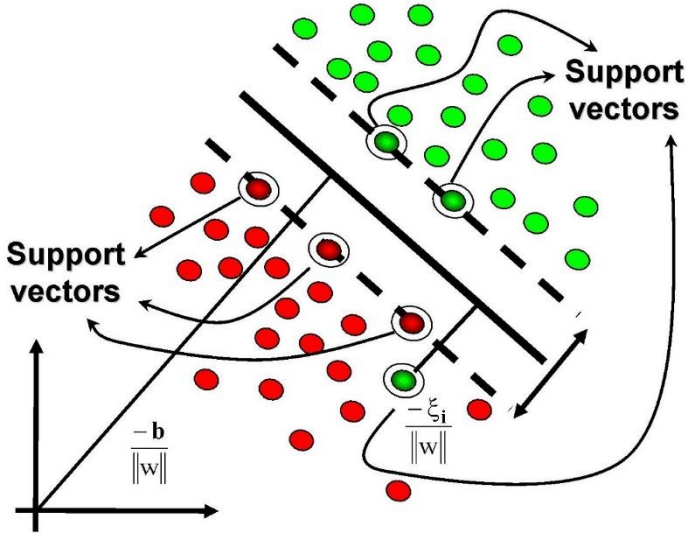
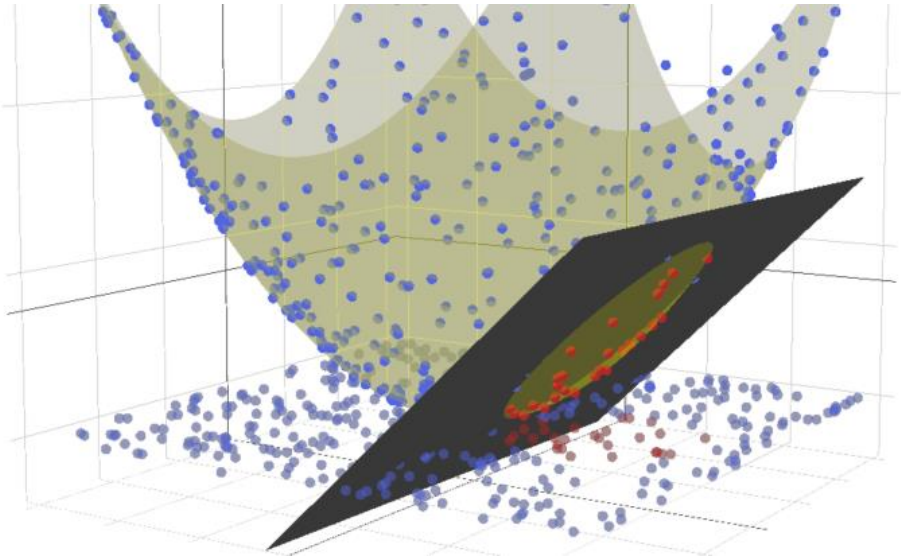
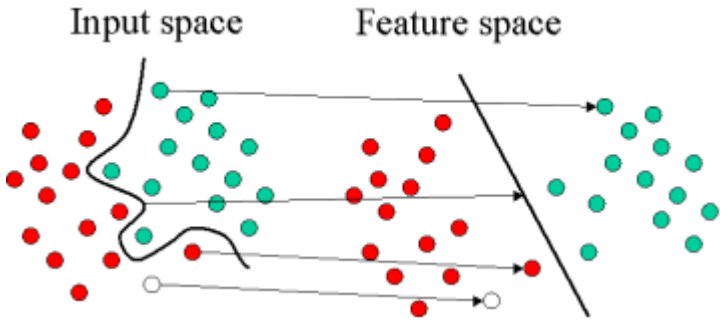
Máquinas de Vector Soporte (Support Vector Machines, SVMs)







Oscar Centeno Mora

Preámbulo

La presente técnica es una de las astucias más importantes que se han creado hasta la fecha. Utiliza una aplicación de la álgebra y una optimización computacional para poder alcanzar el óptimo en la representación de un conjunto de datos.

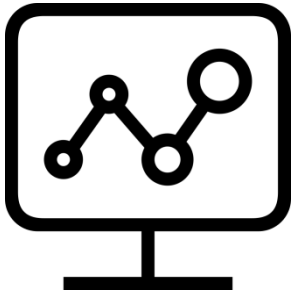
Por el nombre ***Máquinas de Vector Soporte*** pensamos en algo casi que galáctico, la técnica busca mejorar la captación de la adecuación de los datos en la búsqueda de la mejor representación posible de estos.

Auque sus orígenes fueron en la clasificación, es ampliamente usada para la predicción.

Preámbulo

Aunque a nivel práctico o de desarrollo computacional no encontré una verdadera aplicación para series de tiempo, sin duda no tardará en llegar: se posee todo el sustento teórico para poder hacerlo:

http://www-ai.cs.uni-dortmund.de/EVENTS/FGML2001/FGML2001-Paper-Rueping.pdf
http://www.svms.org/regression/MSRS00.pdf
http://eprints.whiterose.ac.uk/83992/1/acse%20research%20report%20780.pdf
https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-33052010000100008
https://www.eecis.udel.edu/~shatkay/Course/papers/USVMHeartBeatClassifier.pdf
https://people.eng.unimelb.edu.au/baileyj/papers/kdd685-ristanoski.pdf
https://pdfs.semanticscholar.org/a244/c47a1d4a8c2894b22807df8c7eec16cc110a.pdf
ftp://ftp.esat.kuleuven.be/sista/falck/estsp08.pdf



Esperemos que dentro de poco R o Python realicen la respectiva implementación.

Preámbulo

Antes de empezar, ¿qué creemos que entendemos por ***Máquinas de Vector Soporte*** ?

