Máster Universitario en Ingeniería Informática

Tratamiento Inteligente de Datos

TRABAJO TEÓRICO

Support Vector Machine

UNIVERSIDAD DE GRANADA

Pablo Alfaro Goicoechea
Carlos Morales Aguilera
Carlos Santiago Sánchez Muñoz

30 de noviembre de 2020

Índice

1. Presentación

Las máquinas de vectores de soporte (del inglés Support Vector Machines, SVM) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T (Wikipedia).

Esta técnica está pensada con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento podemos etiquetar las clases y entrenar SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase.

Más formalmente, SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

En este trabajo se intentará dar una idea de qué problemas puede SVM resolver con garantías y algunas aplicaciones actuales de dicha técnica.

2. ¿Qué son las SVM?

Su traducción es Máquinas de soporte vectorial y son un tipo de modelo de clasificación lineal supervisada. La particularidad de estos algoritmos es que se centran en mejorar la generalización. Esto quiere decir que será más robusto a la hora de clasificar nuevos ejemplos que no se parezcan tanto a los de entrenamiento.

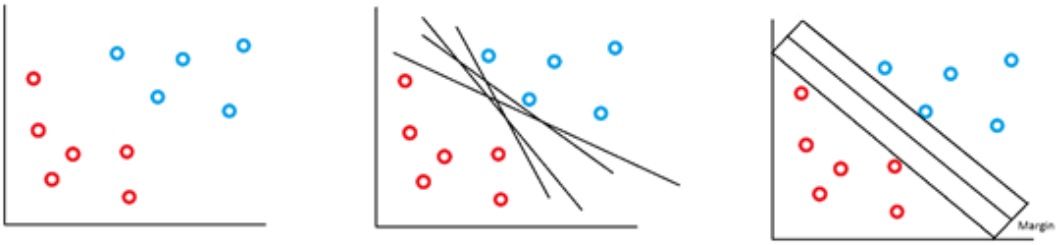


Imagen 1: Ejemplo de SVM [3].

En la Imagen 1 tenemos 3 esquemas. En el primero se observan los ejemplos que tenemos para entrenar y según su color, rojo o azul, son de una clase o de otra. En el esquema central se pueden ver diferentes fronteras de decisión que clasifican correctamente los ejemplos. Alguna de esas fronteras de decisión están no guardan demasiado margen con los ejemplos de las clases y eso puede ser problemático a la hora de clasificar nuevos ejemplos que puedan estar cerca de la frontera escogida. Finalmente en la imagen de la derecha podemos ver la frontera que trazaría un SVM. Dejaría el mismo margen entre los elementos de las distintas clases y de esta manera tendría menos problemas para clasificar ejemplos centrales.

3. Fundamentos

Para clasificar el conjunto de los datos de entrenamiento crea un hiperplano de separación que cumple con las siguientes restricciones para todos los ejemplos.

- $\theta^T x^{(i)} \geq 0$ si $y_i = +1$
- $\theta^T x^{(i)} \leq 0$ si $y_i = -1$

En estas ecuaciones el valor de θ sería el peso de cada una de las variables que evaluemos para cada ejemplo, x^i es el valor que tiene el ejemplo para la variable i y finalmente la variable y_i es la clase asociada al ejemplo i. Este hiperplano que se crea no es único, existen infinitos que pueden cumplir con estas condiciones. Esto se puede apreciar en la siguiente imagen.

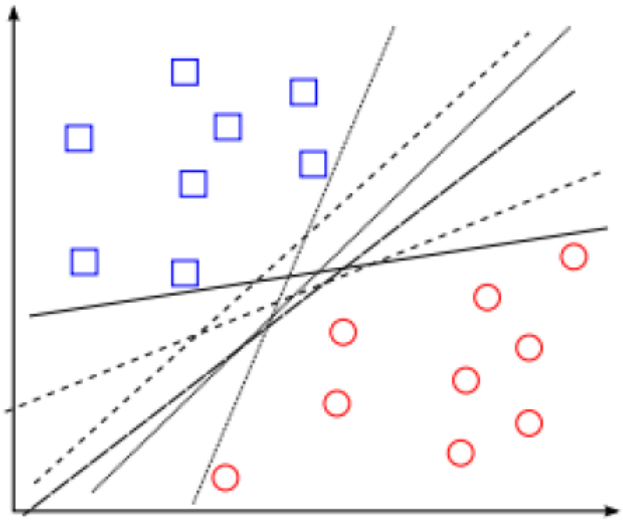


Imagen 2: Posibles hiperplanos.

El objetivo de los SVM es encontrar el mejor hiperplano de los posibles, minimizando el error en la generalización al clasificar nuevos ejemplos como se ha explicado en el Apartado 2.

3.1. Principios matemáticos

Para hablar de los SVM hay primero que ver cómo funcionan los modelos de regresión logística. Estos clasificadores estiman usando una función sigmoide construida de la siguiente manera:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- Si $y = 1$, buscamos que $h_\theta(x) \approx 1, \theta^T x \gg 0$
- Si $y = 0$, buscamos que $h_\theta(x) \approx 0, \theta^T x \ll 0$

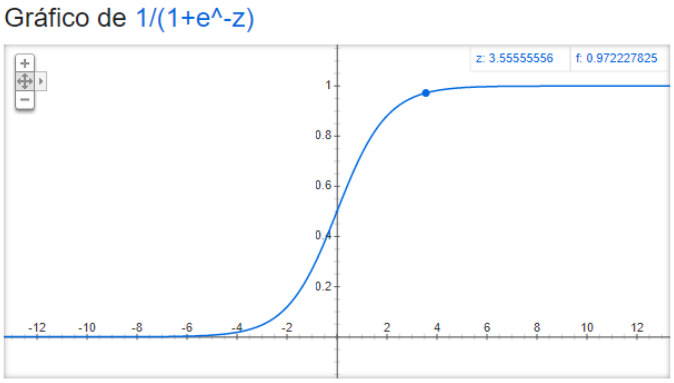


Imagen 3: Función sigmoide.

Para poder evaluar estos modelos se establece una función de coste. Esta función nos indica lo bien que está clasificando los ejemplos. Para calcular el conste de clasificar un ejemplo (X, y) se utiliza la siguiente fórmula:

$$\text{Coste de un ejemplo} = -(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$$

$$\text{Coste del conjunto} = \frac{1}{m}[\sum_{i=1}^{m}(-\log h_\theta(x^{(i)}))+(1-y^{(i)})(-\log(1-h_\theta(x^{(i)})))]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

En la segunda ecuación, la que sirve para calcular el coste de todo el conjunto, el valor de m es el número de ejemplos y el parámetro n son las variables de estudio para cada ejemplo. Por otro lado λ es un parámetro que de regularización escogido. Esto se va a explicar a continuación. Una buena manera de comparar diferentes modelos de regresión logística es ver cuál es el que minimiza el coste de clasificar el conjunto de entrenamiento.

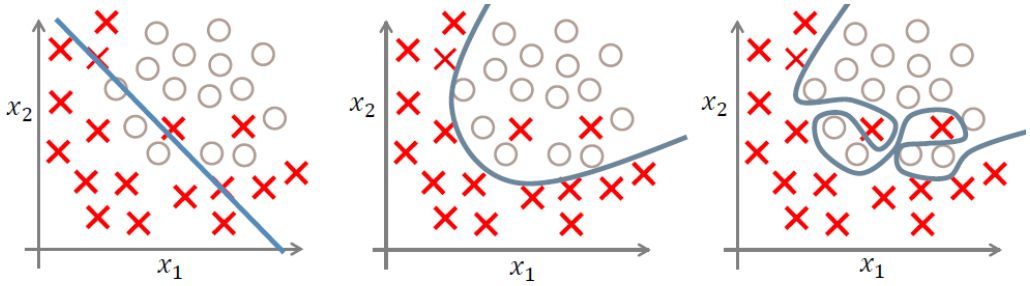


Imagen 4: Variación de λ.

En la Imagen 4 podemos ver cómo cambiaría el resultado del hiperplano generado por el modelo según el valor de λ que elijamos. Si se elije un valor muy bajo, tendremos un resultado que se aproximará al de la izquierda. Tendremos un modelo con un bias alto que no se ajusta a los ejemplos, como el de la figura de la izquierda. Si por el contrario, tomamos un λ alto, le daremos mucho peso a los ejemplos mal clasificados y de esta manera estaremos forzando a que clasifique bien todos los ejemplos. Haciendo esto tendremos un modelo con una varianza alta, que generará hiperplanos poco realistas, muy dependientes de los ejemplos de entrenamiento, tal y como se ve en la figura de la derecha. A esto se le conoce como sobreaprendizaje y hace que a la hora de clasificar nuevos ejemplos pueda no hacerlo correctamente. Lo que se intenta es buscar un valor de λ que sea capaz de trazar un hiperplano que no aprenda de ejemplos que puedan estar mal o

que no sigan la tendencia general. Se obtendría una frontera como la de la figura del centro.

