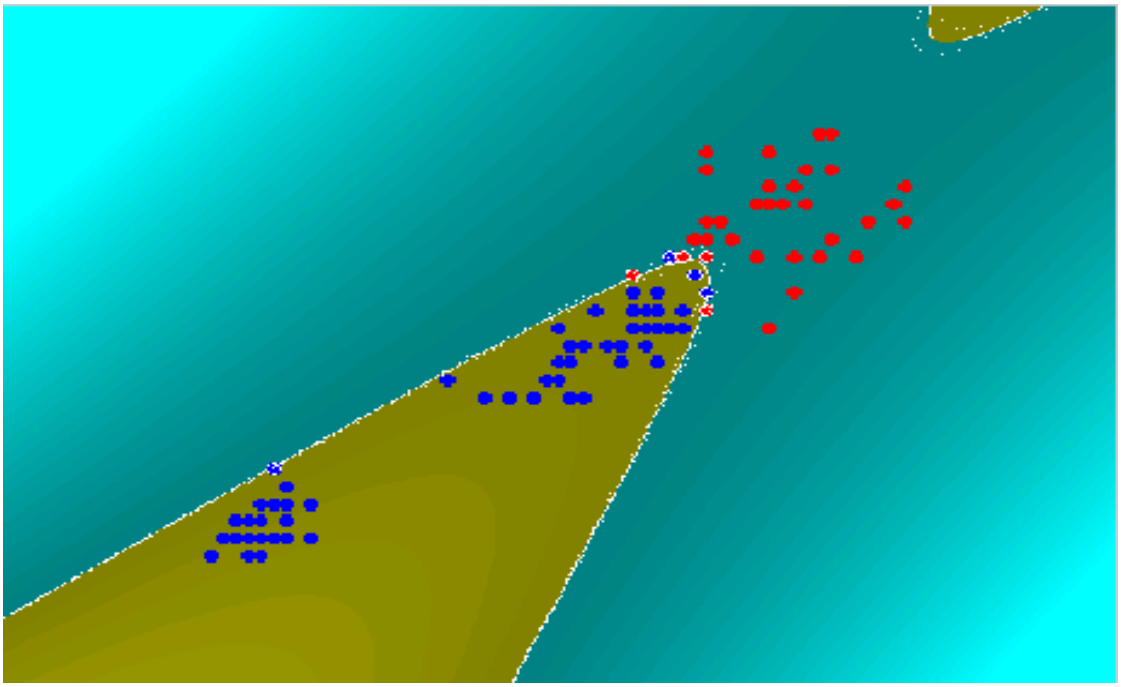


Ilustración 4: Utilización de un kernel polinomial.

Otro tipo de kernel son los de función de radio base (del Ingles “Radial Basis Function”). Este tipo de kernels han recibido especial atención, en concreto los de forma gaussiana:

$$K\left(x, x'\right) = exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

Este tipo de kernel ha sido el elegido para la implementación del proyecto dado su alto rendimiento incluso en casos de muy alta dimensionalidad, como es el nuestro. Este tipo de kernel a parte del previamente comentado parámetro *C*, necesita de otro parámetro para realizar la clasificación, el denominado parámetro Gamma. Utilizando estos dos parámetros y como previamente habíamos comentado, introduciéndolos el valor adecuado, un clasificador RBF puede alcanzar un alto rendimiento en la tarea de clasificación.

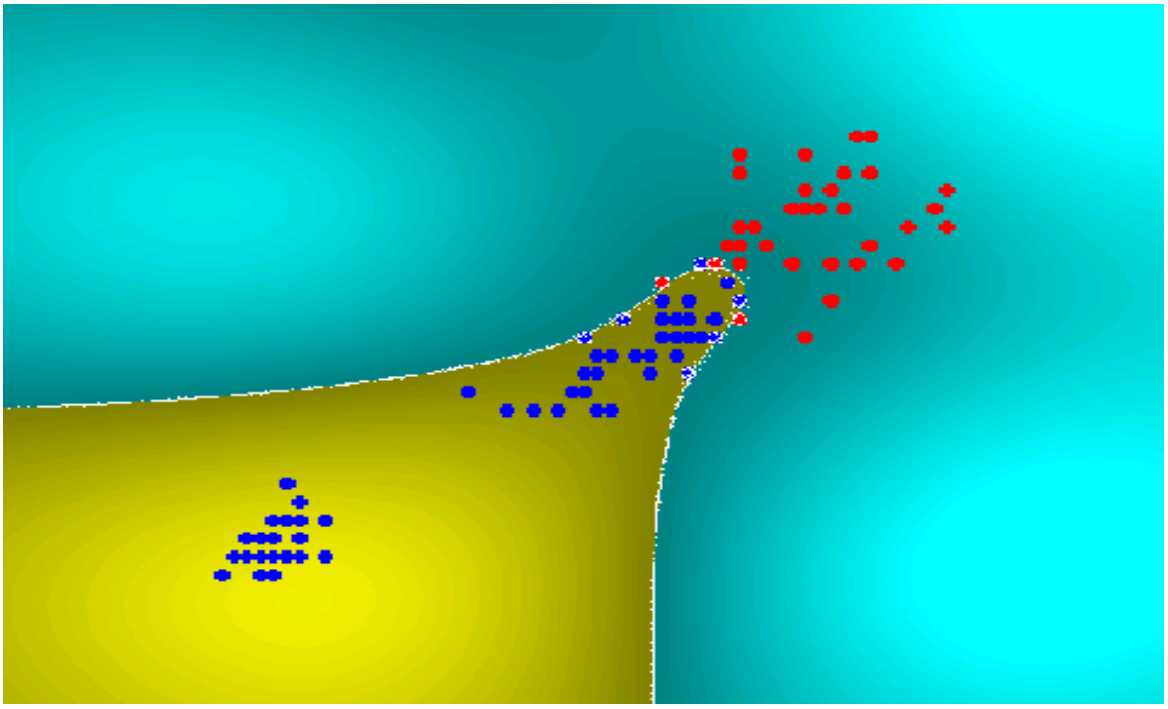


Ilustración 5: Utilización de un kernel RBF.

Se puede observar a simple vista que en esta distribución de datos, un kernel RBF tiene un rendimiento similar o ligeramente superior a un kernel polinomial y por supuesto, el rendimiento del kernel RBF se hace mas fuerte comparándolo con un kernel lineal.

Generalmente esta situación es la usual y a veces se magnifica, es decir, un kernel RBF siempre podrá clasificar lo que clasifica un kernel lineal pero un kernel lineal en multitud de caso no podrá clasificar lo que clasifica un kernel RBF.

Si a este hecho agregamos la sobrecarga de datos que sufrirán nuestros clasificadores, vemos que la opción más correcta es escoger un kernel RBF.

2.3.2 Optimización de los clasificadores SVM

Vistos los análisis anteriores, nos centraremos en la elaboración de un ECOC Sublineal, el cual

solventará el problema de escalabilidad de los ECOC. Escogiendo pues esta alternativa, la tarea de la clasificación base es delegada en los SVM, en concreto, en el SVM con un kernel RBF el cual ha demostrado tener los mejores resultados en problemas de discriminación binaria.

En este apartado analizamos el estado del arte en lo que a selección de los parámetros de manera óptima o sub-óptima a recibir por el SVM se refiere, ya que dichos hyperparámetros pueden afectar sobremanera el comportamiento del hyperplano de separación óptima, haciéndolo desde prácticamente inservible a prácticamente perfecto.

En un principio los estudios se centraron en una búsqueda exhaustiva de los parámetros, acotando el cerco de búsqueda en un determinado conjunto de valores, y utilizando aquellos que demostraban tener un menor error. Estos tipos de metodologías se pueden observar que son ineficientes dado que sin saber la topología del espacio a analizar, no hay ninguna seguridad de que sea el mejor valor de optimización para dicho problema. Además, este método se vuelve completamente ineficiente cuando los hyperparámetros necesarios por el SVM son dos o más.

Desde la perspectiva de optimización, distintas alternativas han sido tratadas. Todas ellas intentando evitar, en la medida de lo posible, una búsqueda exhaustiva de los parámetros necesarios para los SVM.

En un principio, el método usado para encontrar dichos valores fue el gradiente descendiente, que no es más que el algoritmo de búsqueda de un mínimo de una función. Este método lleva consigo el inherente problema del estancamiento en un mínimo relativo, sin poder llegar al mínimo absoluto, véase ilustración 6, que sería en términos de nuestro problema, encontrar los parámetros del SVM que minimiza el error.

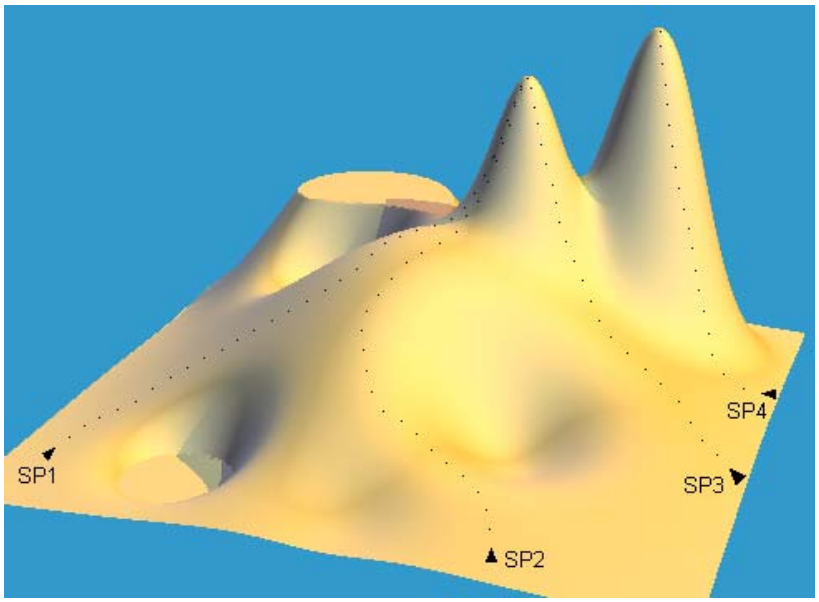


Ilustración 6: Gradiente Descendiente.

En sus estudios, los autores de [13] propusieron un marco de gradiente descendiente para optimizar kernels, minimizando el límite superior del error generalizado, donde el límite es igual al ratio de puntos de el espacio de entrada clausurados por una esfera de radio *r*. Estimar el radio de la esfera de clausura es un problema de minimización cuadrática que sería computacionalmente duro de trabajar para problemas de dimensionalidad alta, como los “Large Escale Problems”.

Partiendo de este estudio muchos han sido los que han propuesto distintas formas de optimizar los SVM, dado que nuestra alternativa en este tema es novedosa. Vamos a comentar de forma breve las propuestas más interesantes en este campo.

Una de las propuestas más interesantes ha sido la utilización del algoritmo de Levenberg-Marquardt para la minimización del radio de la esfera contenedora. La primera aplicación de dicho algoritmos fue la minimización cuadrática de problemas de modelado de curvas. Donde el objetivo era; dado un grupo de pares de puntos, encontrar los parámetros (Beta) de la curva, para que la suma de los cuadrados de las derivaciones fuese mínima.

Así pues, en su estudio de la optimización de los SVM [14] propone la utilización de dicho

algoritmo, obteniendo algunos buenos resultados de optimización. Esto es debido a que dicho algoritmo es una interpolación entre el antes comentado gradiente descendiente y el método de optimización Gauss-Newton. De esta manera, mientras la solución no esté próxima se utiliza el gradiente descendiente y a medida que se aproxima a un mínimo se pone en marcha la minimización Gauss-Newton.