Índice

Índice

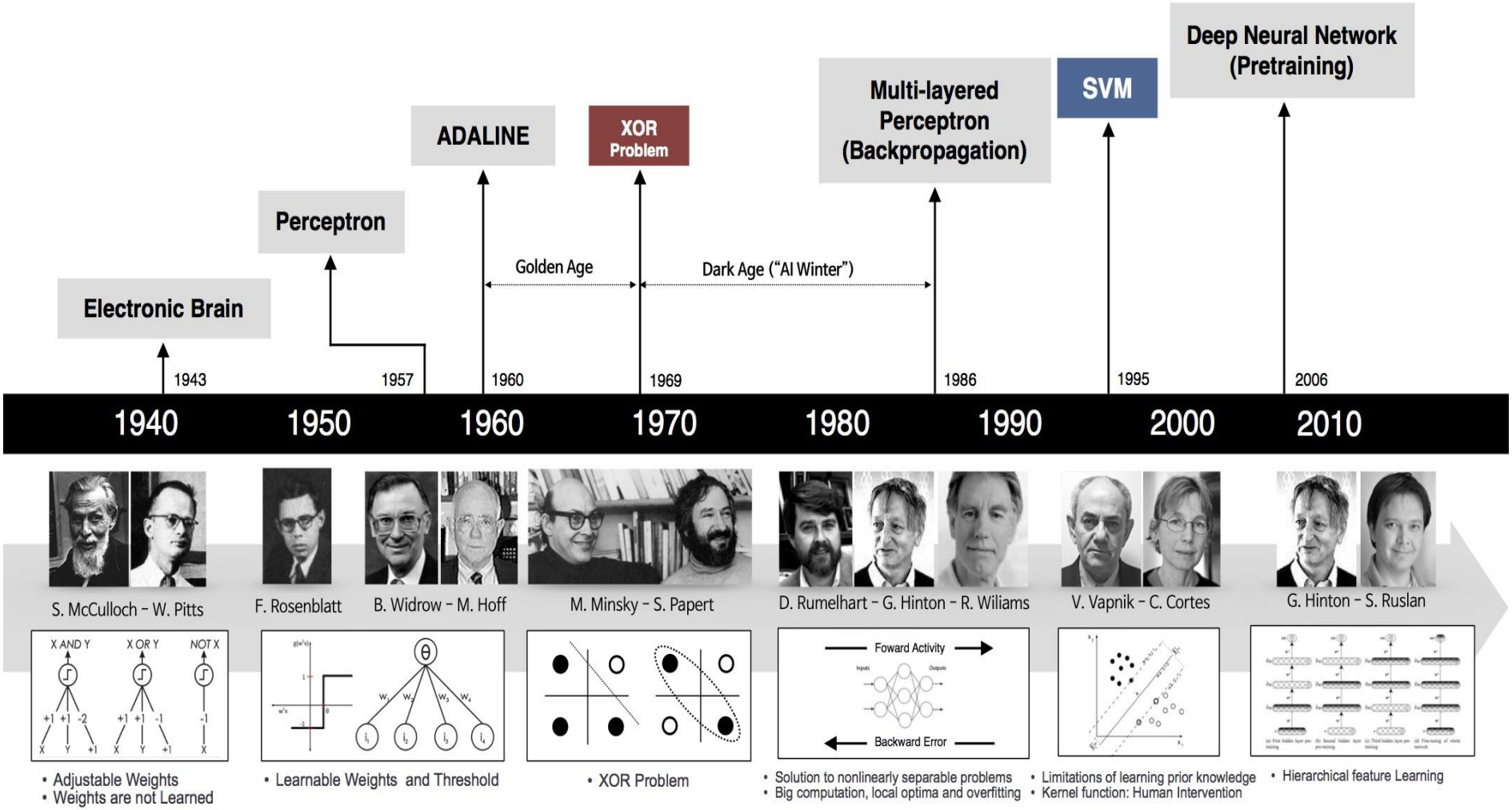
Introducción

El método de clasificación-regresión Máquinas de Vector Soporte (*Vector Support Machines, SVMs*) fue desarrollado en la década de los 90, dentro de campo de la ciencia computacional.

Si bien originariamente se desarrolló como un método de clasificación binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión.

El *SVMs* ha resultado ser uno de los mejores clasificadores para un amplio abanico de eventos, por lo que se considera uno de los referentes dentro del ámbito de aprendizaje estadístico y el aprendizaje supervisado (machine learning).

Introducción

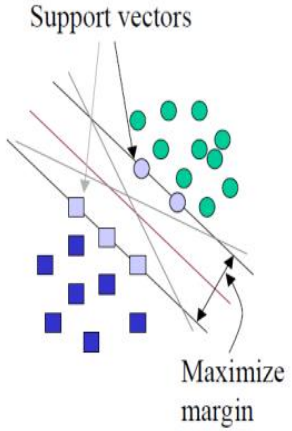


Introducción

Las Máquinas de Vector Soporte se fundamentan en el ***Maximal Margin Classifier***, que a su vez, se basa en el concepto de hiperplano. Pronto veremos estos conceptos.

Comprender los fundamentos de las *SVMs* requiere de conocimientos sólidos en álgebra lineal… ¿cómo estamos en este punto?

Una descripción detallada en el libro *Support Vector Machines Succinctly by Alexandre Kowalczyk,* proporciona todas los desarrollos matemáticos y algebráicos:



http://jermmy.xyz/images/2017-12-23/support_vector_machines_succinctly.pdf

(desarrollo en Python…)

Introducción

- En R y R Studio, las librería e1071 y Liblinear R contienen los algoritmos necesarios para obtener modelos de clasificación simple, múltiple y regresión, basados en *Support Vector Machines*.

Índice

Hiperplanos: en búsqueda del MMC

- En un espacio p-dimensional, un hiperplano se define como un subespacio plano y afín de dimensiones $p - 1$. El término afín significa que el subespacio no tiene por qué pasar por el origen.
-
- En un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, es decir, una recta. En un espacio tridimensional, un hiperplano es un subespacio de dos dimensiones, un plano convencional. Para dimensiones $p > 3$ no es intuitive visualizer un hiperplano, pero el concepto de subespacio $p - 1$ dimensiones se mantiene.

- La definición matemática de un hiperplano es bastante simple. En el caso de dos dimensiones, el hiperplano se describe acorde a la ecuación de una recta:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

Hiperplanos: en búsqueda del MMC

Dados los parámetros β_0, β_1 y β_2, todos los pares de valores $x = (x_1, x2)$ para los que se cumple la igualdad, son puntos del hiperplano.

Lo anterior se puede generalizar para p variables. Sea la siguiente ecuación para p-dimensiones:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

de igual manera, todos los puntos definidos por el vector $x = (x_1, x_2, \ldots, x_p)$ que cumple la ecuación, pertenecen al hiperplano.

¿Qué pasa si un determinado x no cumple con la igualdad nula?

Hiperplanos: en búsqueda del MMC

Cuando x no satisface la ecuación:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p < 0$$
$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p > 0$$

el punto $\boldsymbol{x}$ cae a un lado o al otro del hiperplano. De esta forma, se puede entender que un hiperplano divide un espacio p-dimensional en dos mitades. Para saber en qué lado del hiperplano se encuentra un determinado punto $\boldsymbol{x}$, solo hay que calcular el signo de la ecuación.

Ejemplifiquemos lo anterior con la siguiente ecuación:

$$1 + 2x_1 + 3x_2 = 0$$

Hiperplanos: en búsqueda del MMC

La siguiente imagen muestra el hiperplano de un espacio bidimensional. La ecucación que describe el hiperplano, una recta, es $1 + 2x_1 + 3x_2 = 0$. La region azul representa el espacio en el que se encuentran todos los puntos para los que $1 + 2x_1 + 3x_2 > 0$ y la region roja para los que $1 + 2x_1 + 3x_2 < 0$.

Hiperplanos: en búsqueda del MMC

- Veamos el caso de una clasificación binar. Cuando se dispone de n observaciones, cada una con p predictores cuya variable respuesta tiene dos niveles (1 y -1), se pueden emplear hiperplanos para construir un clasificador que permita predecir a que grupo pertenece una observación en función de sus predictores. Este mismo problema puede abordarse también con otros métodos (regresión logística, LDA, árboles de clasificación…) cada uno con ventajas y desventajas.

- Para facilitar la comprensión, las siguientes explicaciones se basan en un espacio de dos dimensiones, donde un hiperplano es una recta. Sin embargo, los mismos conceptos son aplicables a dimensiones superiores.

Hiperplanos: en búsqueda del MMC

Si la distribución de las observaciones es tal que se pueden separar linealmente de forma perfecta en las dos clases (+1 y -1), entonces un hiperplano de separación cumple que:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p < 0, si\ y_i = 1$$
$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p > 0, si\ y_i = -1$$

Al identificar cada clase con +1 o -1, dado que multiplicar dos valores negativos resultan en un valor positive, las dos condiciones anteriores pueden simplificarse en una única:

$$y_i(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p) > 0, para\ i = 1, \ldots, n$$

Hiperplanos: en búsqueda del MMC

Bajo este escenario, el clasificador más sencillo consiste en asignar cada observación a una clase dependiendo del lado del hiperplano en el que se encuentre. Es decir, la observación $\boldsymbol{x}^*$ se clasifica acorde al signo de la función:

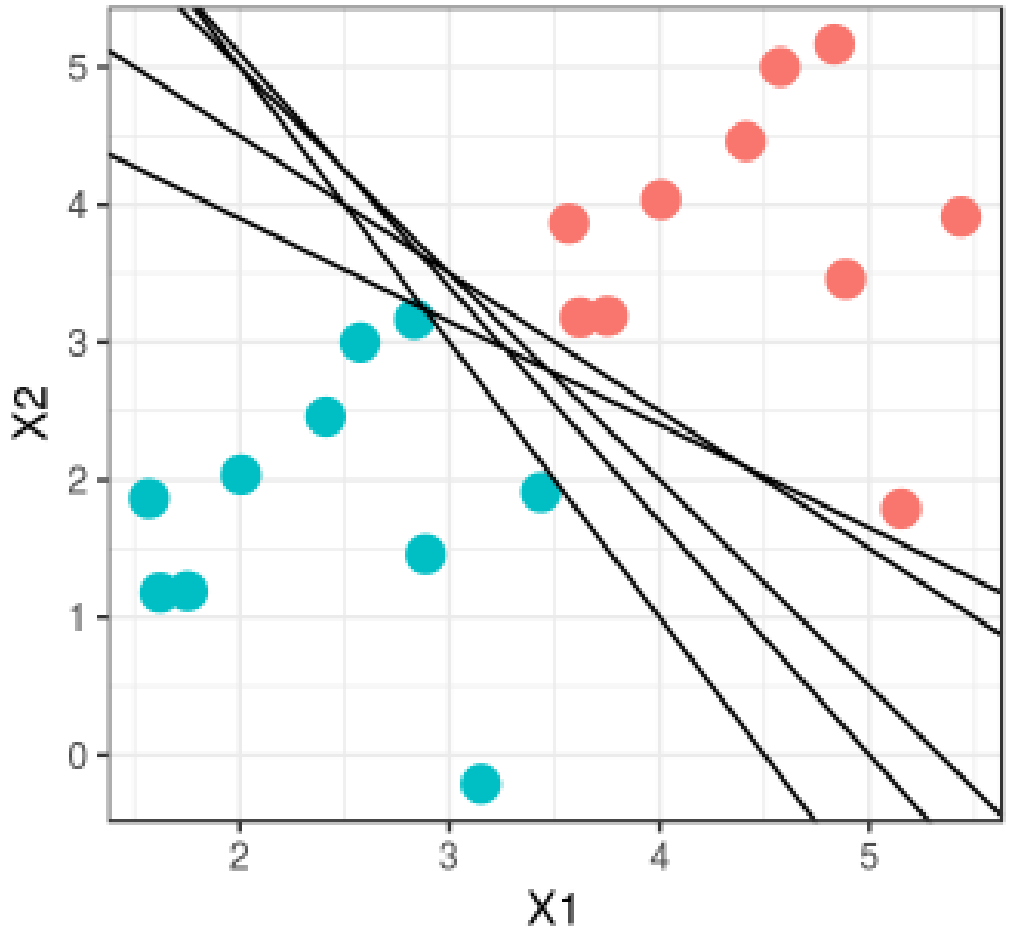
$$f(\boldsymbol{x}^*) = \beta_0 + \beta_1 {x_1}^* + \beta_2 {x_2}^* + \cdots + \beta_p {x_p}^*$$

Si $f(\boldsymbol{x}^*)$ es positiva, la observación se asigna a la clase 1, si es negativa, a la clase −1. Además, la magnitud de $f(\boldsymbol{x}^*)$ permite saber cómo de lejos está la observación del hiperplano y con ello la confianza de la clasificación.

La definición del hiperplano para casos perfectamente separables linealmente resulta en un número infinito de posibles hiperplanos, lo que hace necesario un método que permita seleccionar uno de ellos como clasificador óptimo: ***Máximal Margin Classifier***.

Hiperplanos: en búsqueda del MMC

- Veamos el caso anterior: ¿cuántos posibles hiperplanos o rectas podrían existir?



Hiperplanos: en búsqueda del MMC

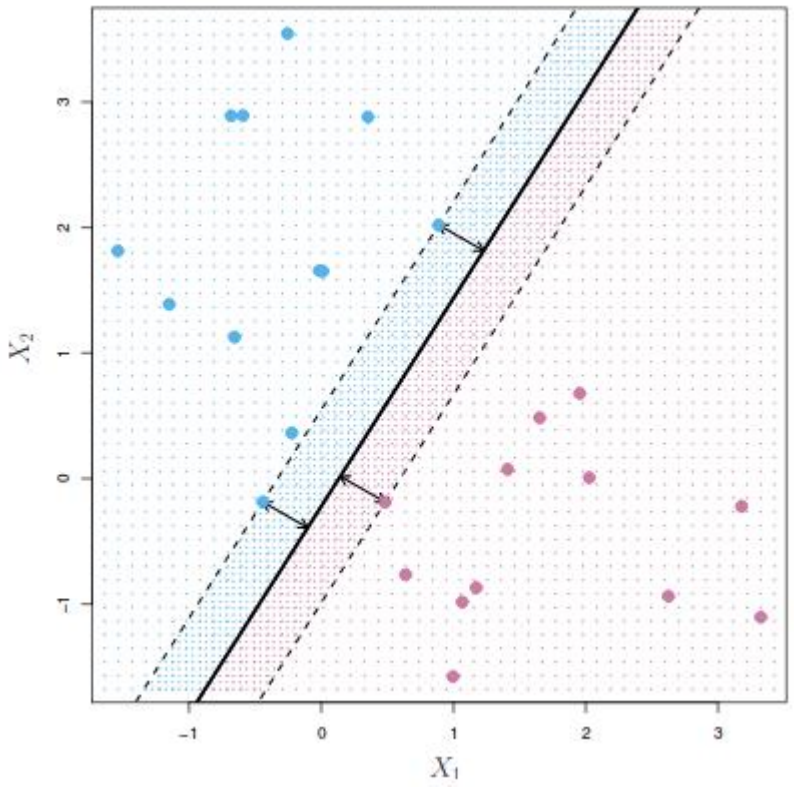
La solución a este problema consiste en seleccionar como clasificador óptimo al que se conoce como ***maximal margin hyperplane*** o ***hiperplano óptimo de separación***, que se corresponde con el hiperplano que se encuentra más alejado de todas las observaciones de entrenamiento.

Para obtenerlo, se tiene que calcular la distancia perpendicular de cada observación a un determinado hiperplano. La menor de estas distancias (conocida como margen) determina como de alejado está el hiperplano de las observaciones de entrenamiento. El ***maximal margin hyperplane*** se define como el hiperplano que consigue un mayor margen, es decir, que la distancia mínima entre el hiperplano y las observaciones es lo más grande posible.

Aunque esta idea suena razonable, no es posible aplicarla, ya que habría infinitos hiperplanos contra los que medir las distancias. En su lugar, se recurre a métodos de optimización.

Hiperplanos: en búsqueda del MMC

¿Por qué decimos que hay entonces infinitos hiperplanos?
¿Y por qué entonces recurrimosa optimización numérica?



Hiperplanos: en búsqueda del MMC

La imagen anterior muestra el ***maximal margin hyperplane*** para un conjunto de datos de entrenamiento.

Las tres observaciones equidistantes respecto al ***maximal margin hyperplane*** se encuentran a lo largo de las líneas discontinuas que indican la anchura del margen. A estas observaciones se les conoce como **vectores soporte**, ya que son vectores en un espacio *p*-dimensional y soportan (definen) el ***maximal margin hyperplane***.

Cualquier modificación en estas observaciones (vectores soporte) conlleva cambios en el ***maximal margin hyperplane***. Sin embargo, modificaciones en observaciones que no son vector soporte no tienen impacto alguno en el hiperplano. ¿Cuál es el verdadero riesgo de este método?

Hiperplanos: en búsqueda del MMC

Analicemos un caso más realista, en donde no poseemos una separación tan marcada entre los grupos.

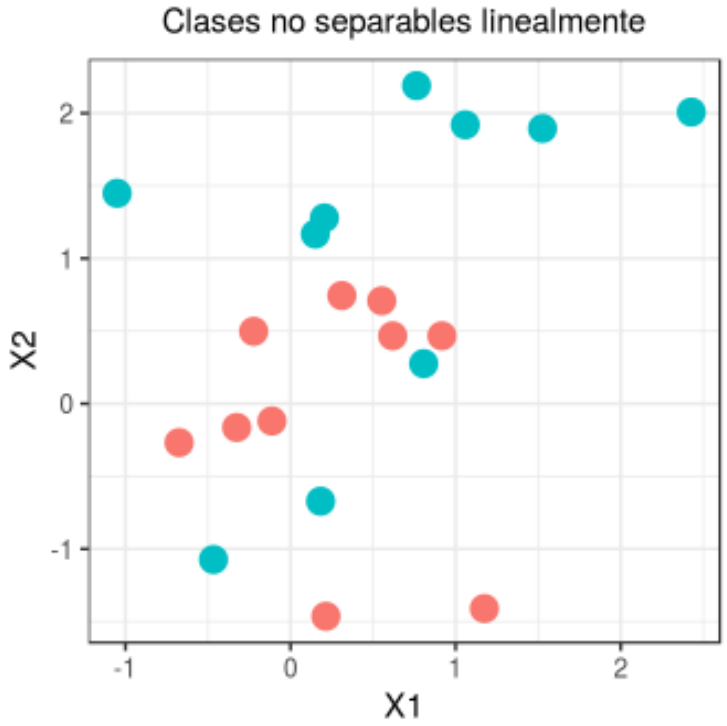
En la gran mayoría de casos reales, los datos no se pueden separar linealmente de forma perfecta, por lo que no existe un hiperplano de separación y no puede obtenerse un *maximal margin hyperplane* como el anterior.