Para calcular el coste del modelo SVM, se hace de una manera un poco diferente. Utilizo la función *cost* para 0 o 1 dependiendo de la clase del ejemplo y se elimina el $\frac{1}{m}$ porque no cambia el resultado. Además, en esta ecuación el parámetro de regularización es C. Este parámetro es $C = \frac{1}{\lambda}$.

$$\text{Coste SVM} = C\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

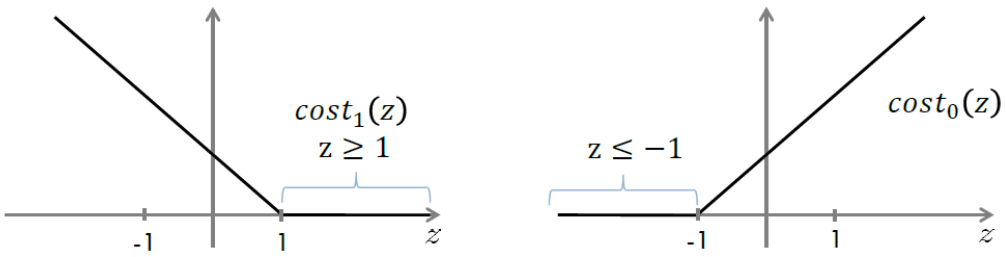


Imagen 5: Función para calcular el coste de clasificar un ejemplo con un SVM.

Existen distintas funciones que evalúan el coste de clasificar un ejemplo, pero todas deben de tener una forma como la del las de la Imagen anterior. De esta forma el coste aumenta si el ejemplo se acerca a la frontera de decisión y se reduce si se aleja. Eligiendo un valor de C alto fuerzas que la primera parte sea igual a 0 por lo que te centras en acertar con un margen se seguridad. Teniendo lo anterior en cuenta, lo que se buscaría minimizar es esta parte: $\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$.

3.2. Fronteras de decisión

Asumiendo una simplificación para un conjunto bidimensional de $\theta_0 = 0$ y $n = 2$. Los resultados son extensibles a conjuntos de otras dimensiones. Se podría calcular lo siguiente:

$$\frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \Rightarrow \frac{1}{2}(\theta_1^2+\theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2+\theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$$

De esta manera vemos que la SVM se centra en minimizar la norma de θ. Esto está sujeto a:

$\theta^T x^{(i)} \geq +1$ si $y^{(i)} = +1$

$\theta^T x^{(i)} \leq -1$ si $y^{(i)} = -1$

Utilizando la teoría de los productos escalares, $\theta^T x^{(i)}$ se podría ver de esta manera:

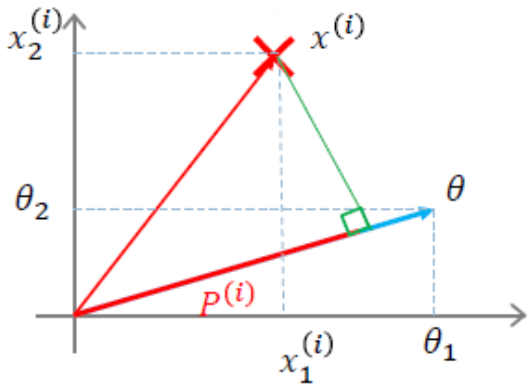


Imagen 6: $\theta^T x^{(i)}$.

Viendo lo anterior, se puede concluir lo siguiente:

$\theta^T x^{(i)} = P^{(i)} \|\theta\|$

Por lo tanto para calcular el coste del SVM habiendo elegido un valor de C alto el modelo estaría sujeto ahora a:

$P^{(i)} \|\theta\| \geq +1$ si $y^{(i)} = +1$

$P^{(i)} \|\theta\| \leq -1$ si $y^{(i)} = -1$

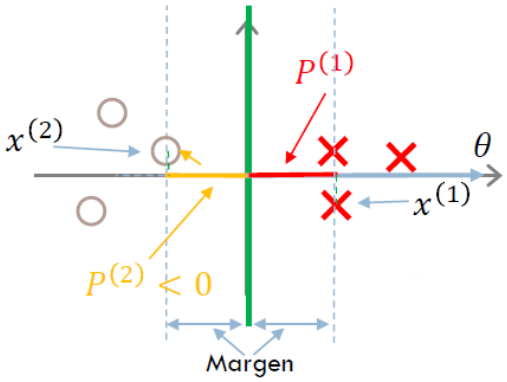


Imagen 7: Cálculo del margen.

Por tanto se busca que las proyecciones de los ejemplos sobre θ sean lo más grandes posibles. Esto es lo que provoca que la SVM busque márgenes grandes, haciendo esto, la SVM puede obtener una norma de θ menor.Al final es un problema de minimización cuadrática con restricciones.

3.3. Truco del kernel

En ocasiones, la frontera que hay que establecer no es lineal y para poder elegir el hiperplano hay que recurrir al Kernel. La idea intuitiva del uso de los kernels es que de esta manera se mapea el espacio de características a uno mucho mayor, el hiperplano de margen máximo se busca en este nuevo espacio.

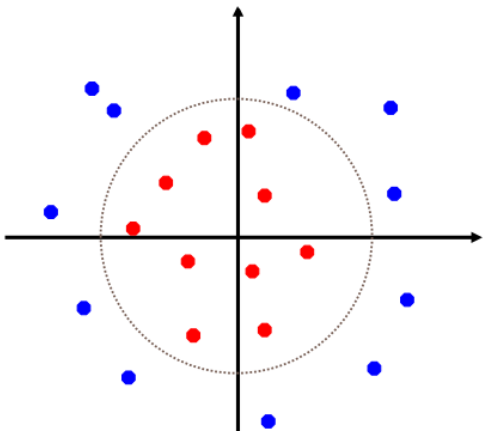


Imagen 8: Espacio original.

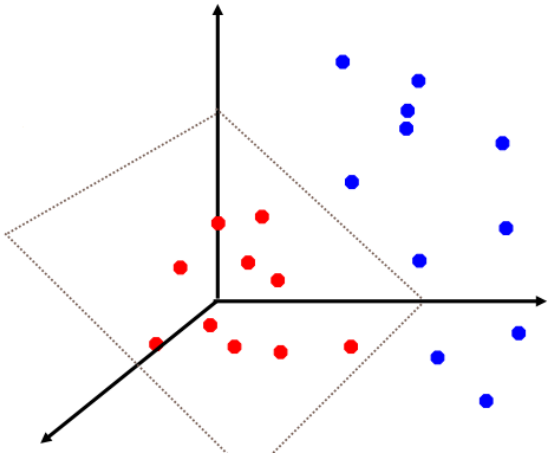


Imagen 9: Espacio nuevo.

Gracias al uso de las funciones kernel, se evita que el coste computacional aumente. El resultado de la función kernel entre dos ejemplos es el mismo que el de trasladar cada ejemplo por separado a un espacio de características mucho mayor y luego calcular su producto escalar. Es decir, en realidad no necesitamos calcular el nuevo conjunto de características para cada ejemplo.

4. Aplicaciones

En este apartado se van a detallar algunas aplicaciones actuales que tiene esta técnica de aprendizaje así como los problemas tipo de la misma.

4.1. Problemas tipo

Las máquinas de vectores de soporte eran muy utilizadas antes de la era del aprendizaje profundo. Para muchas aplicaciones se prefería el uso de SVM en lugar de redes neuronales. La razón era que el razonamiento matemático subyacente en SVM se entiende de muy bien y la propiedad de obtener el margen de separación máximo resultaba muy atractivo. Las redes neuronales podrían realizar clasificación de forma equivocada como hemos visto en los ejemplos anteriores.

SVM funciona bien para problemas de clasificación y también de regresión, tanto lineal como no lineal. Este modelo, al igual que todos, tiene sus **ventajas y desventajas**. En este artículo se encuentran algunas. Las ventajas de SVM son:

- SVM trabaja bien cuando existe un margen claro de separación de las clases.
- SVM es más efectivo en espacios de dimensión alta.
- SVM ofrece buenos resultados en casos donde el número de dimensiones en mayor al número de ejemplos.
- SVM es relativamente eficiente en memoria.

Algunas desventajas de SVM son:

- SVM no es adecuado para grandes conjuntos de datos.
- SVM no da buenos resultados cuando el conjunto de datos tiene ruido o se solapan las clases.
- En los casos en que el número de características para cada punto de datos excede el número de muestras de entrenamiento, SVM tiene un rendimiento inferior.
- Como el clasificador de vectores de soporte funciona poniendo puntos de datos, por encima y por debajo del hiperplano de clasificación no hay una explicación probabilística para la clasificación.