Otra alternativa en el marco de la optimización de los SVM ha sido, como en el apartado anterior, la "unión" de dos métodos de optimización, con la esperanza de poder obtener todos sus puntos a favor. De esta manera, en su estudio de la optimización, [15] propone un modelo de selección basado en la metodología de minimización de la probabilidad del error estimado en el conjunto de validación. Dicho método incluye un proceso automático de minimización que reduce eficientemente el error empírico. De esta manera, se optimiza locamente el conjunto de clasificadores SVM estimando la probabilidad del error en un conjunto de validación.

Visto las alternativas en el estado del arte de la optimización de los kernels del SVM, en el desarrollo del proyecto se ha optado por una nueva alternativa, delegar la optimización del kernel en un algoritmo genético. Dicho algoritmo realizará una computación no exhaustiva de los parámetros a utilizar por el kernel RBF.

En el siguiente apartado se introduce un amplio análisis a dicha solución, como también de su funcionamiento interno y de la adaptación al problema de la optimización del kernel RBF.

2.4 Computación Evolutiva

2.4.1 Introducción

Como previamente se ha comentado, el proceso de optimización de los SVM se ha delegado a un proceso evolutivo. Este tipo de proceso se basa en un algoritmo genético, que no es más que un tipo de algoritmo dentro del conjunto de la computación evolutiva. Es este apartado se analizan, dichos algoritmos profundamente, así como su aplicación en el sistema.

La computación evolutiva es una amplio conjunto de métodos inspirados en la teoría de la evolución de Darwin [1]. La evolución natural se puede observar como un proceso de optimización basado en la población. Dicho proceso, puede ser observado como un proceso iterativo de adaptación al medio por parte de la población existente. El proceso iterativo consta de diferentes etapas que ocurren una después de otra, conformando lo que se conoce como una Máquina de Darwin Genética (ver ilustración 7).

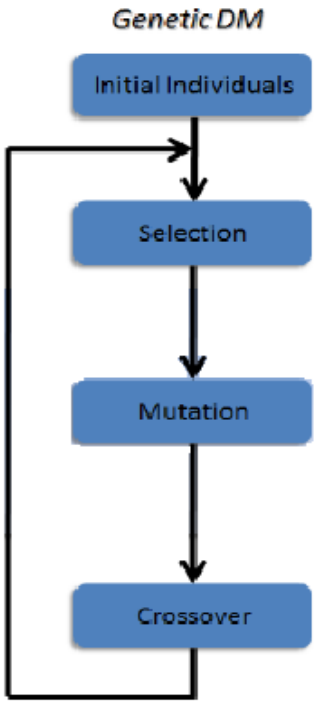


Ilustración 7: Maquina de estados de un proceso evolutivo

En la naturaleza todo ser viviente está “codificado” en términos de su material genético. Ya sea en su ADN o más ampliamente en sus genes, o incluso más ampliamente en su genoma. Dicho material genético es utilizado como los ingrediente de una receta a la hora de crear un nuevo ser vivo. Así pues, la adaptabilidad de un cierto sujeto al medio vendrá dada por su material genético, y éste a su vez vendrá dado por el material genético de sus predecesores. La evolución en este término tiene que ver con el proceso que permite la utilización del material genético y su mejora, con el objetivo de una mejor adaptación al medio.

Las bases de los algoritmos evolutivos recaen sobre la “Selección Natural”, donde los mejores individuos pueden sobrevivir y reproducirse con el objetivo de perpetuar la especie. La información genética de dichos individuos es transmitida a lo largo de las generaciones, creando cada vez

individuos más adaptados al medio, y mejorando la especie. En general, en la evolución natural los individuos son clasificados por su adaptación al medio, es decir, los mejores individuos son los más adaptados al medio.

A parte del anterior comentado paso de información genética a lo largo de las generaciones, tenemos otras dos fuentes de evolución, el apareamiento y la mutación.

- **Apareamiento o Reproducción**

  En el proceso reproductivo de todo ser vivo, dos individuos de la especie intercambian material genético, así pues, el individuo resultante adquiere en su genoma partes de sus dos predecesores. Cuando esa mezcla deriva en una mejor adaptación al medio, ese nuevo individuo sobrevivirá y su información genética se mantendrá a lo largo de la generación. Por el contrario si dicha mezcla deriva en un individuo peor adaptado, no sobrevivirá y esa información se perderá.

- **Mutación**

  En contraste con la reproducción, donde la mezcla de material genético existente es combinaba, un nuevo material genético puede ser generado. La mutación básicamente consiste en pequeños cambios aleatorios en la información genética de un ser vivo. Análogamente a la reproducción, cuando dicha mutación conduce a un individuo mejor adaptado al medio, dicha información perdura a lo largo de las generaciones, en caso contrario, se pierde dado que el individuo no sobrevive.

  Darwin explica la evolución de las especies como inferencia de dichos métodos, y posteriores estudios en el campo de la biología demuestran que esto se puede generalizar

para todos los seres vivos del planeta. Así pues, dichas operaciones son utilizadas para simular un proceso evolutivo desde una perspectiva computacional.

2.4.2 Algoritmos genéticos

Los algoritmos Genéticos son probablemente la aplicación más común de la computación evolutiva y de la simulación de la evolución natural. La idea debajo de este tipo de algoritmos es la reproducción de la evolución natural mediante programas de ordenador, utilizando una representación basada en la genética de la vista del problema e implementando desde un punto de vista funcional los diferentes métodos de evolución implicados en dicho proceso. Usualmente dichos algoritmos se utilizan en procesos de optimización, dado que los GA (del inglés Genetic Algorithms) pueden optimizar la mayoría de parámetros sea cual sea la topología del espacio de soluciones. Normalmente al utilizar este tipo de algoritmos sólo encontramos dos componentes que sean dependientes del problema a tratar. Éstos son, la codificación de los individuos y la función de evaluación. El resto de operadores son estándar con el tipo de codificación que se elija. En los próximos apartados definimos brevemente las distintas alternativas y nos centramos en las utilizadas en el desarrollo del proyecto.

- **Codificación**

  Este problema viene dado por la elección del tipo de representación de las soluciones o puntos en el espacio de soluciones por medio de genotipos o alternativamente cromosomas.

  Aunque la codificación depende básicamente del problema a resolver, existen un conjunto de aproximaciones estándar para la resolución de este problema:

  - **Codificación Binaria:** es la más utilizada dado que es la primera que se utilizó en este tipo de algoritmos. Consiste en la codificación de una

cadena de bits donde cada bit indica un cierto aspecto de la solución en términos del problema. Este tipo de codificación usualmente no es natural para muchos problemas y a veces se han de corregir las operaciones de mutación y cruce. La representación binaria más útil para este tipo de codificación es la conocida como codificación de Gray.

- **Codificación Permutada:** es un tipo de codificación donde cada cromosoma consiste en una cadena de números que representan un número en una secuencia. Este tipo de codificación es muy útil en problemas de ordenación.
- **Codificación por Valor**: este tipo de codificación puede ser utilizada en problemas, donde valores complicados han de ser tratados, como por ejemplo números reales. Ya que la utilización de un codificación binaria podría resultar muy tediosa. Los valores pueden ser cualquier tópico relacionado con el problema. Este tipo de codificación puede ser muy útil para algunos tipos de problemas. Por otra parte, normalmente es necesario desarrollar las operaciones de mutación y reproducción específicas para el problema.
- **Codificación en árbol:** la codificación en árbol es utilizada para codificación de expresiones o programas, para computación evolutiva. En esta codificación cada cromosoma es un árbol con ciertos objetos. Este tipo de programación es buena para evolucionar programas.

- **Reproducción (Crossover)**

La reproducción consiste en la creación de un nuevo individuo mediante la recombinación de dos individuos. Aunque la reproducción es uno de los procesos estándar en los algoritmos genéticos, muchas de las estrategias de reproducción esta creadas para codificaciones

binarias. La idea que existe detrás de este operador es que un nuevo individuo mejor será creado tomando las mejores partes del genoma de sus dos progenitores. El estado del arte de esta estrategia se describe en los siguientes apartados.

- **N-puntos:** es un operador de reproducción que aleatoriamente escoge *n* puntos en un cromosoma e intercambia el material genético entre esos puntos para producir el nuevo individuo.
- **Uniforme:** este operador decide con un cierta probabilidad dada (ratio de combinación) que padre contribuirá con que gen en el nuevo individuo. Esto permite a los progenitores recombinarse a nivel de gen, mientras que en *n*-puntos la recombinación se produce a nivel de segmento.
- **Heurística:** en este operador se usa el valor de adaptación de cada operador para determinar la dirección de la recombinación, que se producirá en la dirección del progenitor con un nivel de adaptación más alto. Dicho nivel de adaptación se multiplica por un ratio aleatorio.

- **Mutación**

  La mutación consiste en la alteración ocasional y aleatoria de pequeñas partes de un cromosoma. Dicha operación tiene como objetivo la no pérdida de algún bit. Por ejemplo, usando la codificación binaria es posible que después de algunas generaciones la selección lleve a todos los bits en alguna posición a ser 0 o 1. Si esto pasa con el algoritmo convergiendo hacia una solución satisfactoria, entonces el algoritmo ha convergido prematuramente. Esto puede ser un problema especialmente con poblaciones muy pequeñas. Sin la mutación no tendríamos la posibilidad de reintroducir el bit perdido. Los operadores de mutación utilizados usualmente son:

- **Flip bit (giro de bit):** simplemente invierte el valor de un gen previamente escogido. Este operador solo puede ser usado para codificaciones binarias.
- **Boundary (limitador)**: se intercambia el bit con el del final o el del principio de la cadena (escogido aleatoriamente). Este tipo de operador solo se puede utilizar para genes de tipo “Integer” o “Float”.
- **Uniform (uniforme)**: se reemplaza el gen con un gen obtenido con una distribución aleatoria uniforme entre los límites indicados para ese gen. Este tipo de operador solo se puede utilizar para genes de tipo “Integer” o “Float”.
- **Gaussian (Gausiana):** este operador suma al gen elegido una unidad Gausiana distribuida aleatoriamente. Este tipo de operador solo se puede utilizar para genes de tipo “Integer” o “Float”.

A parte de la estrategia de mutación elegida es necesario garantizar un ratio de mutación pequeño para evitar que el proceso estocástico pase a ser aleatorio.

- **Selección**

Esta etapa se refiere a la simulación de la *Selección Natural* de forma computacional. En ésta, los individuos mejor adaptados sobreviven mientras que los menos adaptados desaparecen. El valor de adaptación es establecido por la función de evaluación, una función que asigna un valor a cada individuo basándose en su adaptación al problema. Son muchas las veces que se utilizan los términos función de evaluación y función de adaptabilidad como sinónimos sin embargo divergen en que una función de evaluación asigna un valor al individuo en términos de adaptación al problema, mientras que la función de adaptabilidad asigna el valor de adaptabilidad a un medida de oportunidades de reproducción. En los siguientes apartados son descritas las funciones de evaluación más utilizadas.