El ***maximal margin hyperplane*** descrito en el apartado anterior es una forma muy simple y natural de *clasificación* siempre y cuando exista un hiperplano de separación.

Ante lo anterior, se debe recurrir a otro tipo de estrategias para poder adecuar el clasificador.

Hiperplanos: en búsqueda del MMC

Para solucionar estas situaciones, se puede extender el concepto de *maximal margin hyperplane*para obtener un hiperplano que casi separe las clases, pero permitiendo que cometa unos pocos errores. A este tipo de hiperplano se le conoce como *Support Vector Classifier* o *Soft Margin*.



Índice

Support Vector Classifier o Soft Margin SVM

El *Maximal Margin Classifier* descrito en la sección anterior tiene poca aplicación práctica: rara vez se encuentran casos en los que las clases sean perfecta y linealmente separables. Incluso cumpliéndose estas condiciones ideales, en las que exista un hiperplano capaz de separar perfectamente las observaciones en dos clases, esta aproximación sigue presentando dos inconvenientes:

- El hiperplano que separar perfectamente las observaciones es muy sensible a variaciones en los datos. Incluir una nueva observación puede suponer cambios muy grandes en el hiperplano de separación (poca robustez)
- El *maximal margin hyperplane* se ajuste perfectamente a las observaciones de entrenamiento para separarlas todas correctamente suele conllevar problemas de sobre ajuste (*overfitting)*.

Support Vector Classifier o Soft Margin SVM

Por estas razones, es preferible crear un clasificador basado en un hiperplano que, aunque no separe perfectamente las dos clases, sea más robusto y tenga mayor capacidad predictiva al aplicarlo a nuevas observaciones (menos problemas de *overfitting*).

Esto es exactamente lo que consiguen los **clasificadores de vector soporte**, también conocidos como ***soft margin classifiers*** o ***Support Vector Classifiers***.

Para lograrlo, en lugar de buscar el margen de clasificación más ancho posible que consigue que las observaciones estén en el lado correcto del margen; se permite que ciertas observaciones estén en el lado incorrecto del margen o incluso del hiperplano.

Support Vector Classifier o Soft Margin SVM

La siguiente imagen muestra un clasificador de vector soporte ajustado a un pequeño set de observaciones

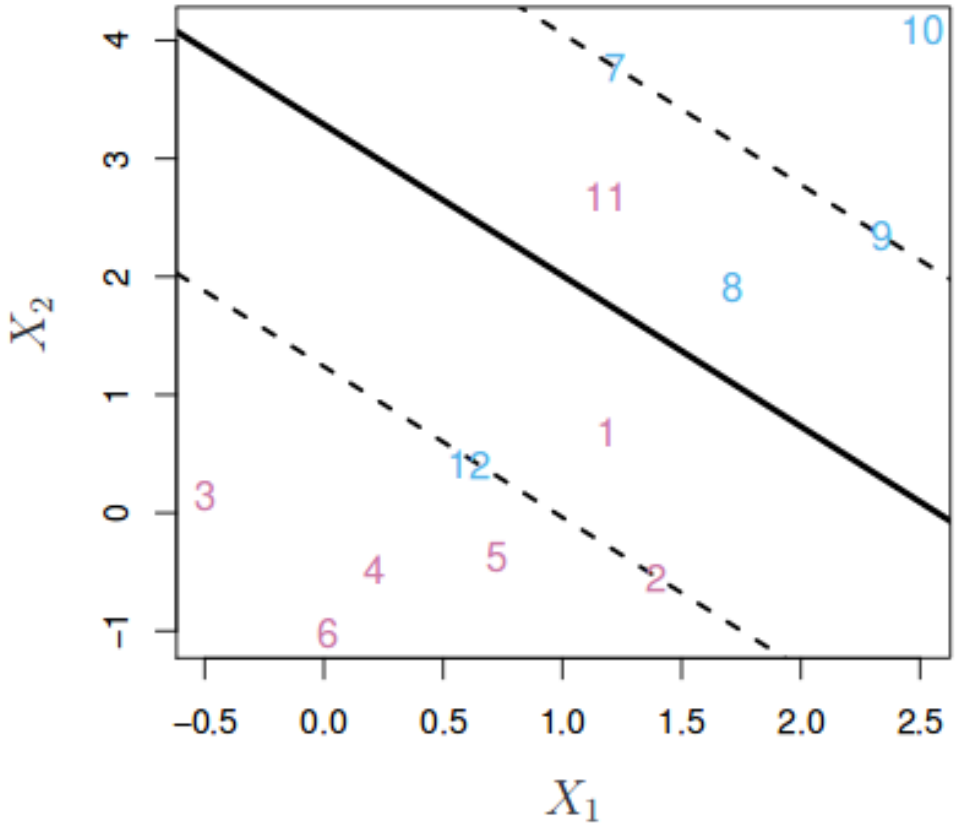


Imagen clasificador vector soporte obtenida del libro ISLR

Support Vector Classifier o Soft Margin SVM

La línea continua representa el hiperplano y las líneas discontinuas el margen a cada lado.

Las observaciones 2, 3, 4, 5, 6, 7 y 10 se encuentran en el lado correcto del margen (también del hiperplano) por lo que están bien clasificadas.

Las observaciones 1 y 8, a pesar de que se encuentran dentro del margen, están en el lado correcto del hiperplano, por lo que también están bien clasificadas. Las observaciones 11 y 12, se encuentran en el lado erróneo del hiperplano, su clasificación es incorrecta.

Todas aquellas observaciones que, estando dentro o fuera del margen, se encuentren en el lado incorrecto del hiperplano, se corresponden con observaciones de entrenamiento mal clasificadas.

Support Vector Classifier o Soft Margin SVM

La identificación del hiperplano de un clasificador de vector soporte, que clasifique correctamente la mayoría de las observaciones a excepción de unas pocas, es un problema de optimización convexa.

Es importante mencionar que el proceso incluye un hiperparámetro de *tuning (¿?)* $\boldsymbol{C}$: este controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste.

Si $\boldsymbol{C} = \infty$, no se permite ninguna violación del margen y por lo tanto, el resultado es equivalente al *Maximal Margin Classifier* (teniendo en cuenta que esta solución solo es posible si las clases son perfectamente separables).

Cuando más se aproxima $\boldsymbol{C}$ a cero, menos se penalizan los errores y más observaciones pueden estar en el lado incorrecto del margen o incluso del hiperplano. $\boldsymbol{C}$ es a fin de cuentas el hiperparámetro encargado de controlar el balance entre el sesgo y varianza del modelo.

Support Vector Classifier o Soft Margin SVM

El proceso de optimización tiene la peculiaridad de que solo las observaciones que se encuentran justo en el margen o que lo violan influyen sobre el hiperplano (determinan el margen).

A estas observaciones se les conoce como vectores soporte y son las que definen el clasificador obtenido. Esta es la razón por la que el parámetro $\boldsymbol{C}$ controla el balance entre *el sesgo* y varianza.

Cuando el valor de $\boldsymbol{C}$ es pequeño, el margen es más ancho, y más observaciones violan el margen, convirtiéndose en vectores soporte. El hiperplano está, por lo tanto, sustentado por más observaciones, lo que aumenta el *bias* pero reduce la varianza.

Cuando mayor es el valor de $\boldsymbol{C}$, menor el margen, menos observaciones serán vectores soporte y el clasificador resultante tendrá menor *bias* pero mayor varianza.

Support Vector Classifier o Soft Margin SVM

Otra propiedad importante que deriva de que el hiperplano dependa únicamente de una pequeña proporción de observaciones (vectores soporte), es su robustez frente a observaciones muy alejadas del hiperplano.

Esto hace al método de *clasificación vector soporte* distinto a otros métodos tales como *Linear Discrimiant Analysis (LDA)*, donde la regla de clasificación depende de la media de todas las observaciones.