En general, por el teorema No Free Lunch, este algoritmo es en media sobre todos los problemas igual al resto. Para ciertos **problemas tipo** es sabido que SVM ofrece resultados satisfactorios:

1. Reconocimiento óptico de caracteres
2. Detección de caras para que las cámaras digitales enfoquen correctamente filtros de *spam* para correo electrónico.
3. Reconocimiento de imágenes a bordo de satélites (saber qué partes de una imagen tienen nubes, tierra, agua, hielo, etc.)

Actualmente, las redes neuronales profundas tienen una mayor capacidad de aprendizaje y generalización que los SVM. Se ha consultado este enlace.

4.2. Aplicaciones actuales

En esta subsección vamos a dar una introducción a algunas de las aplicaciones actuales de SVM. Las referencias bibliográficas están sacadas de [1] para la primera aplicación y de [4] el resto.

4.2.1. Aplicación de las Máquinas de Soporte Vectorial (SVM) al diagnóstico clínico de la Enfermedad de Párkinson y el Temblor Esencial

Los enfermos de Párkinson (EP) y de temblor esencial (TE) suponen un porcentaje importante de la casuística clínica en los trastornos del movimiento, que impiden a los sujetos afectados el llevar una vida normal, produciendo discapacidad física y una no menos importante exclusión social en muchos de los casos. Las vías de tratamiento son dispares, de ahí que sea crítico acertar con precisión en el diagnóstico en las etapas iniciales de la enfermedad. Hasta la actualidad, los profesionales y expertos en medicina, utilizan unas escalas cualitativas para diferenciar la patología y su grado de afectación. Dichas escalas también se utilizan para efectuar un seguimiento clínico y registrar la historia del paciente. En este trabajo se propone la utilización de clasificadores binarios centrados en las Máquinas de Soporte Vectorial (SVM) para obtener un diagnóstico diferencial entre las dos patologías de temblor mencionadas.

El incremento de la esperanza de vida experimentada en los países desarrollados y en desarrollo en los próximos años, se traduce en un envejecimiento poblacional gradual. Se estima que en el periodo 2013- 2050 la franja de la población mayor de 60 años se incremente a 2020 millones, aumentando en un promedio del 21 % a nivel mundial (United Nations, 2013). El envejecimiento de la población, trae consigo una serie de cambios, especialmente en la salud, con un incremento de las enfermedades crónicas y neurodegenerativas que afectan a la tercera edad, como son la Enfermedad de Párkinson (EP) y el temblor de acción o el Temblor Esencial (TE).

La potencial relación que puede existir entre ambas enfermedades, en muchos casos puede llevar a confusiones y aumentar la probabilidad de error en el diagnóstico, hasta un 40 %, lo que conduce a la prescripción de terapias erróneas. Es por ello que se justifica la necesidad de implementar métodos objetivos y precisos para analizar y cuantificar el temblor, y en base a su medida, clasificarlo. El objetivo es brindar a los expertos de la medicina una herramienta práctica y eficaz que permita automatizar la clasificación entre pacientes con TE y la EP, utilizando SVM.

La base de datos de pruebas se obtuvo a partir de un sistema basado en un dispositivo háptico en conjunción con un protocolo de pruebas desarrollado y consensuado junto con profesionales especialistas y expertos (neurólogos y neurofisiólogos) en el área de ciencias de la salud.

Utilizando SVM obtuvieron buenos ratios de acierto, presentando un 94 % para kernel

lineal, un 95 % para polinomial y un 98 % para kernel gaussiano. En este caso los registros se obtuvieron mediante la medida de aceleración de las manos de una muestra total de 25 sujetos. En la sección de conclusiones expondremos los resultados obtenidos con SVMs aplicando la herramienta de ACP (ya que no son necesarias la totalidad de características) y la eliminación de atípicos con Curtosis y Mahalanobis, llegando a un ratio de acierto del 100 % en la clasificación entre TE y EP.

4.2.2. Fusión de Información Acústica e Ideolectal mediante SVM's para Tareas de Reconocimiento de Locutor en Habla Conversacional

En la actualidad la mayoría de los sistemas de reconocimiento de locutor están basados casi exclusivamente en información acústica de bajo nivel. Los Modelos de Mezclas de Gausianas (GMM) adaptados mediante criterios MAP a partir de un Modelo Universal de Locutor (UBM) representan el estado del arte en reconocimiento de locutor independiente de texto. Esta estrategia permite obtener buenas tasas de reconocimiento de locutor pero posee el inconveniente de estar sometida a factores que degradan notablemente su rendimiento tales como la variabilidad del canal y el ruido ambiente. Sin embargo, hay otros tipos de información en la señal de voz relacionados con la identidad del locutor que no se encuentran bajo la influencia de dichos factores.

Entre otros patrones, las diferencias ideolectales entre diferentes locutores han demostrado ser una de las fuentes de información de la identidad del locutor más prometedoras. Para explotar estos beneficios potenciales es necesaria una considerable cantidad de habla que permita llevar a cabo un adecuado entrenamiento de los modelos de Bigramas. Por lo tanto, sólo un subconjunto de las posibles tareas de reconocimiento de locutor presenta un escenario propicio para la fusión de diferentes niveles de información de locutor. Un buen ejemplo de éstas es el área de la acústica forense, y más concretamente aquellas tareas en las que un sistema de adquisición remoto puede ser implementado fácilmente.

Particularmente, el escenario español se ajusta completamente a esta situación ya que, una vez que un juez ha emitido una autorización legal para llevar a cabo una investigación, la adquisición de grandes cantidades de datos se puede llevar a cabo de manera automática mediante "pinchazo" de la línea telefónica. Además, las transcripciones manuales de la información de voz son requeridas por el juez para proceder al seguimiento del caso.

Kittler et al. consideraron en la tarea de combinar clasificadores dentro de un marco probabilístico bayesiano. De entre el conjunto de combinadores (suma, producto, max, min,...) propuestos, la regla de la suma (adición de las puntuaciones individuales normalizadas a un rango [0.,1]) demostró ser el que mejores prestaciones ofrecía en el conjunto de pruebas experimentales.

A este conjunto de estrategias nos referiremos como *fusión basada en reglas simples*. La fusión de información de diferentes niveles también puede ser tratada como un problema de clasificación de patrones de dos clases si las puntuaciones generadas por los clasificadores individuales se consideran como patrones de entrada que deben ser aceptados/rechazados (para la tarea de verificación). Bajo este punto de vista, cualquier sistema basado en aprendizaje puede ser utilizado como una estrategia de fusión. En contribuciones recientes basadas en este enfoque , el paradigma SVM ha demostrado ser superior

a otras estrategias de *fusión basada en aprendizaje* (como redes neuronales). Estos resultados objetivos han motivado la fusión de información acústica e ideolectal mediante SVM.

Experimento

Más de nueve horas de habla conversacional en español han sido grabadas durante un periodo de tiempo de dos meses, asegurando así suficiente variabilidad tanto para las características acústicas como ideolectales. La base de datos está compuesta por nueve locutores (7 masculinos, 2 femeninos) cuyos ficheros de voz se han dividido en un subconjunto de entrenamiento y otro de reconocimiento. Para los experimentos basados en ideolectos, el subconjunto de ficheros de transcripciones para el entrenamiento se dividió en tres ficheros de texto de una duración equivalente a 30, 15 y 10 minutos de habla. El subconjunto de prueba se dividió en 5 segmentos de tres minutos de habla. Con respecto a la información acústica se siguió un esquema de división similar a los desarrollados en NIST . Se crearon segmentos de 2 minutos para el entrenamiento de los modelos acústicos del locutor y los ficheros de prueba contenían 20 segundos de audio extraídos de los 5 segmentos de prueba disponibles.

Para los experimentos de fusión basados en SVM se uso el método leave-one-out con el fin de maximizar el tamaño del conjunto de datos de entrenamiento y prueba del SVM. Las puntuaciones del sistema acústico e ideolectal de cada usuario se combinaron por medio de un SVM entrenado sobre el resto de usuarios generando así 5 puntuaciones fusionadas de usuarios y 40 de impostores. Esta estrategia se desarrolló para cada uno de los 9 usuarios obteniéndose así $5 \times 9 = 45$ puntuaciones fusionadas de usuarios y 40×9=360 de impostores. El Kernel utilizado fue de tipo gaussiano y se evaluaron diferentes configuraciones del parámetro σ^2 con el objetivo de alcanzar un buen grado de generalización.