- **Roulette Wheel (Ruleta):** es un proceso de elección de individuos proporcional a su valor de adaptación al medio. Es decir, a mayor valor de adaptación al medio, mayor probabilidad de ser elegido. Una de las desventajas de este proceso es que no se garantiza que el mejor indivíduos de esa generación pase a la siguiente.
- **Rank (posición):** en este proceso de selección se ordenan los individuos por su valor de adaptación al medio y son escogidos para pasar a la siguiente generación los *m* primeros individuos, donde $m<I$ (*I* siendo el número de individuos de una generación).
- **Tournament (torneo):** en esta estrategia se escogen *n* individuos y se guarda el mejor adaptado, el modo de torneo más usado es el binario, donde sólo dos individuos son elegidos.

Ya que la mayoría de procesos de selección suele estar basados en probabilidades, a menudo se introduce un valor de elitismo que garantiza que los *n* individuos mejor adaptados pasaran a la siguiente generación.

- **Evolución**

Una vez descritas las partes que trabajan conjuntamente en un GA, se describe la evolución simulada como el proceso iterativo de la máquina de estados, donde los individuos son llevados generación a generación a una adaptación al medio óptima (en el mejor de los casos). Los criterios de parada de la evolución han de ser definidos con conocimiento del problema y a menudo suelen estar relacionados con el tiempo de cómputo o con el número de generaciones que pueden pasar sin mejora de la adaptación media de la población. En la

siguiente figura se describe el algoritmo iterativo de un proceso evolutivo.

ALGORITMO EVOLUTIVO

entrada: Función de evaluación y parámetros necesarios

Generar una población inicial aleatoria P0

repetir

usar la función de evaluación para calcular la adaptabilidad de la población *Pt*

seleccionar a los padres de la población *Pt*

utilizar reproducción para generar la población *Pt+1*

utilizar mutación para generar la población *Pt+1*

hasta el mejor individuo es suficientemente bueno o se llega a un criterio de parada

salida : el mejor individuo de la última población.

En el apartado siguiente se analiza la solución propuesta para resolver los problemas de alta dimensionalidad propuestos. Analizando la mejora que proporciona cada parte con las alternativas del estado del arte en los diferentes tópicos relacionados con el desarrollo del proyecto.

Para terminar este apartado, comentaremos que para los procesos evolutivos utilizados se ha optado por un proceso de selección por ranking.

Para el operador de reproducción se optó por una estrategia uniforme para la optimización de los kernels RBF y por un proceso de reproducción heurístico para la evolución de las matrices ECOC dado que representa significantes mejoras a niveles de convergencia para esta población en concreto. Comentar la utilización de una función híbrida: cuando el GA acaba su tarea, esta función híbrida hace una pequeña búsqueda exhaustiva alrededor del punto mostrado por el GA para confirmar que sea el valor mínimo.

2.5 ECOC SVM Sublineal Evolutivo

En este capítulo se hace una descripción y un análisis a fondo de la aproximación utilizada por nuestro estudio para resolver los problemas de clasificación, y como dicha solución puede ser aplicable a los problemas de alta dimensionalidad como los "Large Escale Problems". No obstante, procederemos a un análisis del estado del arte de las propuestas de resolución en este campo.

Para empezar comentaremos que la resolución de los problemas de alta dimensionalidad se ha venido estudiando durante los últimos años sin llegar a una solución eficiente y computacionalmente viable. Nuestra aproximación pretende acabar con este problema dando un marco fiable con el que tratar dicho tipo de problemas.

Inicialmente, uno de los problemas a tratar es que el tiempo de cómputo de entrenamiento del SVM crece rápidamente con el número de ejemplos. La razón de esto es que el problema núcleo de los SVM es un problema de programación cuadrática (QP). Los resolutores de problemas QP tienden normalmente a tener una escalabilidad de $O(n^3)$, así pues, la manera de entrenar un SVM de forma eficiente es aún un problema a resolver por los investigadores. En un principio se optó por una computación paralela de los SVM donde el conjunto de aprendizaje se particionará en diferentes subconjuntos y cada uno de estos se entrenara paralelamente. En la partición de el conjunto total de aprendizaje se entrena una red neuronal para participar dicho conjunto de forma óptima. Esta estrategia de resolución se dio por obsoleta dada la dependencia de los datos en el entrenamiento de los SVM y su incompatibilidad con los sistemas distribuidos.

Estudios posteriores, proponen marcos de computación paralelas para resolver de forma viable los problemas QP en el entrenamiento de los SVM con un conjunto de entrenamiento de alta dimensionalidad. Por ejemplo, el entrenamiento de SVM en cascada donde se entrenan clasificadores SVM. En este sistema se entrenan inicialmente un número reducido de clasificadores,

estos resultados son combinados después en un proceso jerárquico, como el descrito en la ilustración 8, donde el resultado del último clasificador es utilizado como feedback en un proceso con retroalimentación.

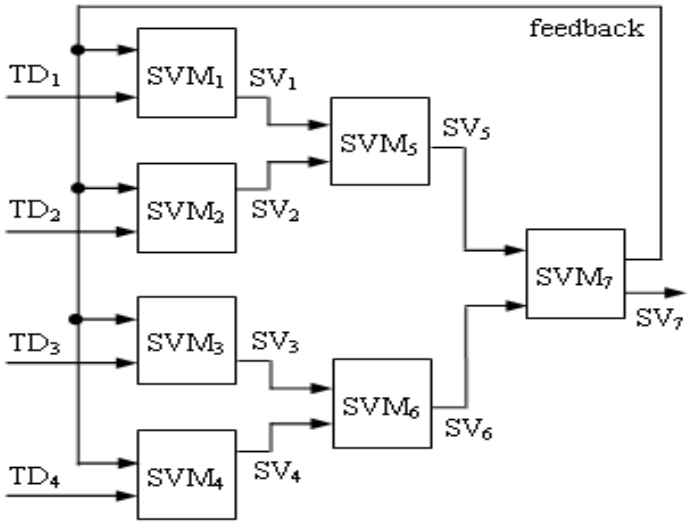


Ilustración 8: Entrenamiento en cascada de 7 clasificadores SVM

Esta estrategia sí que es viable para ser utilizada en sistemas distribuidos, y la recolección del feedback hace que el método de cascada mejorada en [16] muestre en sus experimentos unos resultados eficientes, pero aún tiene problemas a resolver, como la óptima partición del conjunto de entrada.

Por otra parte, otra estrategia para abordar el problema del entrenamiento de los SVM han sido los denominados IRSVM (Incremental Reduced Support Vector Machines), que comienzan su entrenamiento con un conjunto de entrenamiento muy reducido y se incrementa ligeramente, basándose en una serie de pequeños problemas de programación cuadrática. El tamaño del primer conjunto reducido de aprendizaje es determinado dinámicamente por el algoritmo. Los resultados demostraron que el uso de esta estrategia mejora el rendimiento en generación de clasificadores y

kernels no lineales. Así pues, vistos los estudios en este campo, se propone la alternativa de un ECOC-SVM Sublineal Evolutivo, que pretende mejorar los estudios existentes en los siguientes puntos:

- **ECOC**

  Dicho marco de trabajo presenta mejoras sustanciales, en lo que a problemas de multi-clasificación se refiere, a otros marcos como por ejemplo los BCH o el MLP. Así pues, será éste el marco para tratar los problemas de multi-clasificación.

- **SVM**

  Los clasificadores SVM y en concreto los SVM-RBF son los clasificadores que nos han ofrecido un rendimiento y una fiabilidad más grande en problemas de clasificación binaria.

- **Sublineal**

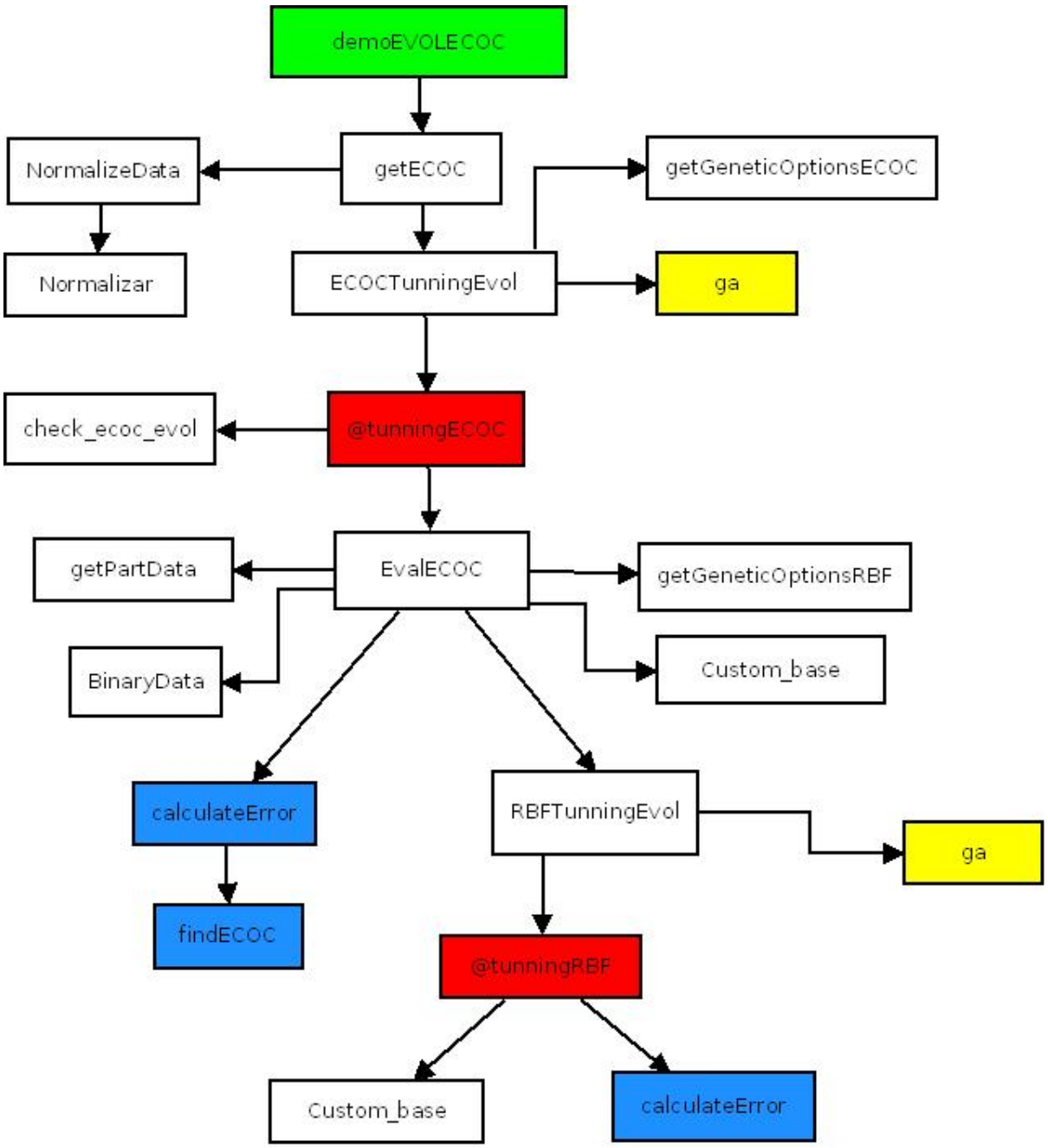
  Las estrategias previas utilizaban codificaciones OneVsAll o Sparse Random (entre otros), con lo que el número de clasificadores necesarios para que el problema cumpliese las condiciones ECOC eran enormes para problemas de alta dimensionalidad. Con esta nueva codificación sólo son necesarios $\log_2 N$ clasificadores, donde N es el número de clases.