¿Es correcto entonces suponer un método que no se base en todas las observaciones? Esta es una de las tantas peculiaridades de la presente técnica…

Support Vector Classifier o Soft Margin SVM

Mostremos el uso de los Support Vector Classifier en una clasificación binaria. Se emplea la función svm() :

```
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

type

- `C-classification`
- `nu-classification`
- `one-classification` (for novelty detection)
- `eps-regression`
- `nu-regression`

type

linear: $u'v$

polynomial: $(\gamma u'v + coef0)^{degree}$

radial basis: $e^{(} - \gamma|u - v|^2)$

sigmoid: $tanh(\gamma u'v + coef0)$

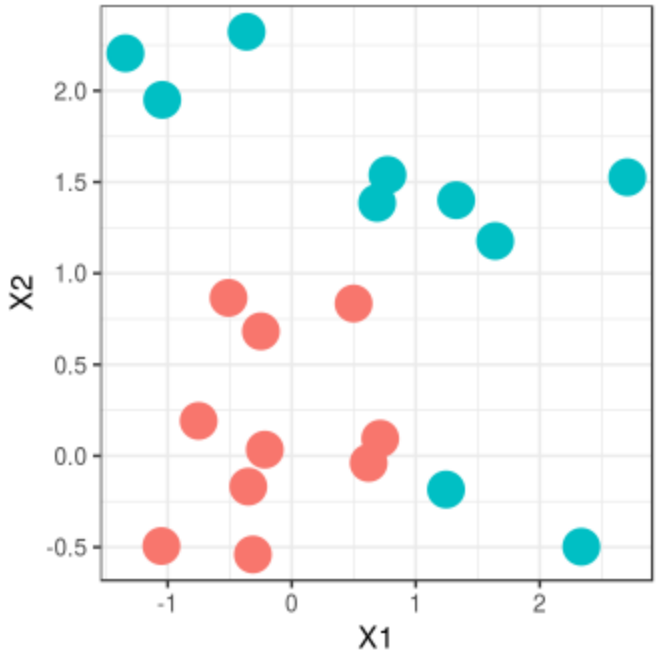
Support Vector Classifier o Soft Margin SVM

- Esta función ajusta *Support Vector Classifier* si se le especifica el argumento kernel="linear“, (como se describe más adelante, el método de *Support Vector Machines* es equivalente al *Support Vector Classifier*cuando el kernel utilizado es lineal).

- El argumento cost determina la penalización aplicada por violar el margen, es el nombre que emplea esta función para el hiperparámetro $\boldsymbol{C}$.

Support Vector Classifier o Soft Margin SVM

```r
set.seed(10111)
coordenadas <- matrix(rnorm(40), 20, 2)
colnames(coordenadas) <- c("X1", "X2")
y <- c(rep(-1, 10), rep(1, 10))
coordenadas[y == 1, ] <- coordenadas[y == 1, ] + 1
datos <- data.frame(coordenadas, y)
ggplot(data = datos, aes(x = X1, y = X2, color = as.factor(y))) + geom_point(size = 6) +
    theme_bw() + theme(legend.position = "none")
```



Support Vector Classifier o Soft Margin SVM

La representación gráfica de los datos muestra que los grupos **no** son linealmente separables. La función svm() identifica automáticamente si se trata de un problema de clasificación, la variable respuesta es de tipo factor, o de regresión, la variable respuesta es tipo numérico.

```
library(e1071)

# Se convierte la variable respuesta a factor
datos$y <- as.factor(datos$y)

# Para que la función svm() calcule el Support Vector Classifier, se tiene
# que indicar que la función kernel es lineal.
modelo_svm <- svm(formula = y ~ X1 + X2, data = datos, kernel = "linear", cost = 10,
    scale = FALSE)
```

En este caso, ambos predictores (X1, X2) tienen la misma escala por lo que no es necesario estandarizarlos. En aquellas situaciones en las que las escalas son distintas, sí hay que estandarizarlos, de lo contrario, los predictores de mayor magnitud eclipsarán a los de menor magnitud. El summary del modelo muestra que hay un total de 6 vectores soporte, 3 pertenecen a una clase y 3 a la otra.

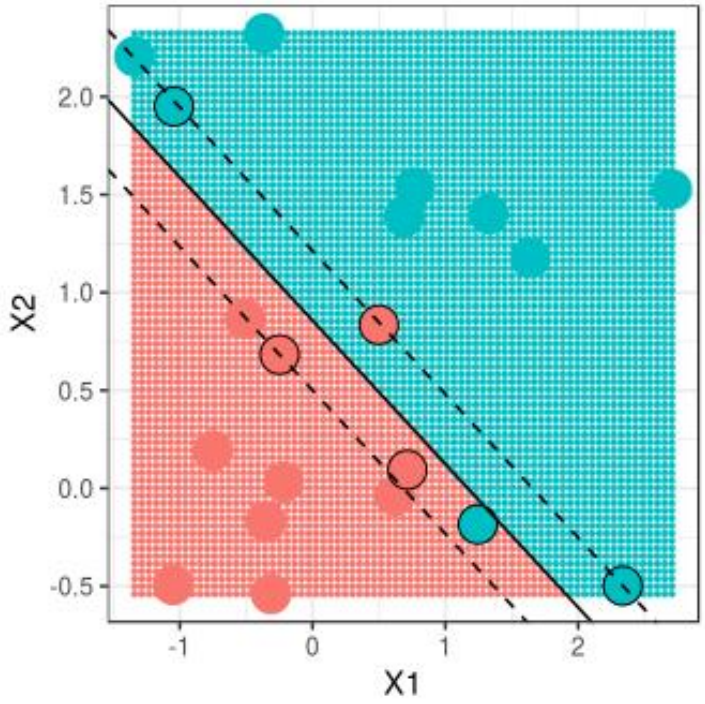
Support Vector Classifier o Soft Margin SVM

```
summary(modelo_svm)
```

```
##
## Call:
## svm(formula = y ~ X1 + X2, data = datos, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  6
##
##  ( 3 3 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

Support Vector Classifier o Soft Margin SVM

Veamos la estimación del SVC para el presente caso

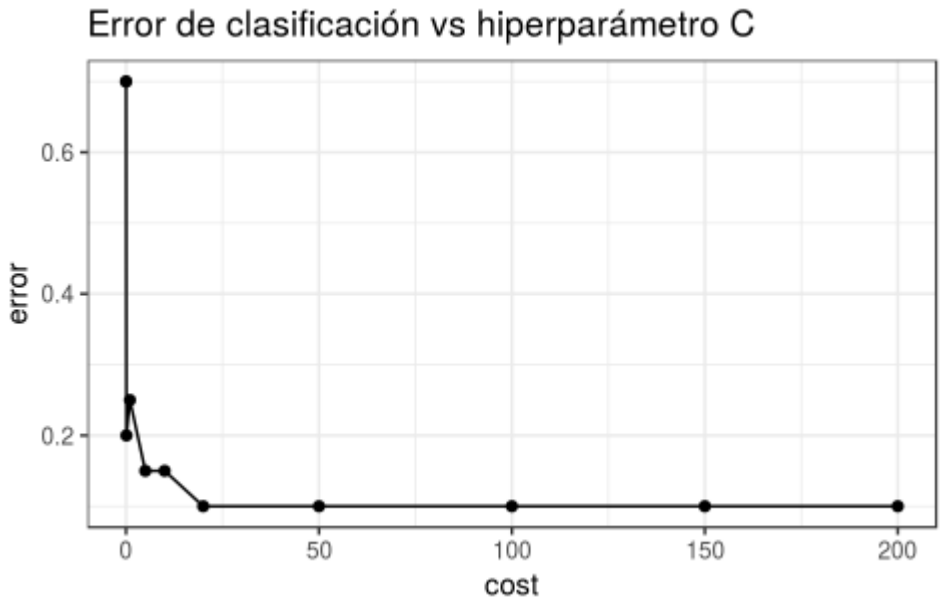


En el ajuste anterior se ha empleado un valor del hiperparámetro de penalización cost = 10. Este hiperparámetro determina el balance sesgo-varianza y por lo tanto, es crítico para la capacidad predictiva del modelo. Utilicemos la función tune() para hallar el mejor costo.