Las técnicas de fusión mediante SVM proporcionan un excelente resultado a la hora de combinar información acústica con modelos del lenguaje para tareas de reconocimiento de locutor. En comparación con el sistema básico que mejores resultados a mostrado (acústico) se ha conseguido una mejora de al menos el 40 % en todos los posibles puntos de trabajo del sistema mediante la fusión con SVM. Por el contrario, las estrategias de fusión basadas en reglas sencillas (suma, producto) no fueron capaces de obtener ningún beneficio de la combinación de las diferentes fuentes.

4.2.3. Sistema de reconocimiento facial para control de acceso automático

Los rasgos faciales a reconocer, son una característica biométrica única de cada individuo, como también lo son las huellas digitales o la forma del iris. Al igual que un sistema biométrico general, nuestro sistema actúa de la siguiente manera: tomamos una muestra de la característica biométrica a reconocer, que en nuestro caso será una imagen. La entrada del sistema, es sometida a un procesado que extrae sus rasgos característicos. Una vez obtenidos estos rasgos característicos, se comparan con los previamente almacenados y el resultado de la comparación verifica o descarta la identidad de la persona.

En el sistema implementaddo, la extracción de parámetros biométricos, consistió en el cálculo de los parámetros ASM (*Active Shape Models*) y AAM (*Active Appereance Models*). Una vez obtenidas las características biométricas, se procede a la comparación de éstas,

sin embargo, existen muchas posibles formas de comparación de ellas. Se ha utilizado un sistema de decisión SVM multiclase, tanto lineal como no lineal.

La extensión al problema multiclase en el sistema, se implementó de dos maneras distintas: SVM uno vs resto (1 vs R) y SVM- DAG (*Directed Acyclic Graph*). El sistema SVM uno vs resto (1 vs R) consiste en realizar comparaciones binarias sucesivas entre cada una de las clases y el resto de ellas. De esta forma, es de suponer que si una de las clases fuera la correcta, sólo esa comparación nos daría un resultado positivo. Esta extensión del problema binario, presenta la ventaja de permitir el resultado de "no perteneciente a ninguna de las clases". También existe la posibilidad de que, debido a un error, varias de las comparaciones fueran positivas, y por tanto habría que realizar algún tipo de "desempate". Este sistema necesitaremos n máquinas y en cada una de ellas utilizaremos todas nuestras muestras.

En el caso SVM- DAG (*Directed Acyclic Graph*), para extender el problema binario, se construye un gráfico de manera que cada nodo del gráfico representará una máquina SVM. Cada uno estos nodos dispondrá de dos ramas, una por cada posible decisión tomada por el SVM. De esta manera, el gráfico completo irá tomando una estructura piramidal. Con esta estructura, iremos descartando una de las posibles categorías en cada uno de los niveles del gráfico. En este caso utilizaremos $n(n-1)/2$ máquinas, pero en cada una sólo utilizaremos las muestras de dos clases, con lo que cada una de las máquinas será entrenada más rápidamente que en el caso anterior.

En las primeras pruebas llevadas a cabo en este trabajo se han utilizado un conjunto de imágenes. Dicho conjunto esta formado por imágenes de formato RAW y tamaño 768×576. Estas imágenes corresponden a 75 individuos varones y a 58 mujeres, disponiéndose de 4 fotografías de cada uno de ellos. Estas 4 imágenes se corresponden con 4 expresiones faciales: neutral, sonriente, seria y bostezando. Con este conjunto de imágenes de pruebas en el que sólo disponemos de cuatro imágenes por persona, no es posible utilizar decisores basados en SVM. Se usaron decisores Euclídeos que tienen un acierto del $50\,\% - 60\,\%$.

En este caso, se sintetizarón nuevas imágenes para cada individuo, 6 en concreto, con lo que tendremos 10 imágenes (con las 4 de que ya disponíamos). Consideraremos las nuevas imágenes como nuevas expresiones. El acierto de SVM 1 vs R es del $90\,\%$ y el de *Directed Acyclic Graph* del $80\,\% - 90\,\%$.

El último conjunto de imágenes es de expresiones reales, es decir, adquiridas por una cámara. Como era de esperar, los resultados son peores, puesto que influyen las imperfecciones en la captación de las imágenes. Aun así, los resultados son satisfactorios y pueden mejorarse con un sistema de adquisición más depurado.

4.2.4. Un Modelo de Recuperación de Información basado en SVMs

Se va a presentar un modelo que puede ser utilizado dentro del campo de la recuperación de información una vez que este haya sido entrenado con un conjunto de consultas que contengan aquellos documentos que se conocen como relevantes. El modelo propuesto puede llegar a mejorar los resultados obtenidos con otros modelos estándares.

Esta aplicación propone una nueva manera de abordar el problema de la jerarquización de documentos basada únicamente en la estadística de las palabras presentes en un documento, la consulta y toda la colección en general. Así, mediante el uso de clasificadores capaces de adaptarse, se pueden generar modelos con ventajas importantes en algunos casos. En particular, se propone una nueva forma de enfrentar problemas de jerarquización de documentos como un problema de clasificación de vectores, el cual es posteriormente resuelto con SVMs.

Los clasificadores, como los SVMs han sido utilizados para la clasificación de documentos de manera muy eficiente, pero su utilidad no ha sido comprobada dentro del área de la recuperación de información para la jerarquización de documentos. Es por ello que se propone una transformación que mapea el proceso de la recuperación de información en un nuevo espacio vectorial en donde un clasificador basado en SVMs es entrenado para aprender el concepto de similitud frente a los documentos.

Dada una consulta q_j , la cual esta compuesta por los pesos de cada uno de los términos que la forman u_{ij}, y un documento d_k, formado por el peso de sus términos $v_{i,k}$, entonces buscamos generar una medida de similitud que sea apropiada para la jerarquización de documentos.

El primer enfoque elegido para generar esta medida de similitud se basó en la idea siguiente: la construcción de un conjunto de datos de entrenamiento, constituida por vectores $z_{j,k} = u_j/v_k$ formado por la concatenación del vector de la consulta y el documento. Así al vector z_j, k se el asoció una etiqueta y_i dada por $y_i = +1$ si el documento es relevante al la consulta y $y_i = -1$ si no lo es.

Posteriormente este conjunto de vectores sirven para entrenar al clasificador, de tal forma que, dada una nueva consulta q_{i^i}, un documento d_k pueda ser clasificado como relevante o irrelevante. Desafortunadamente, esta simple aproximación no permite obtener los resultados esperados debido a la gran cantidad de pares consulta-documento que es utilizada para entrenar el modelo.

Por esta razón, fue necesaria otra manera de abordar este problema. Y así para generar una nueva solución observamos que los términos contenidos en las consultas son los más importantes para determinar la similitud de un documento. Entonces, los términos presentes en la consulta deben aparecer en posiciones fijas en los vectores que son utilizados para entrenar al modelo.

La nueva propuesta reordena el vector $z_{j,k}$ de tal forma que los términos en la consulta se encuentren en posiciones fijas, dando así al algoritmo de aprendizaje una oportunidad de obtener la medida de similitud. En particular, el algoritmo propuesto en este trabajo obtiene un nuevo vector, el cual se basa en el reordenamiento de los componentes del vector de la consulta sumados con los valores de los documentos en las posiciones correspondientes en orden decreciente, agregándole un valor extra a los valores de las consultas para así lograr obtener los valores más altos de las consultas en los primeros lugares. Por último se eliminaron los valores que vienen después del reordenamiento que no dan ninguna aportación al vector.

Una vez que el clasificador de SVM ha sido entrenado, y un modelo de clasificadores

ha sido generado, es posible obtener la clasificación de nuevos documentos a partir de diferentes consultas generando el vector consulta-documento descrito.

Experimento

Se ha usado la colección MEDLINE que contiene un conjunto de resúmenes de la Librería Nacional de Medicina. La colección contiene 1033 documentos y un conjunto de 30 consultas junto con su lista de los documentos que son considerados relevantes. En el primer experimento, se entrenó un SVM con kernel polinomial de tercer grado con las 30 consultas contenidas en la colección y sus documentos relevantes. El clasificador resultante fue utilizado para jerarquizar las mismas 30 consultas.

5. Ejemplo implementación

El ejemplo de implementación se ha realizado con R, y exportado con Rmarkdown. Se puede consultar al final de este documento.

6. Comparativa entre algoritmos de clasificación

Los problemas de clasificación consisten en reconocer, comprender y agrupar elementos en categorías con una seria de características que identifican a cada una. Por lo que a la hora de escoger un algoritmo el punto principal es evidentemente el conjunto de datos que se trata. Mediante un análisis exploratorio podemos identificar las diferentes relaciones, dependencias y características de los datos, los cuales definen las condiciones para poder utilizar un algoritmo de clasificación u otro.