- **Evolutivo**

  Las estrategias de optimización de los SVM distan mucho de ser procesos que optimicen los hyperparámetros en un tiempo computacionalmente viable y que no se estanquen en mínimos locales. La computación evolutiva nos ofrece una potente arma para delegar la optimización asegurándonos que no se tratara el problema desde una perspectiva exhaustiva.

2.6 Diagrama colaborativo, cronograma y costes

Finalmente, antes de mostrar los resultados y como parte final de diseño, a continuación se muestra el diagrama colaborativo de clases del sistema.



Todas las funciones han sido implementadas en matlab, tanto las correspondientes a los diseños de ECOC, los clasificadores SVM, como la programación evolutiva. El diagrama muestra la interacción entre las funciones que implementan el sistema. Como se pueden apreciar las diferentes notaciones se indican a partir de diferentes colores. Las funciones en color verde corresponden a métodos para testear el sistema. Es decir, funciones demo. Las funciones rojas son “function handlers” que necesitan las funciones amarillas, ga, (genetic algorithm), implementadas por el propio Matlab. Las funciones azules son funciones de ayuda de su función padre editadas en el mismo script. Cada función en blanco tiene una funcionalidad concreta y está aislada en un script.

En el siguiente diagrama de barras se muestra en modo de resumen el tiempo que se ha invertido en cada una de las fases que engloban el proyecto.

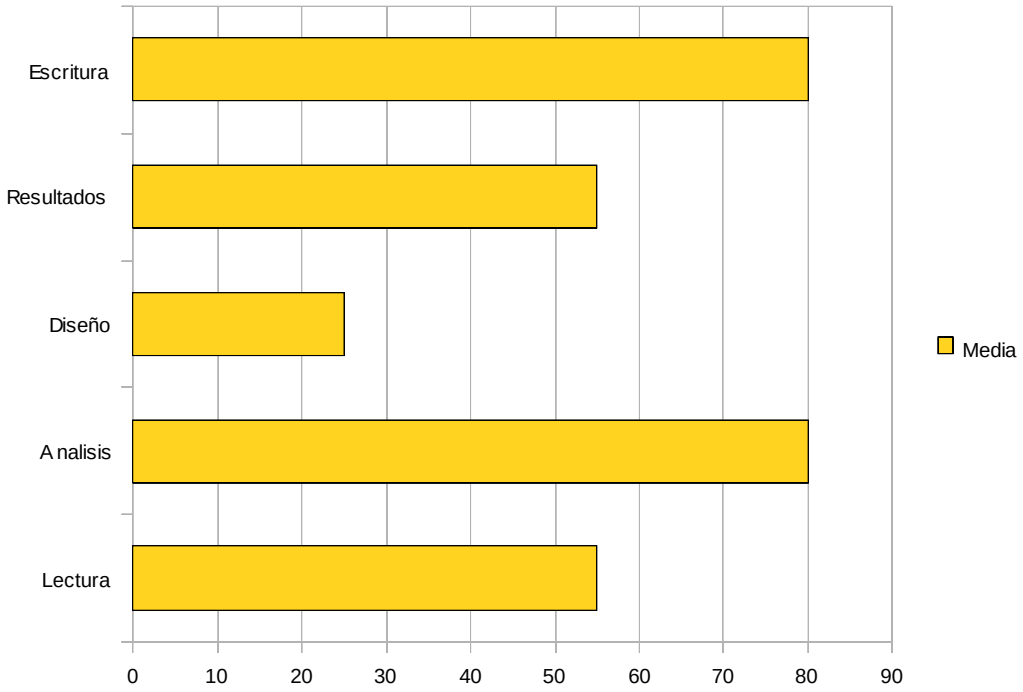


Diagrama de barras del tiempo en horas utilizado en el desarrollo del proyecto

En el cronograma se puede observar como las tareas que más tiempo han comportado han sido las de análisis y escritura. Esto ha sido provocado por la gran carga lectiva del proyecto, el cual tocaba tópicos que no se habían desarrollado durante el transcurso de la carrera.

El volumen de trabajo para el diseño ha sido reducido en comparación al resto dado que se ha utilizado un lenguaje de scripting como es Matlab, que relega todo el diseño al simple uso de scripts.

Procederemos seguidamente a analizar los gastos que hubiesen supuesto la realización del proyecto. Analizaremos el coste de licencias, el precio del analista programador por hora y los ítems que se han llevado a cabo.

Items	Hrs analista	Licencia (al año)	Total
Codificación ECOC Sublineal	10	1000	
Integración OSU-SVM	15	1000	
Evolución kernel RBF	15	1000	
Evolución ECOC	15	1000	
Batería TEST	20	1000	75*50+1000
Total			4750 €

Observamos que a 50 € por hora que podría cobrar un analista. El precio final es de 4750 €.

3.Resultados

3.1 Introducción

En este apartado se presentan los experimentos realizados con el objetivo de demostrar el funcionamiento del sistema. Para ello se desarrollado una batería de pruebas sobre algunos datasets de la UCI. En esta batería de pruebas se confirma tanto el objetivo de conseguir un porcentaje de error en la clasificación que mejore los resultados del estado del arte como el funcionamiento de los procesos evolutivos para la optimización de los hyperparámetros de los SVM tanto de kernel lineal como de kernel RBF (así como la evolución del ECOC).

Así pues, los datasets elegidos fueron extraídos del UCI Machine Learning Repository, dichos datasets son "estándar" a la hora de demostrar el funcionamiento de nuevas alternativas al problema de multi-clasificación. Dichos datasets son descritos en la siguiente tabla.

	#ejemplos	#atributos	#clases	#tipo
Iris	150	4	3	real/multivariable
EColi	336	8	8	real/multivariable
Vowel	528	10	11	real/multivariable
Yeast	1484	8	10	real/multivariable

Tabla 3: Datasets utilizados en la batería de pruebas

Así pues, el método que vamos a utilizar para obtener soluciones al problema de clasificación va a ser el propuesto en este estudio como ECOC-SVM Sublineal Evolutivo. Los resultados obtenidos son analizados en profundidad y comparados con los que se han obtenido en algunos estudios como [18], [19] y [20]. Demostrando la mejora que reporta trabajar con la estrategia propuesta en este estudio.

La métrica utilizada para llevar a cabo los experimentos será la denominada "stratified ten-fold

cross validation". Éste es un proceso iterativo (*i*=1:10), en el que para cada ECOC generado, se particionarán los ejemplos del dataset en 10 subconjuntos. En la primera iteración obtendremos el error del ECOC utilizando el *i*-ésimo subconjunto para testear y los demás para entrenar. De esta manera, después de diez iteraciones obtendremos un valor promedio de eficiencia de ECOC. Reiteramos que todos los resultados mostrados en estos estudios son relativos a este tipo de métrica.
De esta manera, este capítulo tendrá cuatro apartados, uno para cada dataset a clasificar, evaluando el uso de un kernel RBF contra un kernel lineal, el uso de un algoritmo evolutivo contra una búsqueda exhaustiva, el correcto funcionamiento de la evolución posterior de los ECOC y, finalmente, comparar dichos resultados con los obtenidos en el estado del arte para estos problemas.

Procederemos pues en este apartado a demostrar la viabilidad de la metodología para los problemas de multi-clasificación propuestos en este estudio. El orden de los datasets será de mayor complejidad a menor complejidad en lo que a tarea de clasificacion se refiere: Iris, Ecoli, Vowel y Yeast.
De esta manera, algunos conceptos han de ser asumidos antes de poder interpretar las ilustraciones resultantes de la batería de pruebas. En primer lugar se debe comprender que en las ilustraciones referentes a un kernel lineal, el cambio de valor del hyperparámetro *C* se ve reflejado en el eje *y*, es decir, los cambio se visualizaran en forma de "filas". No obstante, para interpretar las ilustraciones referentes a kernels RBF, hemos de saber que además de que el eje *y* representa el cambio de valor en el hyperparámetro *C*, el eje *x* representa el cambio de valor en el hyperparámetro Gamma. Oscilando *C* entre 1 y 50 y Gamma entre 0 y 5. Para interpretar los diagramas de evolutivos se ha de saber que cada valor de error está asociado a un par de valores (*C*, Gamma), es decir, que los individuos de nuestra población serán pares de valores (*C*, Gamma).

3.2 Iris

Este dataset tiene tres clases, cada clase asociada a un tipo de planta de iris, Setosa, Virgínica y Versicolor. Se podría decir que es el dataset más conocido y utilizado para estudios de clasificación. Normalmente este dataset es fácil de clasificar, y en muchos estudios se ha conseguido sin ningún tipo de error 100%. Veamos pues los resultados para este conjunto.

Éste es un recorrido exhaustivo de los hyperparámetros necesarios para cada kernel.

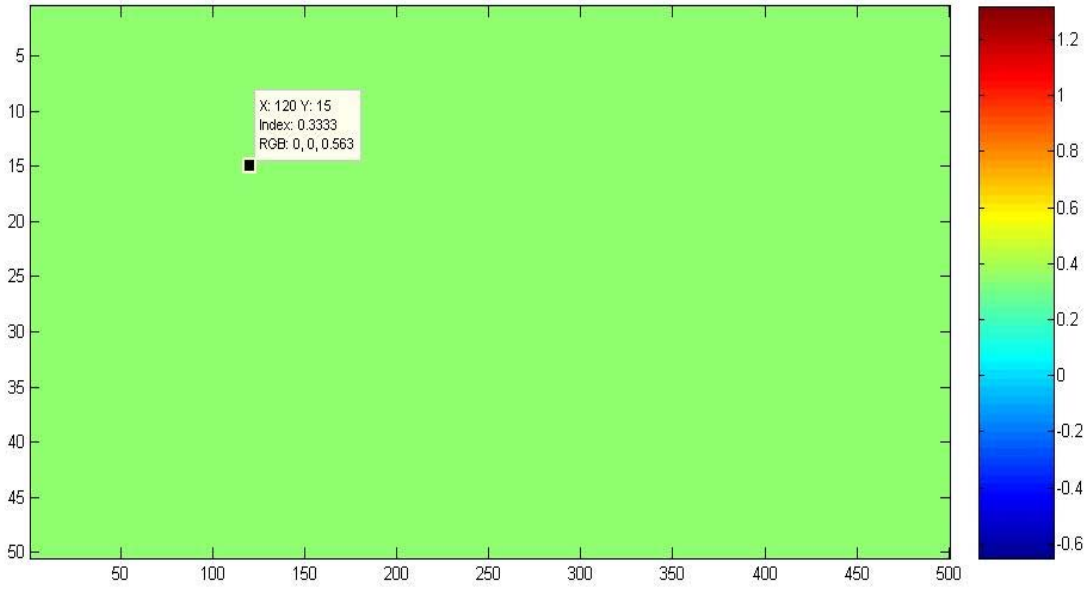


Ilustración 9: Error en función del parámetro C de un kernel lineal para la clasificación de Iris.