Support Vector Classifier o Soft Margin SVM

```r
set.seed(1)
svm_cv <- tune("svm", y ~ X1 + X2, data = datos, kernel = "linear", ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 20, 50, 100, 150, 200)))
summary(svm_cv)
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    20
##
## - best performance: 0.1
##
## - Detailed performance results:
##       cost error dispersion
## 1    0.001  0.70  0.4216370
## 2    0.010  0.70  0.4216370
## 3    0.100  0.20  0.2581989
## 4    1.000  0.25  0.2635231
## 5    5.000  0.15  0.2415229
## 6   10.000  0.15  0.2415229
## 7   20.000  0.10  0.2108185
## 8   50.000  0.10  0.2108185
## 9  100.000  0.10  0.2108185
## 10 150.000  0.10  0.2108185
## 11 200.000  0.10  0.2108185
```



Support Vector Classifier o Soft Margin SVM

El proceso de *cross-validation* muestra que el valor de penalización con el que se consigue menor *error rate* es 20 o superior. La función tune() almacena el mejor modelo de entre todos los que se han comparado:

```
mejor_modelo <- svm_cv$best.model
```

Obtenido el modelo final, se puede predecir la clase a la que pertenecen nuevas observaciones empleando la función predict().

```
table(predicción = predicciones, valor_real = test$y)
```

```
##           valor_real
## predicción -1  1
##         -1  7  1
##         1   1 11
```

18 de las 20 observaciones han sido clasificadas correctamente por el modelo. El test error es del 10 %.

Índice

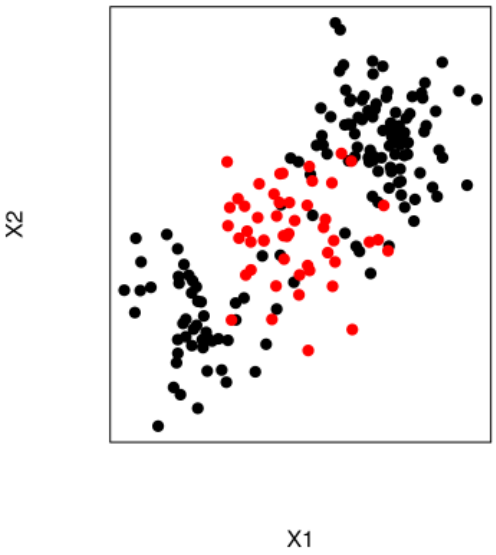
Máquinas de Vector Soporte (SVM)

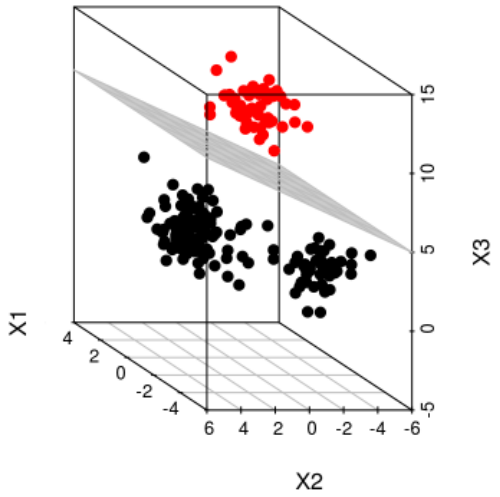
El *Support Vector Classifier* descrito en los apartados anteriores consigue buenos resultados cuando el límite de separación entre clases es aproximadamente lineal. Si no lo es…, su capacidad decae drásticamente. ¿Qué le hacemos? Una estrategia para enfrentarse a escenarios en los que la separación de los grupos es de tipo no lineal consiste en *expandir las dimensiones del espacio original*.

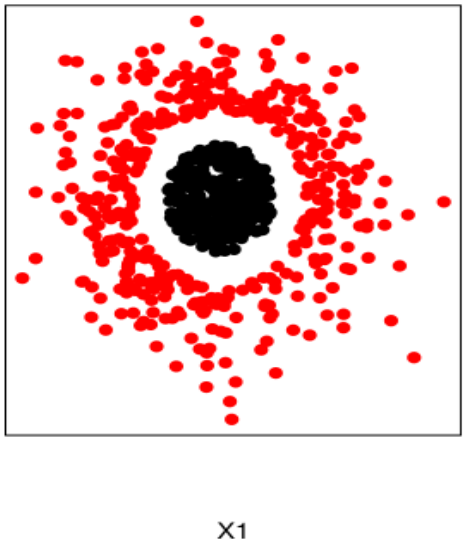
¿Cómo no se nos había ocurrido anterioremnete?

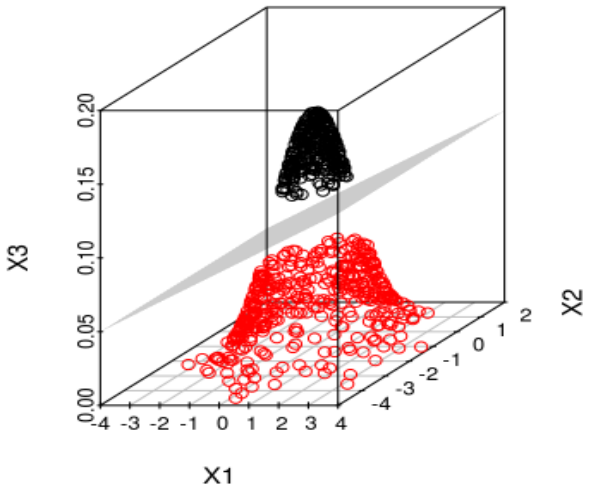
El hecho de que los grupos no sean linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. Las imágenes siguientes muestran como dos grupos, cuya separación en dos dimensiones no es lineal, sí lo es al añadir una tercera dimensión.

Máquinas de Vector Soporte (SVM)









Máquinas de Vector Soporte (SVM)

El método de Máquinas Vector Soporte (*SVM*) se puede considerar como una extensión del *Support Vector Classifier* obtenida al aumentar la dimensión de los datos. Los límites de separación lineales generados en el espacio aumentado se convierten en límites de separación no lineales al proyectarlos en el espacio original.

Una vez definido que las Máquinas de Vector Soporte siguen la misma estratégica que el *Support Vector Classifier*, pero aumentando la dimensión de los datos antes de aplicar el algoritmo, la pregunta inmediata es: ¿cómo se aumenta la dimensión y qué dimensión es la correcta?

Máquinas de Vector Soporte (SVM)

La dimensión de un conjunto de datos puede transformarse combinando o modificando cualquiera de sus dimensiones. Por ejemplo, se puede transformar un espacio de dos dimensiones en uno de tres aplicando la siguiente función:

$$f(x_1, x_2) = ({x_1}^2, \sqrt{2x_1}x2, {x_2}^2)$$

Esta es solo una de las infinitas trasformaciones posibles. Entonces, ¿cómo saber cuál es la adecuada? Es aquí donde los *kernel* entran en juego.

Un *kernel (K)* es una función que devuelve el resultado del producto punto entre dos vectores realizado en un nuevo espacio dimensional distinto al espacio original en el que se encuentran los vectores. Aunque no se ha entrado en detalle en las fórmulas matemáticas empleadas para resolver el problema de optimización, esta contiene un producto punto.

Máquinas de Vector Soporte (SVM)

Si se sustituye este *producto punto* por un *kernel*, se obtienen directamente los vectores soporte (y el hiperplano) en la dimensión correspondiente al *kernel*.

Ha esto se le suele conocer como ”*kernel trick*”, porque, con solo una ligera modificación del problema original, gracias a los *kernels*, se puede obtener el resultado para cualquier dimensión.