El número de algoritmos de clasificación es enorme y su aplicación depende de las condiciones que implican los datos, por lo que para poder entender la importancia de *SVM* realizaremos un breve estudio de cuales son las características idóneas para el mismo y que diferencias presenta frente a otros algoritmos [2].

6.1. Regresión Logística

Es un algoritmo de aprendizaje automático para clasificación. Las probabilidades se calculan utilizando una función logística, y estas probabilidades describen los posibles resultados.

Ventajas	Desventajas
Útil para comprender la influencia de varias variables independientes en una sola variable de resultado.	Funciona solo cuando la variable predicha es binaria. Asume que todos los predictores son independientes entre sí. Asume que los datos no tienen valores perdidos.

6.2. Naive Bayes

Basado en el teorema de Bayes con la suposición de independencia entre cada par de características. Los clasificadores Naive Bayes funcionan bien en muchas situaciones del mundo real, como la clasificación de documentos y el filtrado de spam.

Ventajas	Desventajas
Requiere una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios. Son extremadamente rápidos en comparación con métodos más sofisticados.	Existen algoritmos que clasifican mejor a pesar de tener un mayor tiempo de ejecución.

6.3. Gradiente descendente estocástico

Simple y muy eficiente para adaptarse a modelos lineales. Útil cuando el número de muestras es muy grande. Soporta diferentes funciones de pérdida y penalizaciones por clasificación.

Ventajas	Desventajas
Eficiencia y facilidad de implementación.	Requiere una configuración de hiperparámetros. Sensible a normalización.

6.4. K-Nearest Neighbros (KNN)

Se considera un tipo de aprendizaje "*lazy*", ya que no intenta construir un modelo, sino que almacena instancias de los datos de entrenamiento. La clasificación se calcula a partir de una mayoría simple de votos de los k vecinos más cercanos de cada punto. Es un algoritmo que funciona muy bien con datos que siguen una distribución espacial bien definida.

Ventajas	Desventajas
Simple de implementar. Robustez frente a ruido. Efectividad con datos de entrenamiento son grandes.	Requiere determinar k. Coste computacional alto.

6.5. Árboles de decisión

Produce una secuencia de reglas que se pueden usar para clasificar los datos dado un conjunto de datos con sus clases. Se puede visualizar correctamente las decisiones tomadas mediante una gráfica del árbol y las diferentes ramificaciones que este posee.

Ventajas	Desventajas
Fácil de entender y visualizar. Requiere poca preparación de datos. Puede manejar tanto datos numéricos como categóricos.	Decidir grado de profundidad. Sensibles al ruido.

6.6. Random Forest

Ajusta una serie de árboles de decisión en varias submuestras de conjuntos de datos y utiliza el promedio para mejorar la precisión predictiva del modelo y controla el ajuste excesivo. El tamaño de la submuestra es siempre el mismo que el tamaño de la muestra de entrada original, pero las muestras se extraen con reemplazo.

Ventajas	Desventajas
Más avanzado que árboles de decisión en muchos casos. Reduce el sobreajuste.	Predicción lenta. Difícil de implementar. Alta complejidad. Gran gasto computacional.

6.7. SVM

Es una representación de los datos de entrenamiento como puntos en el espacio separados en categorías por una línea clara que es lo más amplia posible. Los nuevos ejemplos se mapean en ese mismo espacio y se predice que pertenecen a una categoría en función de qué lado de la línea se encuentran.

Ventajas	Desventajas
Eficaz en espacios de alta dimensión. Eficiente en la memoria. Utiliza un subconjunto de puntos de entrenamiento en la función de decisión.	No proporciona estimaciones de probabilidad se calculan mediante una validación cruzada de cinco veces.

6.8. ¿Qué algoritmo escoger?

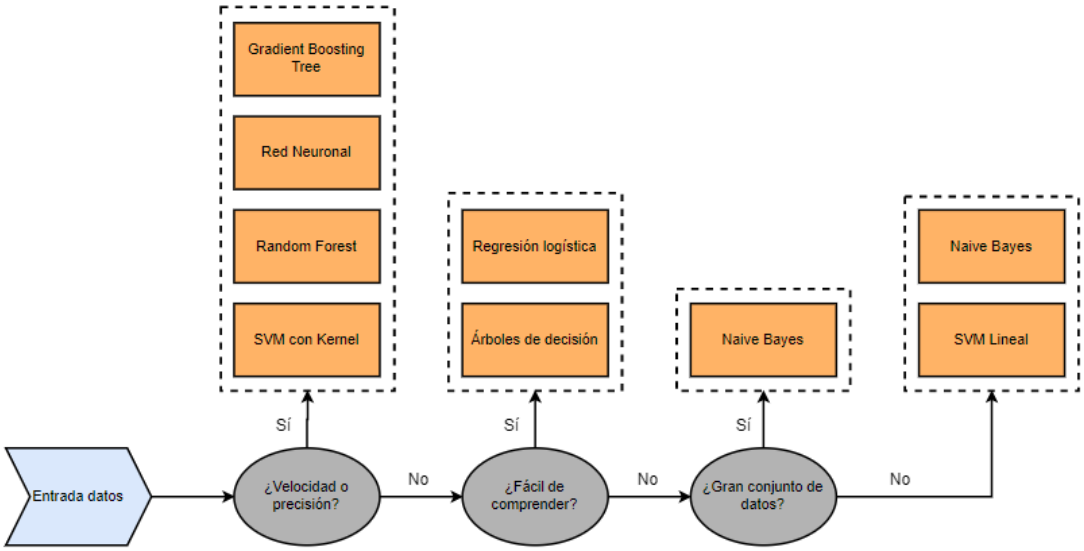


Imagen 10: Espacio original.

7. Conclusiones

Las Máquinas de Vectores de Soporte (SVM) permiten encontrar la forma óptima de clasificar entre varias clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases. Los vectores que definen el borde de esta separación son los vectores de soporte. En el caso de que las clases no sean linealmente separables, podemos usar el truco del kernel para añadir una dimensión nueva donde sí lo sean.

SVM ofrece garantías cuando el conjunto de datos no es excesivamente grande, hay un margen de separación entre las clases y no hay mucho ruido o solapamiento de las clases. En este trabajo se han expuesto algunas de las aplicaciones que esta técnica ha tenido como es dar una herramienta para el diagnóstico de Parkinson y Temblor Esencial, reconocimiento de locutor en habla conversacional, reconocimiento facial y recuperación de la información.

El escenario ideal para un problema en el que se aplique SVM es aquel que se busque una herramienta potente y en el que prime la velocidad de ejecución y la precisión. Para ello se puede utilizar un kernel no lineal como se ha explicado a lo largo del trabajo, mientras que en problemas sencillos SVM se desenvuelve realmente bien obteniendo buenos resultados con un kernel lineal en problemas con un conjunto de datos que no es excesivamente grande.

8. Bibliografía

[1] Roberto González & Antonio Barrientos & Marcelo Toapanta & Jaime del Cerro. *Aplicación de las Máquinas de Soporte Vectorial (SVM) al diagnóstico clínico de la Enfermedad de Párkinson y el Temblor Esencial. Revista Iberoamericana de Automática e Informática Industrial.* URL: https://www.sciencedirect.com/science/article/pii/S1697791217300468.

[2] Rohit Garg. *7 Types of Classification Algorithms.* URL: https://analyticsindiamag.com/7-types-classification-algorithms/.

[3] P. López. *SVM versus a monkey. Make your bets.* URL: https://quantdare.com/svm-versus-a-monkey/.

[4] Universidad de Sevilla. *Detección Multiusuario para DS-CDMA basado en SVM.* URL: http://bibing.us.es/proyectos/abreproy/11185/fichero/Volumen+1_Detector+Multiusuario+para+DS-CDMA+basado+en+SVM%252F10.+Otra+Aplicaciones+de+SVM%252Fotras+aplicaciones+SVM.pdf.

Anexo: Ejemplos de implementación en R

Pablo Alfaro Goicoechea, Carlos Morales Aguilera, Carlos S. Sánchez Muñoz

16/12/2020

Primer ejemplo: Comparativa modelos en Iris

El primer ejemplo consiste en evaluar como se comporta **SVM** en un ejemplo sencillo como es el conjunto de datos *iris*, para ello se comparará con diversos modelos como *árboles de decisión*, *random forest* o *Knn*. Es un ejemplo interesante ya que sin demasiado esfuerzo ni demasiado ajuste se puede comprobar la eficacia de esta herramienta en un problema de clasificación sencillo con tres clases.