En este caso se puede observar una distribución uniforme del error. Este resultado, a priori sorprendente se produce por qué tal y como se argumenta en la documentación de dicho dataset, las clases 1 y 2 son linealmente separables entre sí, no obstante, cualquier separación lineal que se haga de dichas clases con la clase 3 será infructuosa. Como se puede observar, dicha ineficacia se denota

en que el error medio es de 33%, es decir, la perdida de una clase. Por otra parte, mediante el uso de un kernel RBF obtenemos dicha representación:

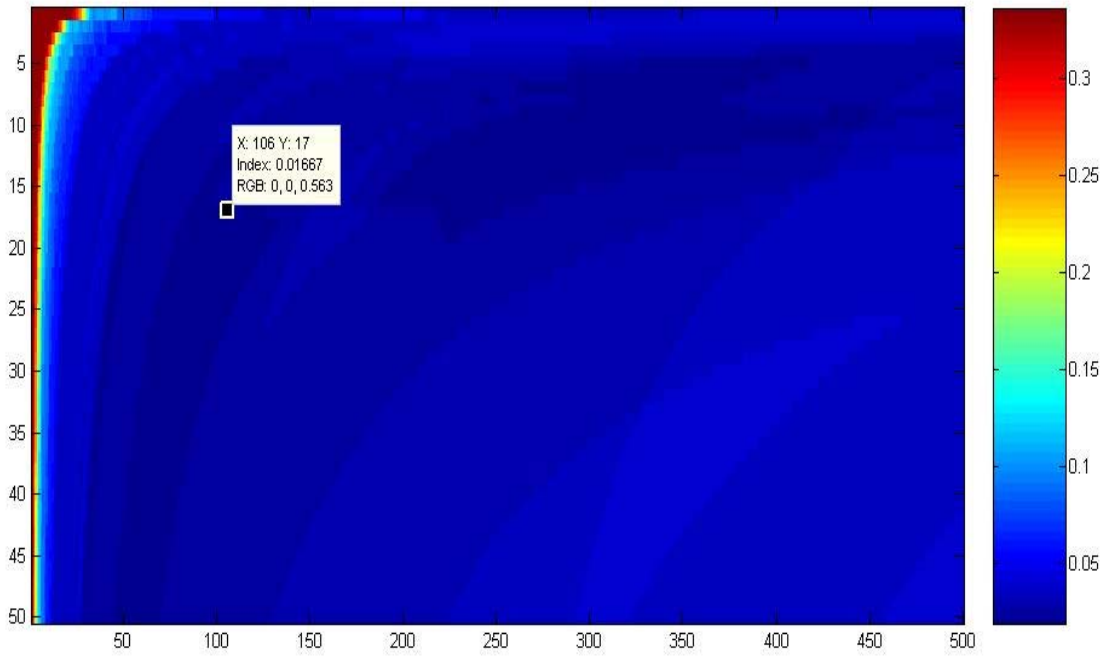


Ilustración 10: Error en función de los parámetros C y Gamma de un kernel RBF para la clasificacion de Iris.

Se pueden observar multitud de cambios. El primero y principal es que el error mínimo obtenido ya no es del 33% sino que esa cota se reduce hasta 1.7%. Como podemos observar, un kernel RBF nos reporta un rendimiento y una eficiencia muy superior a la de un kernel lineal. El siguiente punto a denotar es la no uniformidad del esquema, lo cual demuestra las anteriores afirmaciones de que la elección correcta de los hyperparámetros del SVM puede llevarnos a situaciones muy satisfactorias. Como último punto hacer hincapié en un aspecto que a simple vista no es visible y es la lenta transición de errores más grandes a errores más pequeños. Es decir, no hay un salto brusco de un error del 33% a un error de 1.7%, sino que es un transición suave con el incremento de valor de los hyperparámetros. Este factor es el que va a hacer posible que la búsqueda evolutiva de los

parámetros sub-óptimos del SVM sea un proceso estocástico y no aleatorio.

En la ilustración posterior podemos observar como el evolutivo ha encontrado unos hyperparámetros de clasificación óptimos, ya que producen 0% de error al clasificar la distribución de datos designada por la columna de ECOC [-1 -1 1]. Resulta algo sorprendente, ya que el análisis exhaustivo muestra un error mínimo del 1.4%, esto se produce porque en las ilustraciones anteriores se sigue la métrica stratified ten-fold cross-validation, mientras que en estas pequeñas muestras evolutivas de columna de ECOC se ha decidido no implementarla.

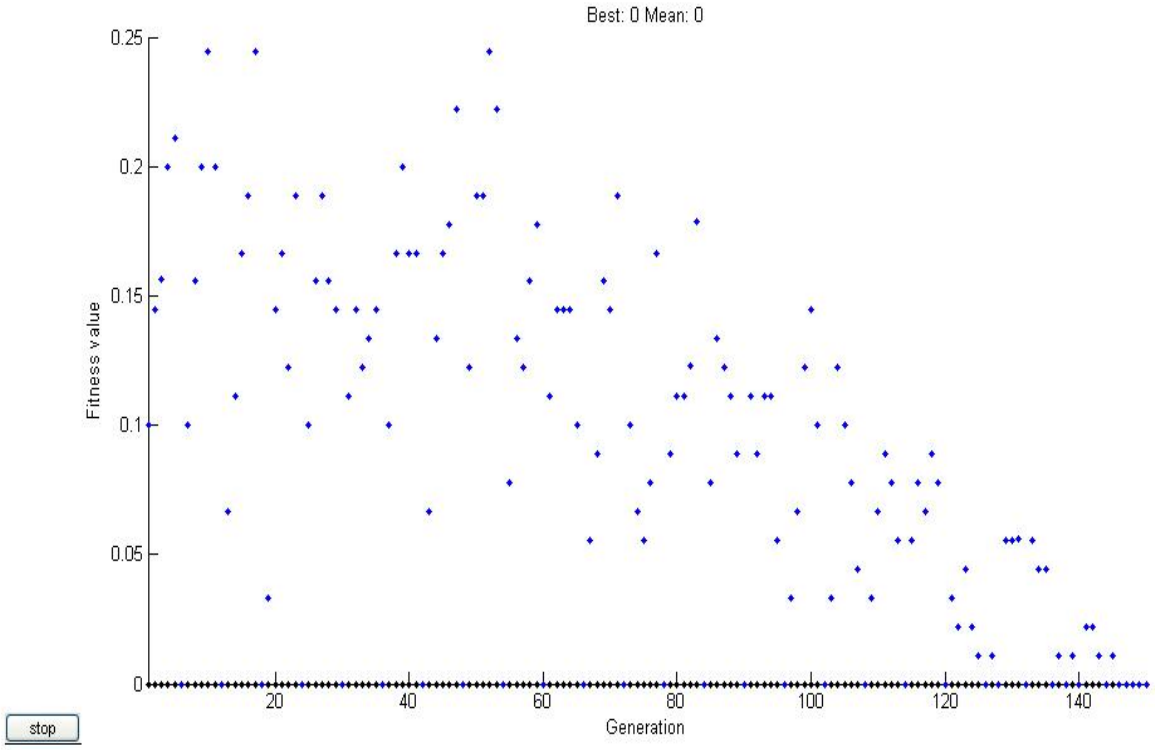


Ilustración 11: Proceso evolutivo de optimización de un kernel RBF en Iris.

El análisis realmente interesante que se debería realizar sería que tal y como muestra la figura 11, el proceso evolutivo tenga una tendencia de convergencia al resultado óptimo.

3.3 EColi (Escheria Coli)

Este dataset contiene 8 clases, asociadas cada una de ellas a la posición de una cierta proteína en un sistema multi celular. Este dataset incrementa sustancialmente la complejidad de clasificación binaria, ya que al ser un clasificación de un conjunto de clases balanceado, contra otro, la carga por clase de datos será mayor (y la distribución de las clases que componen esta agrupación es desconocida, y en la mayoría de casos será muy distinta). Así pues, repitiendo los experimentos para dicho dataset obtenemos los siguientes resultados.

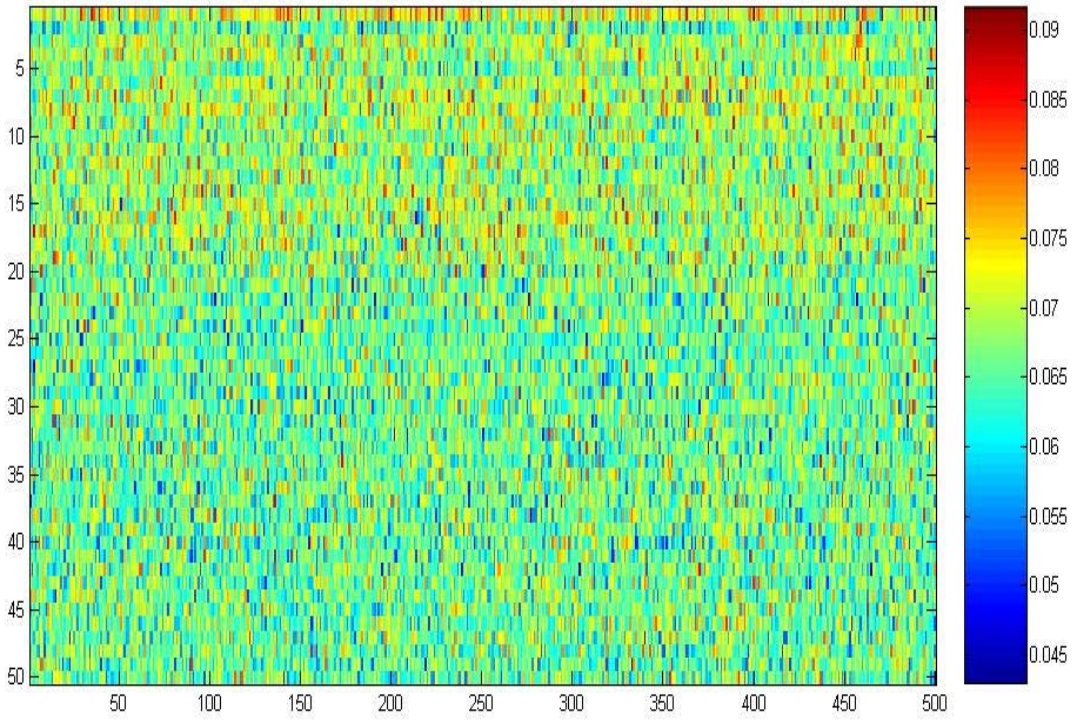


Ilustración 12: Error en función del parámetro C de un kernel lineal para la clasificacion de EColi.

En un principio podríamos pensar que esta distribución sigue un patrón casi aleatorio. Sin embargo, cabe recordar que en un clasificador lineal sobrecargado de ejemplos, como es nuestro caso, la tendencia es ésa. Por otra parte, mirando con atención se puede observar que horizontalmente (el cambio de valor de *C* está representado en el eje Y) se observa para valores de *C* de 0 a 20 unas distribuciones de color amarillo (entre un 7% y un 7,5% de error) y para valores de 20 a 50 tenemos unas distribuciones de color azul claro (error entre el 6% y el 6,5%). Dicha mejora es la que puede suponer la elección sub-optima de parámetros en un kernel lineal. No obstante, si atacamos dicho problema desde la perspectiva de un kernel RBF obtenemos los resultados de la ilustración 13.