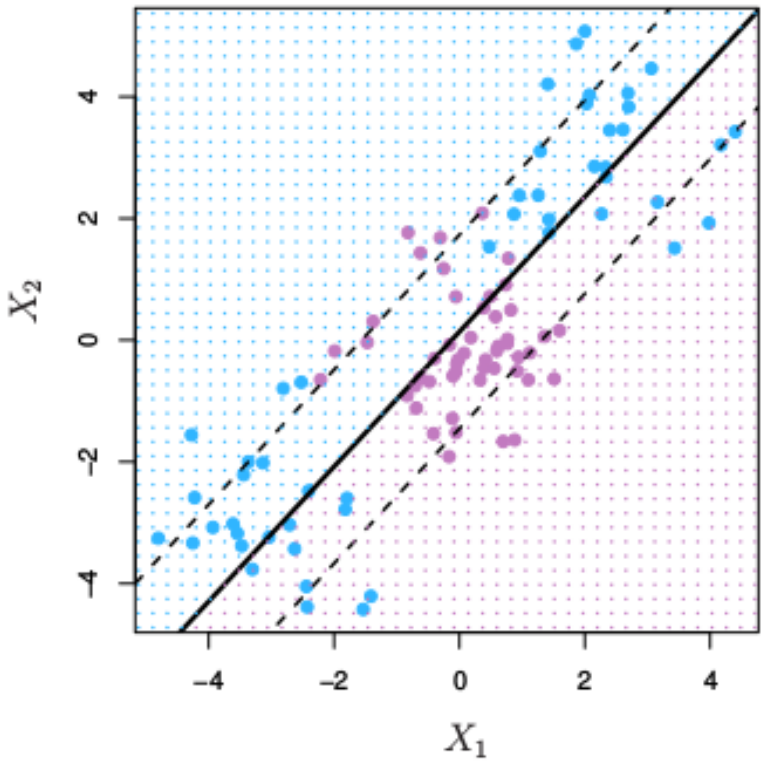
Existen multitud de *kernels* distintos, algunos de los más utilizados son: lineal, polinómico, el base radial, sigmoide, el más amado de está técnica, el gausiano (RBF).

Veamos algunos casos.

Máquinas de Vector Soporte (SVM)

Kernel lineal $K(x, x') = x \cdot x'$

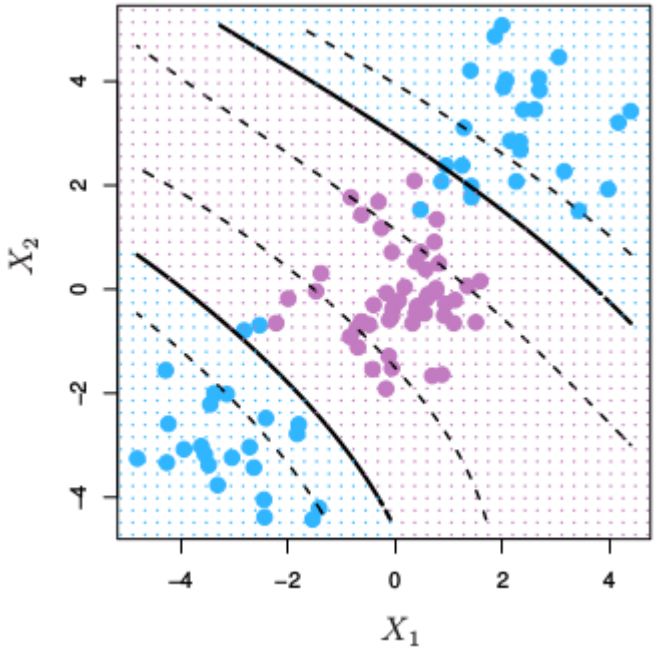
Si se emplea un Kernel lineal, el clasificador *Support Vector Machine* obtenido es equivalente al *Support Vector Classifier*.



Máquinas de Vector Soporte (SVM)

Kernel polinómico $K(x, x') = (x \cdot x' + c)^{d}$

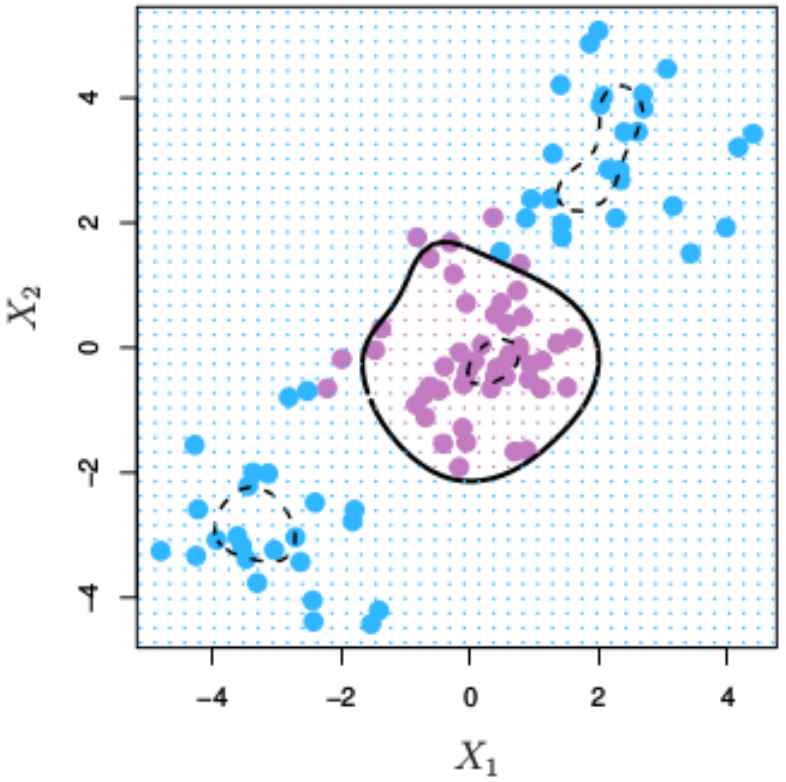
Cuando se emplea $d = 1$ y $c = 0$, el resultado es el mismo que el de un *kernel* lineal. Si $d > 1$, se generan límites de decisión no lineales, aumentando la no linealidad a medida que aumenta d. No suele ser recomendable emplear valores de d mayores 5 por problemas de sobre ajuste.



Máquinas de Vector Soporte (SVM)

Kernel gausiano $K(x, x') = exp(-\gamma \, || \, x \cdot x' ||^2)^{\,d}$

El valor de γ controla el comportamiento del *kernel*. Cuando es muy pequeño, el modelo final es equivalente al obtenido con un *kernel* lineal, a medida que aumenta su valor, también lo hace la flexibilidad del modelo



Máquinas de Vector Soporte (SVM)

Los *kernels* descritos son solo unos pocos de los muchos que existen. Cada uno tiene una serie de hiperparámetros cuyo valor óptimo puede encontrarse mediante validación cruzada.

No puede decirse que haya un *kernel* que supere al resto: depende en gran medida de la naturaleza del problema que se esté tratando.

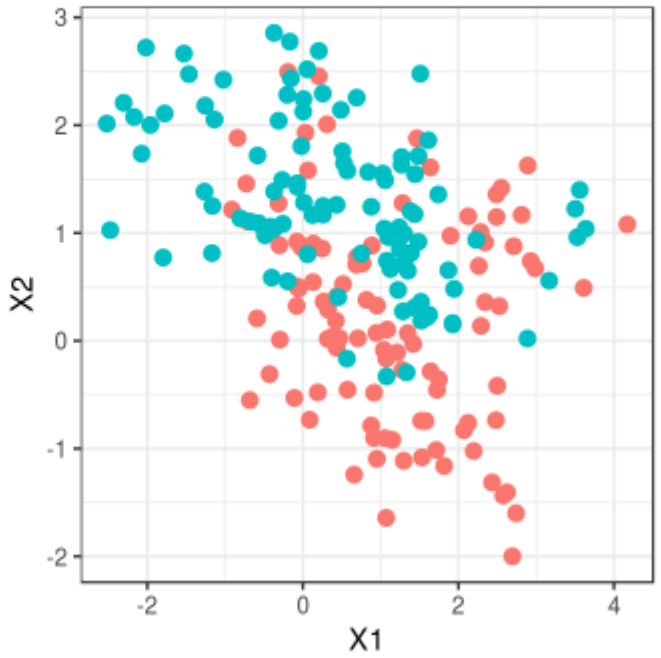
Ahora bien, tal como indican los autores de *A Practical Guide to Support Vector Classification*, es muy recomendable probar el *kernel RBF*. Este *kernel* tiene dos ventajas: que solo tiene dos hiperparámetros que optimizar (γ y la penalización $\boldsymbol{C}$ común a todos los *SVM*) y que su flexibilidad puede ir desde un clasificador lineal a uno muy complejo.