Lo primero es hacer un análisis exploratorio de los datos:

```
data(iris)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

A continuación se lleva a cabo una normalización de los datos, excepto la variable clasificatoria:

```
scaled = as.data.frame(scale(iris[,1:4]))
scaled$Species = iris$Species
summary(scaled)
```

```
##   Sepal.Length       Sepal.Width       Petal.Length      Petal.Width
##  Min.   :-1.86378   Min.   :-2.4258   Min.   :-1.5623   Min.   :-1.4422
##  1st Qu.:-0.89767   1st Qu.:-0.5904   1st Qu.:-1.2225   1st Qu.:-1.1799
##  Median :-0.05233   Median :-0.1315   Median : 0.3354   Median : 0.1321
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.67225   3rd Qu.: 0.5567   3rd Qu.: 0.7602   3rd Qu.: 0.7880
##  Max.   : 2.48370   Max.   : 3.0805   Max.   : 1.7799   Max.   : 1.7064
```

```
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Realizamos una visualización inicial de la distribución de los datos, observando como se relacionan entre sí sus variables.

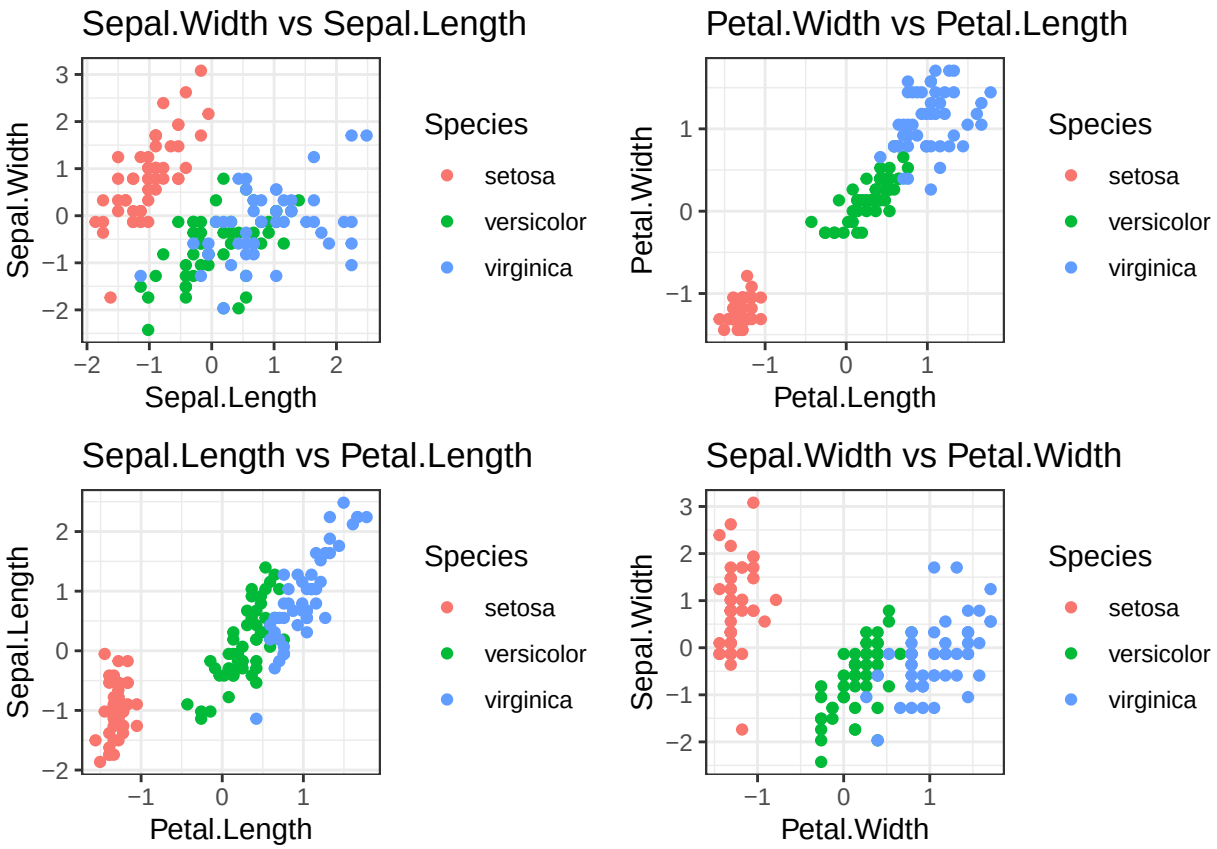
```
# Relationship between Sepal characteristics
plot1 = ggplot(scaled,aes(x =Sepal.Length,y = Sepal.Width,color = Species)) +
  geom_point() + ggtitle("Sepal.Width vs Sepal.Length") +
  theme_bw()

# Relationship between Petal characteristics
plot2 = ggplot(scaled,aes(x =Petal.Length,y = Petal.Width,color = Species)) +
  geom_point() + ggtitle("Petal.Width vs Petal.Length") +
  theme_bw()

# Relationship between Sepal and Petal Length
plot3 = ggplot(scaled,aes(x =Petal.Length,y = Sepal.Length,color = Species)) +
  geom_point() + ggtitle("Sepal.Length vs Petal.Length") +
  theme_bw()

# Relationship between Sepal and Petal width
plot4 = ggplot(scaled,aes(x =Petal.Width,y = Sepal.Width,color = Species)) +
  geom_point()  + ggtitle("Sepal.Width vs Petal.Width") +
  theme_bw()

grid.arrange(plot1,plot2,plot3,plot4,nrow = 2)
```



Creamos conjuntos de *train* y *tests* haciendo un reparto del 75%-25% respectivamente:

```
# Sample size
sample_size =  round(0.75*nrow(scaled))
# Random training index
train_ind = sample(seq_len(nrow(scaled)), size = sample_size)
# Create subsets
train = scaled[train_ind, ]
test = scaled[-train_ind, ]
```

Modelo de árboles de decisión con *Rpart*:

```
# Decision tree with Rpart
model.rpart = rpart(Species ~ . ,data =train)
# Predict
pred.rpart = predict(model.rpart,newdata = test,type = "class")
# Create the confusion matrix
matrix<-confusionMatrix(test$Species, pred.rpart)
# Save accuracy
rpart_accuracy <- matrix$overall["Accuracy"]
# Print table
kable(matrix$table)
```

	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	17	0
virginica	0	1	8

Modelo de Knn con *Knn*:

```
# Save class
cl = train$Species
# Predict
pred.knn = knn(train[,1:4],test[,1:4],cl,k=3)
# Create the confusion matrix
matrix<-confusionMatrix(test$Species, pred.knn)
# Save accuracy
knn_accuracy <- matrix$overall["Accuracy"]
# Print table
kable(matrix$table)
```

	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	16	1
virginica	0	1	8

Modelo de Random Forest con *randomForest*:

```
# Random Forest with randomForest
model.rf = randomForest(Species ~ .,data = train)
# Predict
pred.rf = predict(model.rf,newdata = test)
# Create the confusion matrix
matrix<-confusionMatrix(test$Species, pred.rf)
# Save accuracy
rf_accuracy <- matrix$overall["Accuracy"]
# Print table
kable(matrix$table)
```

	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	16	1
virginica	0	0	9

Modelo de SVM con *SVM*:

```
# SVM with svm
model.svm = svm(Species ~ . ,data = train, method="C-classification", kernel="radial",
          gamma=0.1, cost=10)
# Predict
pred.svm = predict(model.svm,newdata = test)
```

```r
# Create the confusion matrix
matrix<-confusionMatrix(test$Species, pred.svm)
# Save accuracy
svm_accuracy <- matrix$overall["Accuracy"]
# Print table
kable(matrix$table)
```

	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	16	1
virginica	0	0	9

A continuación, podemos observar los resultados obtenidos con los diferentes modelos:

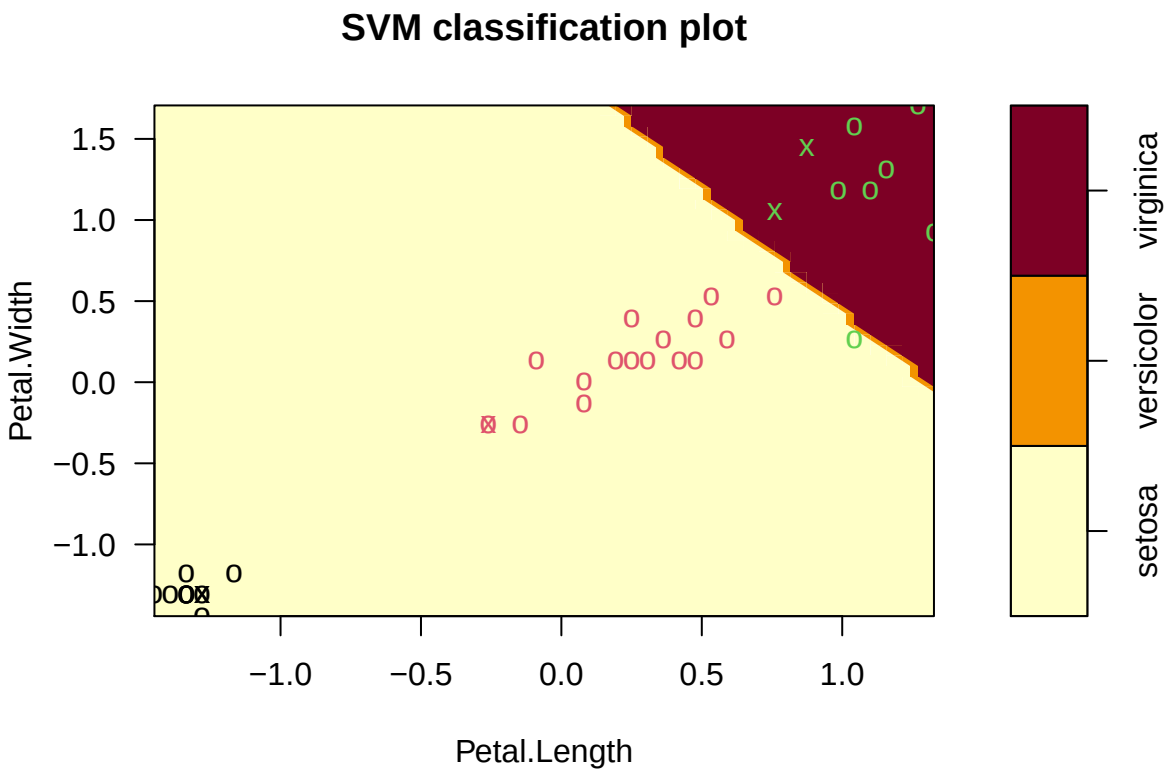
```r
models = data.frame(Technique = c("Decision Tree","kNN","SVM","Random Forest"),Accuracy_Percentage = c(
models
```

```
##       Technique Accuracy_Percentage
## 1 Decision Tree           0.9736842
## 2           kNN           0.9473684
## 3           SVM           0.9736842
## 4 Random Forest           0.9736842
```

Como se puede observar, se obtienen los mismos resultados en este sencillo ejemplo, por lo que se observar que en ejemplos linealmente separables *SVM* funciona al mismo nivel que algunos de los algoritmos más conocidos. La elección de parámetros, kernel, y otros elementos no es una tarea trivial, ya que debe adaptarse al problema en particular para obtener unos buenos resultados.

A continuación se demuestra como se verían los resultados en una gráfica sencilla:

```r
plot(model.svm, test, Petal.Width ~ Petal.Length,
          slice=list(Sepal.Width=3, Sepal.Length=4))
```



Al tratarse de un plano 2D, no se puede apreciar del todo como se realizan las divisiones de forma gráfica, aunque si se puede apreciar que los puntos verdes corresponden a la clase *virginica*, los rojos a la clase *versicolor* y los negros a la *setosa*, y la división es correcta. También se puede observar que el punto verde que se encuentra en la sección correspondiente a *versicolor*, es el único error del conjunto de test.

Segundo ejemplo: Observaciones con función no lineal en 2 dimensiones

Para el siguiente ejemplo, se utiliza el conjunto de datos publicado en el libro *Elements of Satistical Learning*, que contiene observaciones simuladas (2 predictores) con funciones no lineales en un espacio bidimensional.

```
# Obtain data
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
```

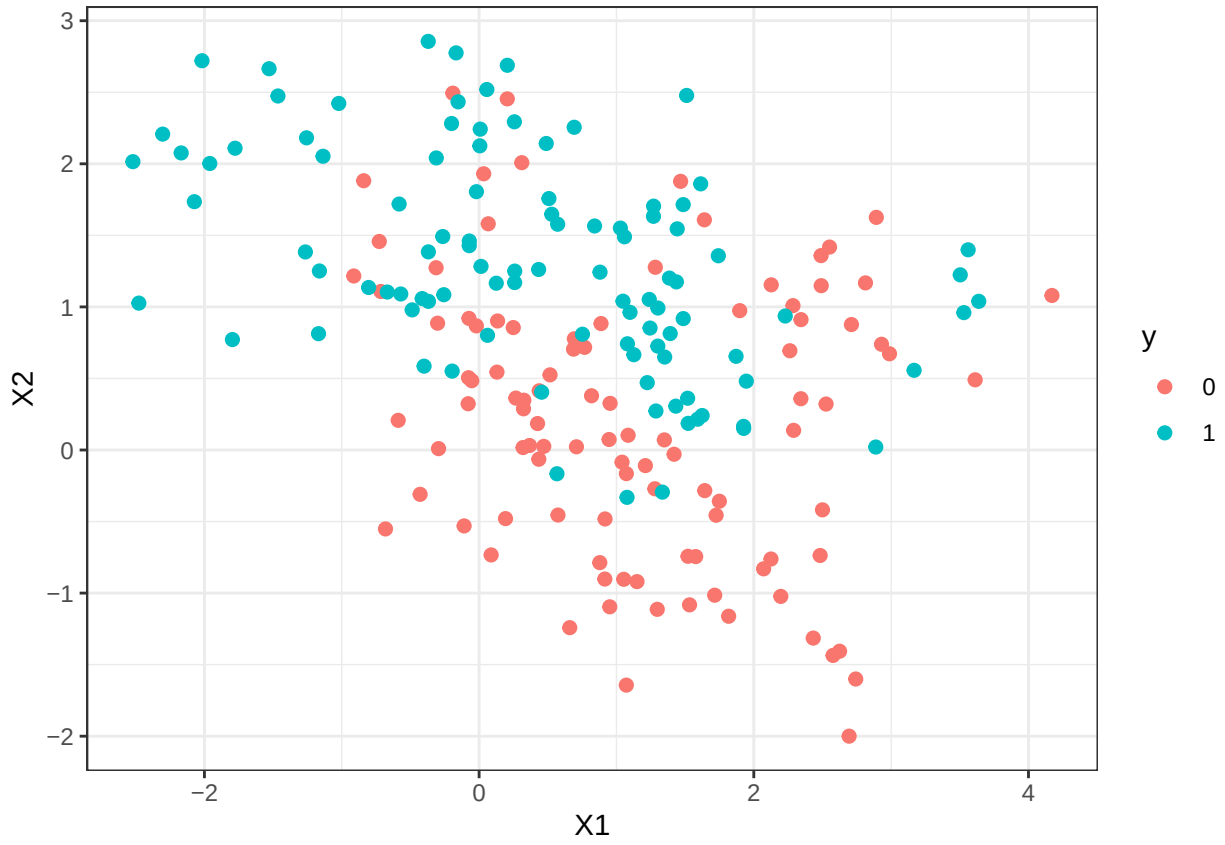
El objeto *ESL.mixture* es una lista que contiene almacenados el valor de los dos predictores en el elemento x y el valor de la clase a la que pertenece cada observación en el elemento y.

```
# Create dataframe
data <- data.frame(ESL.mixture$x, y = ESL.mixture$y)
# Transform classification variable to factor
data$y <- as.factor(data$y)
# Print summary
summary(data)
```

```
##        X1                  X2                y
##  Min.   :-2.52082   Min.   :-1.99985   0:100
##  1st Qu.:-0.07147   1st Qu.: 0.09555   1:100
##  Median : 0.85970   Median : 0.86139
##  Mean   : 0.78467   Mean   : 0.75602
##  3rd Qu.: 1.54344   3rd Qu.: 1.43527
##  Max.   : 4.17075   Max.   : 2.85581
```

A continuación vemos como se distribuyen los datos en un espacio bidimensional:

```
# Plot data
ggplot(data = data, aes(x = X1, y = X2, color = y)) +
  geom_point(size =2) +
  theme_bw()
```