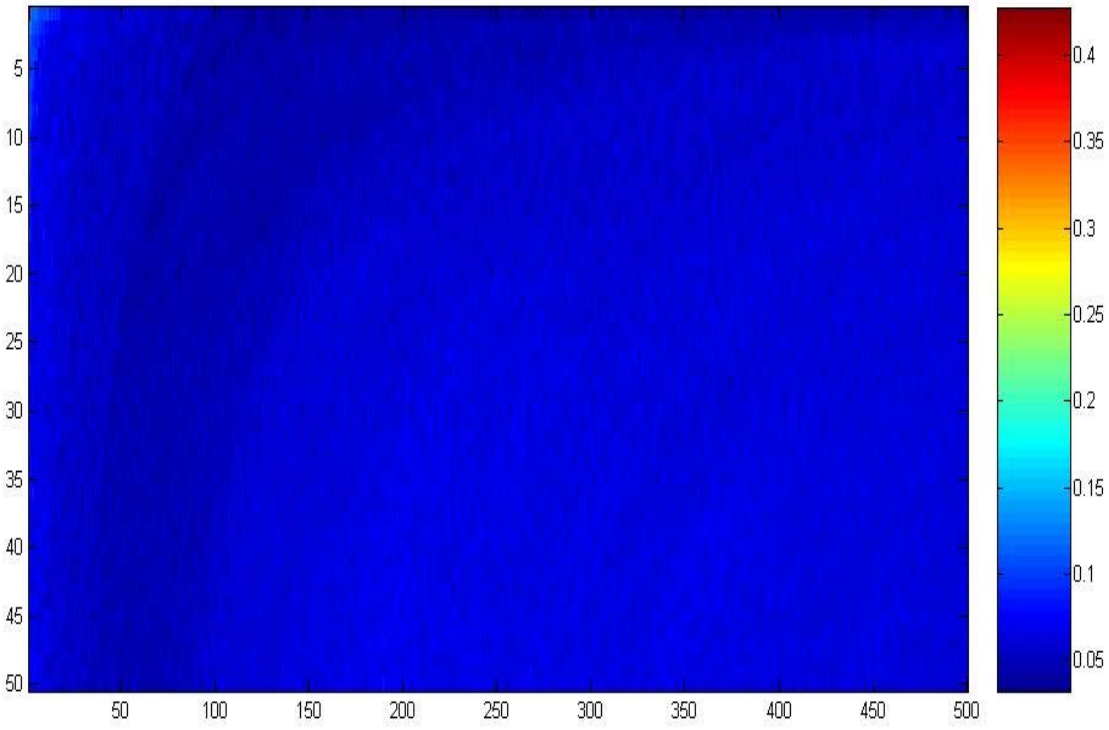


Ilustración 13: Error en función del parámetro C y Gamma de un kernel RBF para la clasificacion de EColi.

Como en el ejemplo de Iris, tenemos varios puntos a remarcar, el primero y más importante vuelve a ser el descenso del error de cuotas del 6.5 % en el mejor de los casos, a un 4.5% en la zona azul

oscuro de la izquierda de la ilustración. La mejora en este caso no es abrumadora, por que de manera aleatoria la columna de ECOC que hemos optimizado sugería una distribución de las clases que podría ser linealmente separable. Dicha situación es, generalmente, poco usual dada la gran cantidad de datos que se entrenan. El otro punto análogamente al ejemplo anterior es la pérdida de esa aleatoriedad previamente comentada y su sustitución por una topología del espacio de soluciones mucho más suave, factor que siempre ayudara al evolutivo a convergir con mayor rapidez. Así pues, el evolutivo en este caso nos muestra lo siguiente:

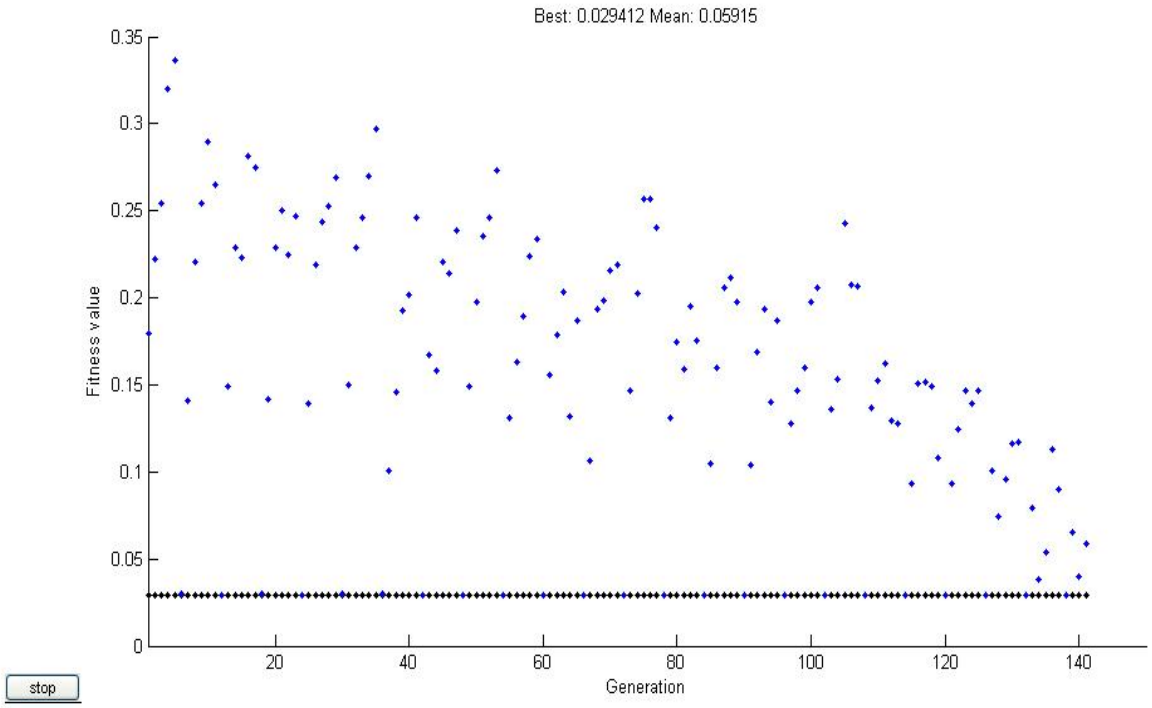


Ilustración 14: Proceso evolutivo de optimización de un kernel RBF en EColi.

Aquí podemos observar como el evolutivo converge a un resultado muy bueno 2.9% de error. Hay que recalcar que la magnitud de ejemplos con la que se trabaja en este dataset empieza a ser grande y su distribución como se ha comentado posteriormente puede ser de topología muy extraña, aun así, un kernel RBF consigue cotas de error muy bajas.

Comentar también como en el apartado anterior la tendencia del algoritmo a converger hacia el mejor valor con el paso de las generaciones. Este hecho que parece ser algo trivial es lo que como se había dicho en apartados anteriores distingue al proceso evolutivo (estocástico) de un mero proceso de búsqueda aleatorio.

3.4 Vowel

Este dataset contiene 10 clases, como se puede ver los datasets fueron escogidos de manera incremental para visualizar como afecta el incremento de la carga de datos en los clasificadores base del nuestro ECOC. Así pues, este dataset consta de 528 ejemplo divididos en las 10 clases comentadas anteriormente. Aplicando el proceso experimental obtenemos los siguientes resultados.

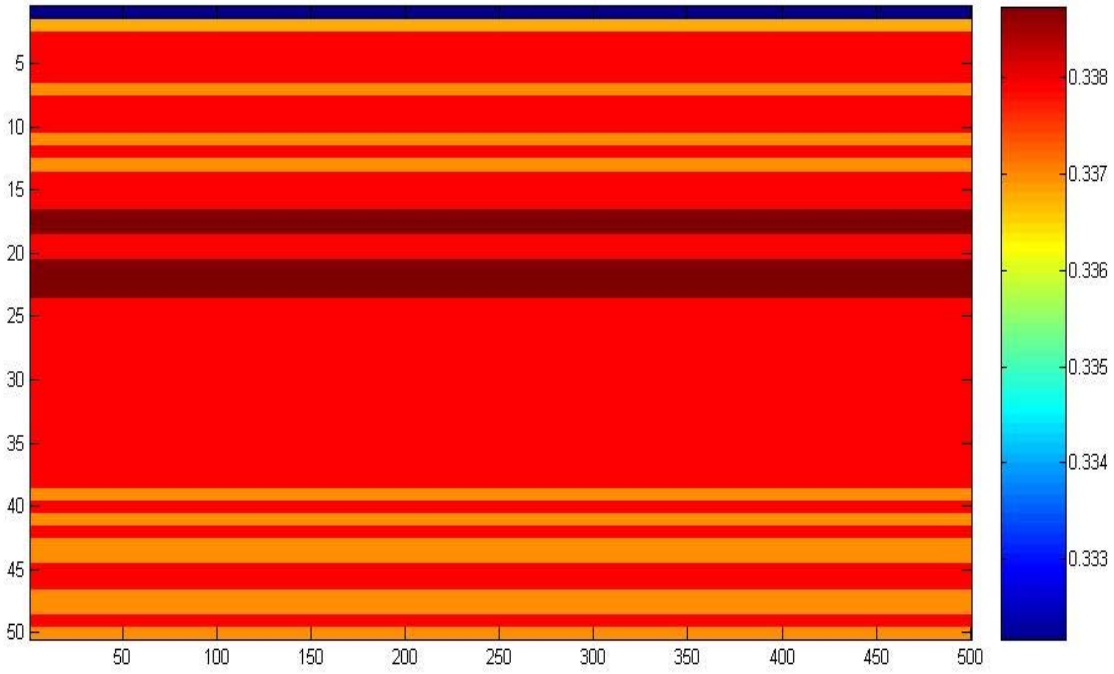


Ilustración 15: Error en función del parámetro C de un kernel lineal para la clasificacion de Vowel.

Como podemos ver, en la ilustración anterior sí se pone de manifiesto la linealidad, perdonando la redundancia, de un kernel lineal. En este ejemplo se puede ver como para distintos grupos de valores del hyperparámetros *C* obtenemos varias franjas de error. Cabe recalcar también que dicho

error sufre una oscilación del 0,5% del parámetro que optimiza la clasificación al parámetro que muestra una menor eficiencia. Esta pequeña diferencia puede ser dada por la sobrecarga de ejemplos por clasificador, ya que cada clasificador aguanta una carga de 528 ejemplos, con distribuciones como antes habíamos dicho completamente desconocidas. No obstante, volviendo a aplicar un kernel RBF se obtuvieron los siguientes resultados:

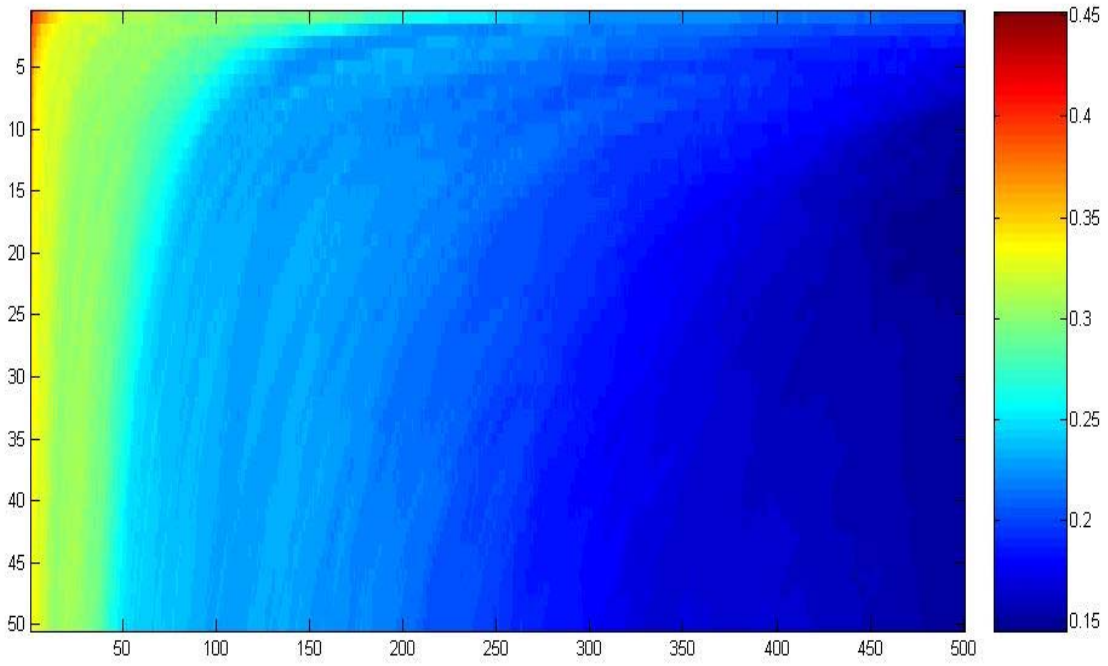


Ilustración 16: Error en función de los parámetros C y Gamma de un kernel RBF para la clasificacion de Vowel.