Máquinas de Vector Soporte (SVM)

- Veamos el siguiente ejemplo.

```
# Descargan los datos. Requiere conexión a internet
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
```

```
ggplot(data = datos, aes(x = X1, y = X2, color = y)) + geom_point(size = 2.5) +
    theme_bw() + theme(legend.position = "none")
```



Máquinas de Vector Soporte (SVM)

¿Qué le aplicarían uds?

Máquinas de Vector Soporte (SVM)

Entre los *kernel* no lineales que acepta la función svm() destacan

kernel = "polinomial", en cuyo caso hay que indicar el grado del polinomio d, y ***kernel = "radial"***, en cuyo caso hay que indicar el hiperparámetro parámetro γ.

Además de los hiperparámetros propios de cada *kernel*, todo *SVM* tiene también el hiperparámetro de penalización $\boldsymbol{C}$.

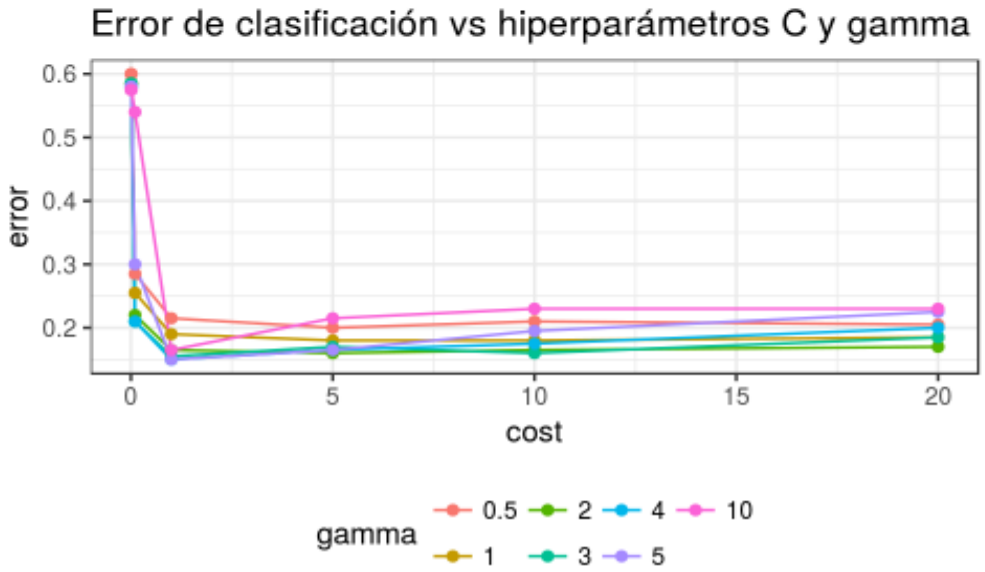
Veamos cómo es que identificamos

Máquinas de Vector Soporte (SVM)

```
library(e1071)
# Como los datos se han simulado en una misma escala, no es necesario
# estandarizarlos si no fuese así, es muy importante hacerlo.
set.seed(1)
svm_cv <- tune("svm", y ~ X1 + X2, data = datos, 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)))

ggplot(data = svm_cv$performances, aes(x = cost, y = error, color = as.factor(gamma))) +
    geom_line() + geom_point() + labs(title = "Error de clasificación vs hiperparámetros C y gamma",

    color = "gamma") + theme_bw() + theme(legend.position = "bottom")
```



Máquinas de Vector Soporte (SVM)

El mejor modelo:

```
svm_cv$best.parameters
```

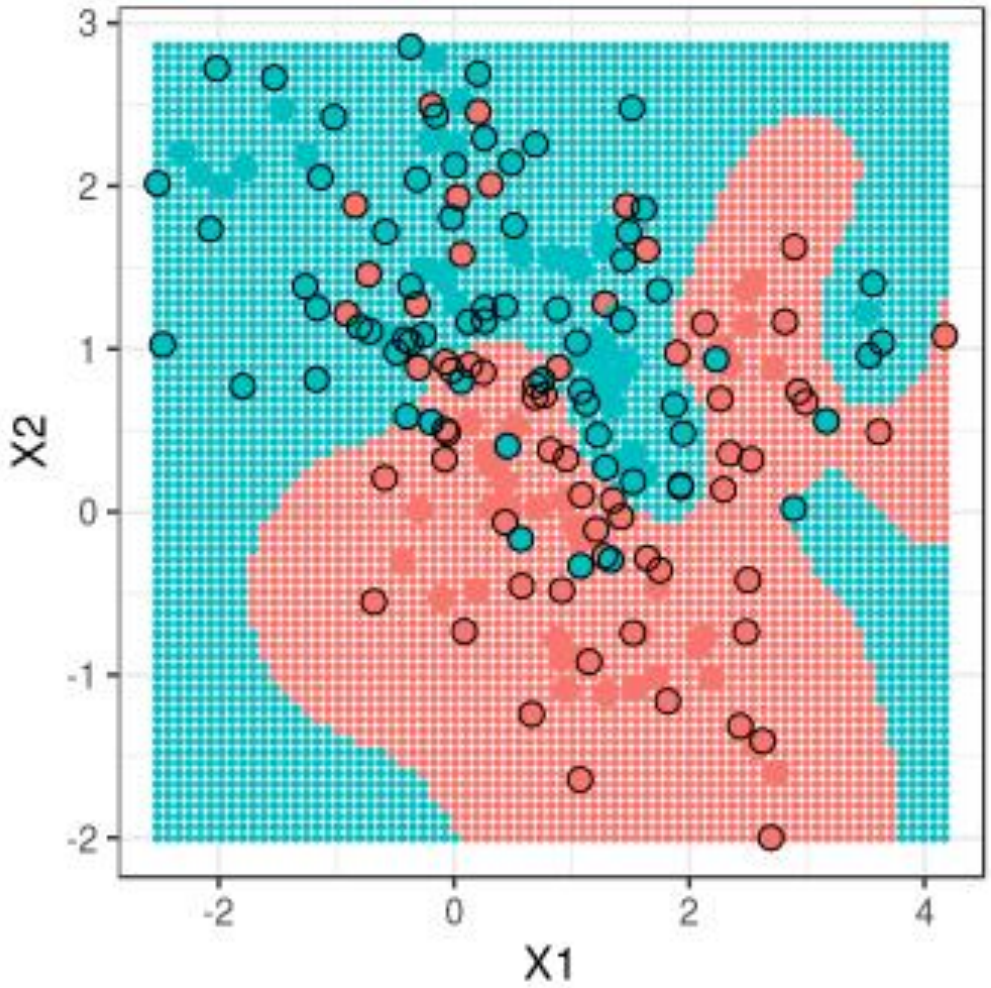
	cost <dbl>	gamma <dbl>
32	1	4

1 row

De entre todos los modelos estudiados, empleando cost = 1 y gamma = 5 se logra el menor *cv-test-error*. Veamos la representación gráfica del clasificador *SVM*.

Máquinas de Vector Soporte (SVM)

¿No les parece genial?



Índice

Otras variantes y consideraciones

SVM para más de dos clases

El concepto de hiperplano cambia. Los métodos más utilizados son los de one-versus-one, one-versus-all y el preferido DAGSMV

Otros algoritmos y formas de estimar

Veran Kernels de PCA y de todo tipo. Uno de los más fomosos es el del Algoritmo Perceptron.

Costo computacional del SVM

Pues es bastante costoso… al utilizar el producto punto de una matriz $n * n$, pues es uno de sus grandes problemas.

Índice

SVM en series temporales

- Esperemos un poco… no va a tardar a ser implementado en R o Pyrthon…

Conclusión y discusión

El presente capítulo presento la técnica de Máquinas de Vector Soporte y sus conceptos relacionados.

La obtención del hiperplano posee diversas consideraciones, dentro de ellas la escogencia del correcto kernel y los hiperparámetros asociados.

Las SVM marca un ante y un después en la conjunción de la escogencia funcional y los procesos de optimización informática.

Debemos esperar desarrollos de SVM para series de tiempos


The End