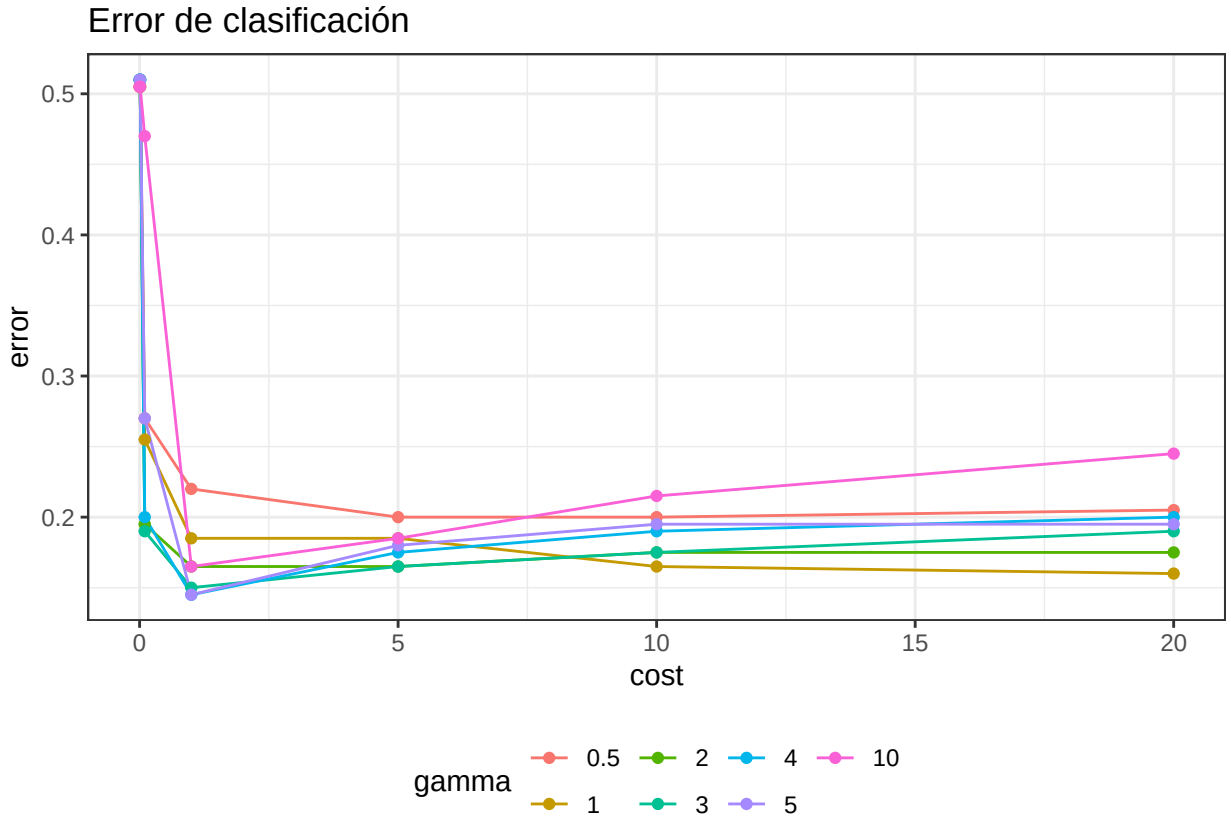
Como no conocemos la mejor configuración posible para el problema, vamos a utilizar la función tune que nos permite probar distintos hiperparámetros, en nuestro caso, con el modelo de *SVM* para encontrar la mejor configuración posible.

El kernel escogido es de tipo *radial*, ya que se trata de un kernel no lineal que nos permite un mayor ajuste que un kernel *polinomial*. En este tipo de kernel está la tarea de determminar el valor γ apropiado y un coste de penalización. Para ello visualizaremos los diferentes errores de clasificación obtenidos y seleccionaremos el mejor modelo obtenido.

```
# Train model with tune, using different hyperparameters
tune_svm <- tune("svm", y ~ X1 + X2, data = data, kernel = 'radial',
```

```
                ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 20),
                              gamma = c(0.5, 1, 2, 3, 4, 5, 10)))

# Plot different performances
ggplot(data = tune_svm$performances, aes(x = cost, y = error, color = as.factor(gamma)))+
  geom_line() +
  geom_point() +
  labs(title = "Error de clasificación", color = "gamma") +
  theme_bw() +
  theme(legend.position = "bottom")
```



A continuación se observan los mejores parámetros y guardamos el mejor modelo de *SVM* construido para el problema:

```
# Print best parameters for problem with SVM
tune_svm$best.parameters
```

```
##    cost gamma
## 32    1     4
```

```
# Save best model
model.svm <- tune_svm$best.model
```

Ya solo quedaría comprobar el resultado obtenido del modelo, para ello se realizará la predicción con el mejor modelo obtenido, se visualizará su matriz de confusión y se visualizará el modelo mediante una gráfica 2D.

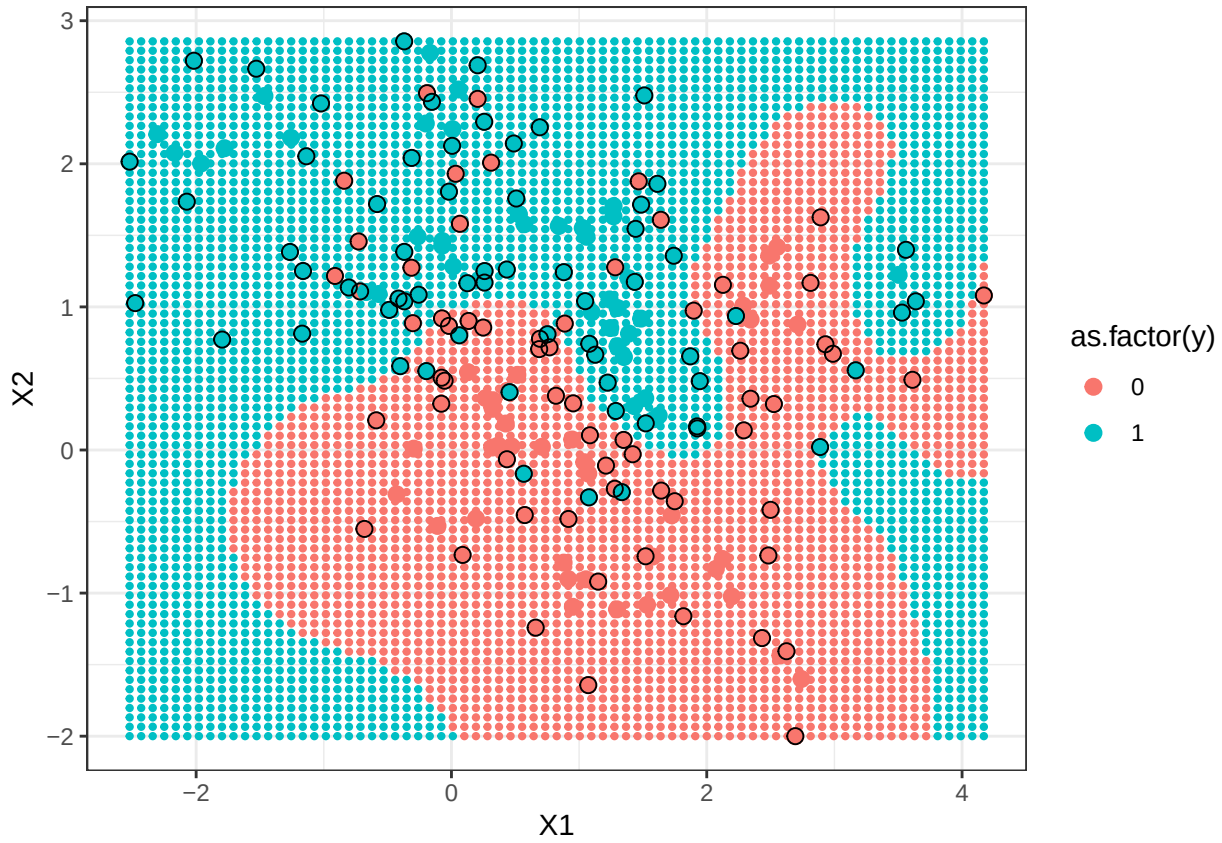
```r
# Obtain ranges
range_X1 <- range(data$X1)
range_X2 <- range(data$X2)

# Interpolate new points for tests
new_x1 <- seq(from = range_X1[1], to = range_X1[2], length = 75)
new_x2 <- seq(from = range_X2[1], to = range_X2[2], length = 75)
new_points <- expand.grid(X1 = new_x1, X2 = new_x2)

# Predict
predictions <- predict(object = model.svm, newdata = new_points)

# Get region color
regions <- data.frame(new_points, y = predictions)

ggplot() +
  # Plot regions
  geom_point(data = regions, aes(x = X1, y = X2, color = as.factor(y)),
             size = 0.8) +
  # Plot points
  geom_point(data = data, aes(x = X1, y = X2, color = as.factor(y)),
             size = 2.5) +
  # Plot points that acts ass vector support
  geom_point(data = data[model.svm$index, ],
             aes(x = X1, y = X2, color = as.factor(y)),
             shape = 21, colour = "black",
             size = 2.5) +
             theme_bw()
```


3
2
1
0
−1
−2
X2
−2
0
2
4
X1
as.factor(y)
0
1