Como podemos observar y como habíamos visto en los 2 casos anteriores, hay 3 puntos que comentar. El primero, es el descenso del error de un 33% a un 14.8 %, esto nos hace pensar que la distribución que nos daba la columna de ECOC para este clasificador no era linealmente clasificable, de ahí que obtuviésemos un error tan grande. No obstante mediante la utilización de un kernel RBF podemos observar cómo se reduce el error. Por otra parte, comentar también el hecho de la suavidad en la variación del error, lo cual hace que este espacio de búsqueda sea propicio para

que el evolutivo converja hacia unos parámetros sub-óptimos en el peor de los casos. Veamos pues los resultados que obtenemos delegando el proceso a un algoritmo genético:

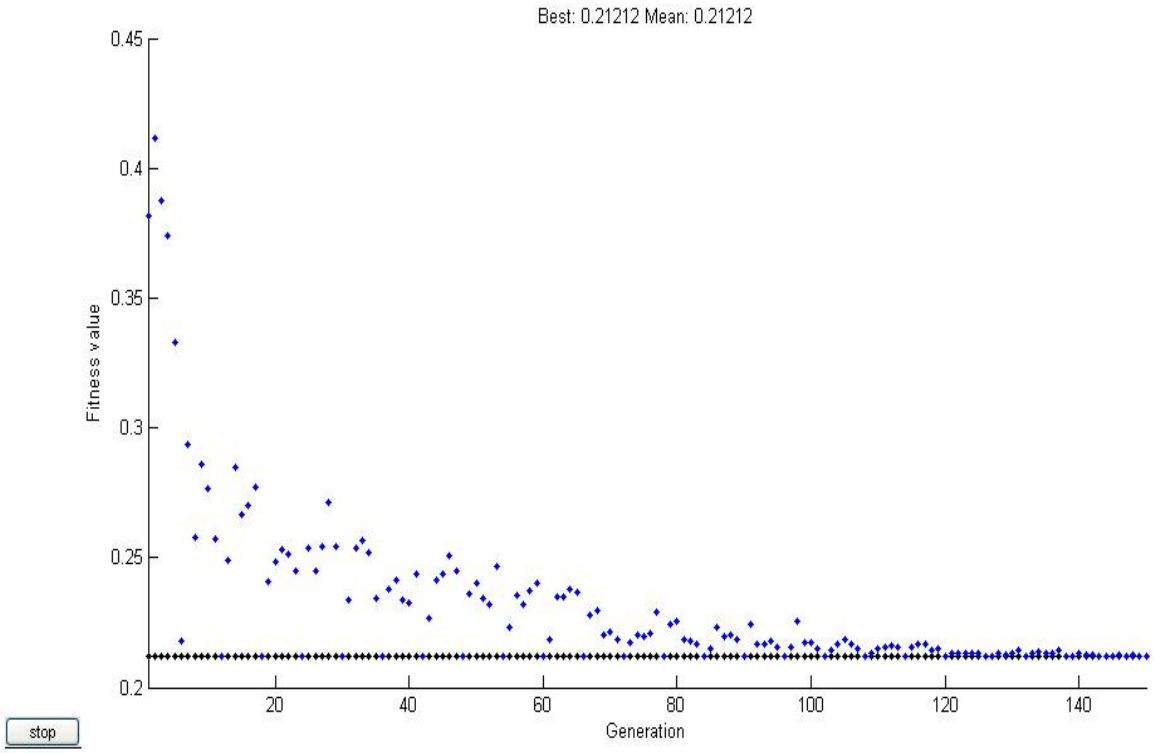


Ilustración 17: Proceso evolutivo de optimización de un kernel RBF en Vowel.

En la figura podemos observar la perfecta convergencia a lo largo de las generaciones hacia el mejor valor. El pensamiento inicial cuando uno utiliza algoritmos genéticos sería pensar que el mejor valor evolucionará convirtiéndose a cada generación en un valor mejor. No obstante, aunque dicha situación se produce muchas veces, se pueden observar otras tantas en la que el mejor valor inicial, el que se obtiene cuando se crean los individuos aleatoriamente en la primera población, se mantiene como mejor valor y se transmite generación a generación gracias al parámetro del elitismo, haciendo que dicha idea de evolución del mejor valor quede substituida por la idea de que

toda la población evoluciona para asemejarse a ese individuo mejor adaptado, que es el par de hyperparámetros que producen un error menor en la clasificación.

3.5 Yeast

Este dataset es el más complejo de todos los que se han escogido para completar la batería de test, consta de 10 clases y de 1484 ejemplos. El dataset está basado en la localización de la proteína “Yeast”. Recalcar que este dataset es uno de los más complejos de la literatura en lo que a problemas de clasificacion se refiere.

Igual que para los anteriores datasets, hacemos pasar a este conjunto de datos por nuestra batería de pruebas obteniendo para un kernel lineal la siguiente topología del espacio de posibles valores del parámetro *C* en un kernel lineal:

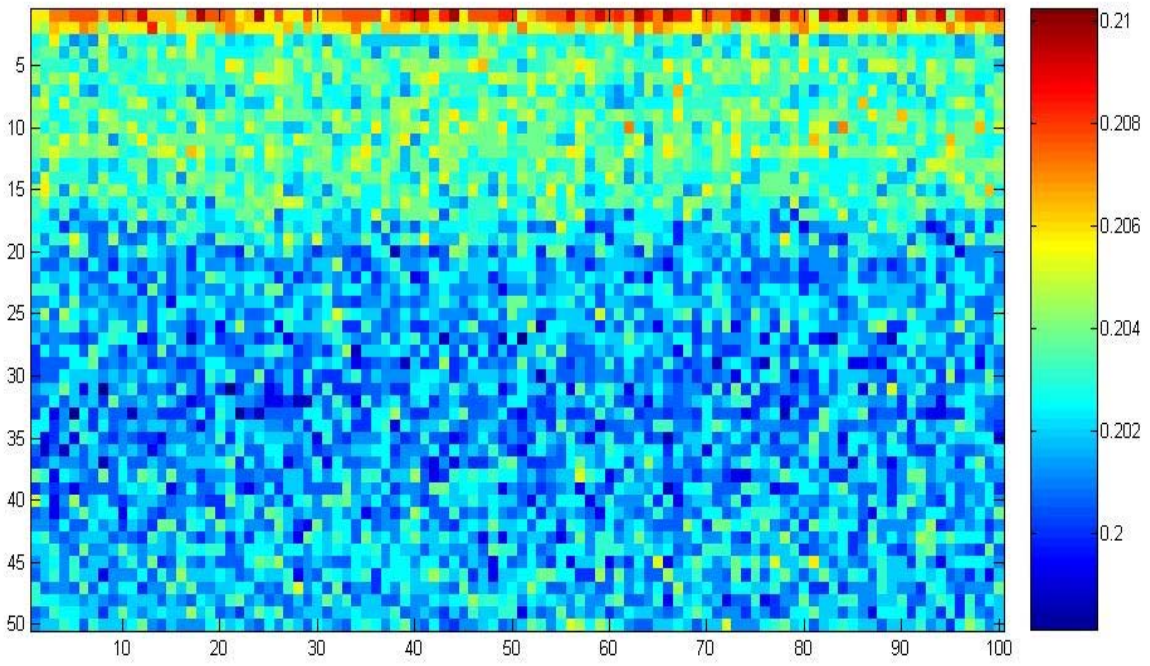


Ilustración 18: Error en función del parámetro C de un kernel lineal para la clasificacion de Yeast.

Como en anteriores apartados podemos observar la linealidad de estos tipos de kernel, en concreto, podemos ver unas franjas de tonos azules oscuros a partir de $C > 20$. No obstante, si nos fijamos en el baremo de error, veremos que el cambio del error, de escoger un parámetro *C* óptimo o sub-optimo a escoger un parámetro no sub-optimo es simplemente de un 1%.

Aplicaremos pues al problema un kernel RBF y obtenemos los siguientes resultados:

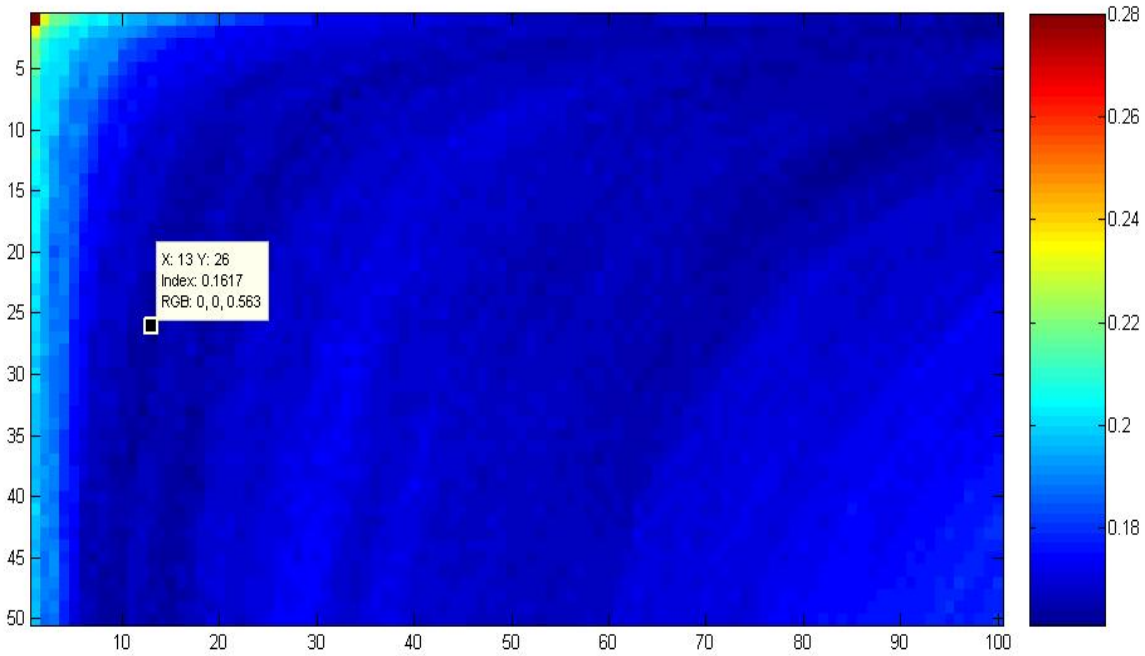


Ilustración 19: Error en función de los parámetros C y Gamma de un kernel RBF para la clasificacion de Yeast.

Como hemos podido observar a lo largo del conjunto de pruebas, la eficiencia que presenta un kernel RBF no tiene comparación con la obtenida por un kernel lineal. Ya no simplemente por la ganancia en el porcentaje de acierto en la clasificación, que en ésta no supera el 5 %, sino por la topología del espacio de soluciones que genera. Dicho espacio, con esas transiciones tan suaves y esa uniformidad son un marco de inmejorables opciones de triunfo para un algoritmo genético.

Así pues, veamos cómo trabaja dicho algoritmo en este dataset.

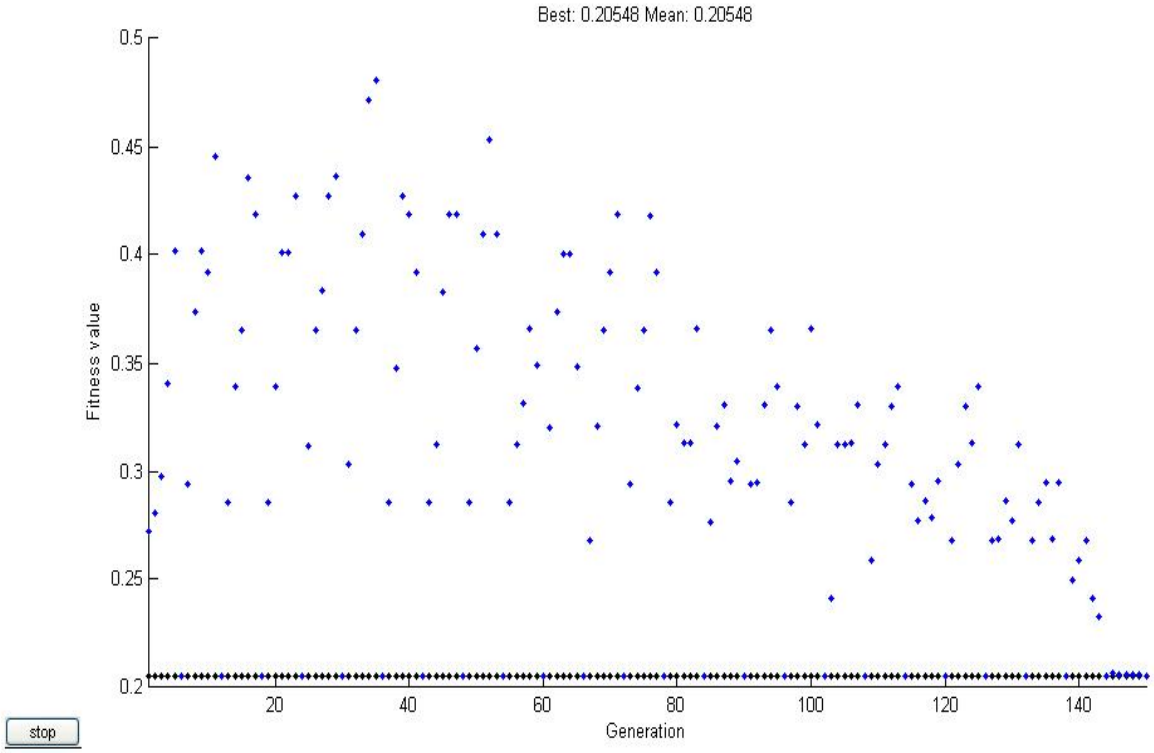


Ilustración 20: Proceso evolutivo de optimización de un kernel RBF en Yeast.

Como hemos podido observar, hay veces en las que el algoritmo genético no encuentra los valores óptimos que minimizan el error en la clasificación. Como se comentan en el apartado de computación evolutiva, dichos algoritmos no aseguran encontrar los parámetros óptimos ni sub-óptimos, lo que si aseguran es una tendencia a converger al mejor valor y una búsqueda no exhaustiva del espacio de soluciones.

Como se ha podido observar a lo largo de todo el proceso de prueba del proyecto, los kernels de tipo RBF presentan un mayor rendimiento que los kernels de tipo lineal en problemas de

clasificación. Por otra parte, también se ha podido ver ampliamente como los procesos evolutivos y en concreto los algoritmos Genéticos tienden a encontrar hyperparámetros óptimos o sub-óptimos sin necesidad de un análisis exhaustivo del espacio de soluciones.

3.6 Conclusiones

Así pues, la suma del proceso presentado y aplicado sobre dichos datasets nos ha reportado dichos resultados en términos de clasificación global. Los resultados obtenidos son comparados con un conjunto de resultados obtenidos en [18].

Dataset	#clasific.	% eficiencia	#clasific. [18]	%eficiencia[18]	Incremento ef.
Iris	2	99.54	3	96.00	3.54
EColi	3	89.64	8	81.47	8.14
Vowel	4	86.84	11	71.77	14.07
Yeast	4	59.36	10	52.17	7.19

Como se puede observar, en la mayoría de los casos se consiguen mejoras bastante importantes, llegando a 14 puntos porcentuales por encima de los niveles presentados en [18] (Vowel), a esta mejora debemos agregar el uso de menos clasificadores para llevar a cabo la tarea, lo cual nos conduce a afirmar que el proceso presentado como SE-ECOC-SVM “Sublinear Evolutive Error Correcting Output Codes Support Vector Machine”, constituye una arma muy poderosa para resolver problemas de clasificación con un número alto de datos.

El uso de un número sublineal de clasificadores respecto el número de clases nos lleva a pensar que este sistema podría ser utilizado en problemas de alta dimensionalidad como los “Large Escale Problems”.

4.Conclusiones y trabajo futuro

Estudio pretendía presentar una alternativa a la resolución de los problemas de multi-clasificación que fuese posteriormente aplicable a problemas de clasificación de muy alta dimensionalidad, como los llamados “Large Escale Problems”. Para ello se ha propuesto la utilización del marco de trabajo de los códigos correctores de errores (ECOC), dado que múltiples estudios han demostrado su eficacia en este tipo de problemas. Con el objetivo de resolver los problemas de escalabilidad de los ECOC manifestados en el estado del arte, una nueva estrategia de codificación ha sido propuesta: ECOC Sublineal. En este nuevo sistema la longitud de palabra clave de clases es la mínima necesaria para distinguir las clases unívocamente, lo que implica el uso de $\log_2 N$ clasificadores para un problema de N clases. Este nuevo número de clasificadores hace viable la clasificación de problemas de alta dimensionalidad.

En lo que a clasificadores base se refiere, se apostó por la utilización de clasificadores SVM, en concreto por el uso de un kernel RBF, ya que dichos clasificadores demuestran un rendimiento sin precedentes en problemas de clasificación binaria donde el conjunto de entrenamiento es amplio y de topología no uniforme.

En lo referente a optimización de los hyperparámetros de los kernel SVM, el estado del arte ha sugerido siempre atacar el problema desde perspectivas analíticas o exhaustivas con el objetivo de encontrar parámetros óptimos que hagan que el clasificador RBF sea eficiente [16] y [17].

De esta manera una novedosa técnica de optimización es propuesta, la cual delega todo el proceso de optimizado a los GA (algoritmos Genéticos), que mediante los principios de la computación evolutiva, tienden a encontrar hyperparámetros óptimos o sub-óptimos de los SVM en tiempos

computacionalmente viables.

Como último punto a destacar del sistema, el problema de la optimización de los ECOC se delega también en un proceso evolutivo, esperando que la topología del espacio de soluciones permita al GA encontrar la codificación de error mínimo.

Finalmente el sistema desarrollado se ha probado en datasets del UCI Machine Learning Repository, comparando los resultados con los que [18] obtuvo en su estudio, y observando sustanciales mejoras, aún y con el uso de un número reducido de clasificadores.

Como trabajo futuro se propone la utilización del sistema en ámbitos de alta dimensionalidad, como los “Large Escale Problems”.

Como ejemplo proponemos llevar a cabo esta tarea en un “dataset” extraído del ODP (Open Directory Project), también conocido como DMoz (Directory Mozilla). Este proyecto está mantenido por voluntarios y consiste en un sitio web que almacena URLs de sitios web propios de Netscape, ya que Netscape es el mantenedor oficial del proyecto. Este directorio básicamente lo que hace es clasificar cada URL según el tópico al que pertenezca. Así pues, URLs que sean de tópicos parecidos estarán dentro del mismo grupo, teniendo este grupo otros subgrupos por debajo con el fin de afinar la clasificación (estructura en árbol). Este directorio se ha hecho más famoso y conocido desde que se sabe que Google lo utiliza como directorio de potenciado de búsqueda y su uso se ha incrementado enormemente en los últimos años. Finalmente comentamos que esperamos mostrar resultados de este problema en el menor tiempo posible, dando por concluido así el estudio.

5.Referencias

[1] C.Darwin, “The Origin of Species”, Londres,1859.

[2] G.Alexander, A.Raja,“The Role of problem classification in Online Meta Cognition”,2006.

[3] Y.Chen, L.Peng, A.Abraham, “Hierarchical RBF Neural Networks for Classification problems”,2001.

[4] J.Kitter, R.Ghaderi, T.WindEatt, J.Matas, “Face verification using error correcting output codes”,2001.

[5] R.Ghani, “Combining Labeled and unlabeled data for text classification with large number of categories”,2001.

[6] D.Lunga, T.Marwala, “Online forecasting of stock market movememt using improved incremental algorithm”,2006.

[7] Q.Wu,P.Sveten,A.Oosterlinck, “Chromosome classification using a multilayer perceptron neural network”,1994.

[8] T.Diettrich, G.Bakiri, “Solving MultiClass learning problems via error correcting output codes”,1995.

[9] S.Rajan, J.Gosh, “An empirical comparison of hierarchical vs two-level approaches to multiclass problems”, 2004.

[10] F.Aiolli, A.Sperduti, “Multiclass classification with multi prototype Support Vector Machines”,2006.

[11] T.Dietterich, G.Bakiri, “Solving multiclass problems via error correcting output codes”,1995.

[12] K.Crammer, Y.Singer, “On the learnbility and design for output codes for multiclass problems”,2000

[13] T.WindEatt, R.Ghaderi, “Coding and Decoding for multiclass learning problems”,2003.

[14] E.Buck, B.Zhang, “SVM kernel optimization: an example in Yeast Protein Subcelullar localization prediction”, XXXX.

[15] N.Ayat, M.Cheriet,C.Sven, “Empirical error based optimization of SVM kernels: Application ti digit image recognition”, 2008

[16] P.Zhang, Z.Li, J.Yang, “A parallel SVM training algorithm on large escale classification problems”, 2005.

[17] Y.Qi, “An Efficient Method for Large Scale Classification Problems”,2008.

[18]X. Baró,“Probabilistic Darwin Machines:A new approach to develop evolutionary object detection systems”,2009.

[19] S. Escalera, P. Radeva, O. Pujol, “On the decoding process in ternary error correcting output codes”,2009.

[20] S.Gunn, “Support Vector Machines for classification and regression”,1998.

6.Anexos

6.1 Anexo A: Contenido del CD

El contenido del CD consta de dos directorios , el primero “/src” contiene los scripts que implementan el proyecto.

El segundo “/test” contiene logs y resultados de ejecución del proyecto para los diferentes datasets.

Para utilizarlo simplemente se ha de ejecutar el script “demoEVOLECOC” con el nombre del dataset que se quiera probar.

ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de Juliol de 2009
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr.

Dr.

Dr.

Amb la qualificació de: