

UNIDAD 5. KNN, SVM (Support Vector Machines).

KNN (k-nearest neighbors)

KNN, k vecinos más próximos (Cover y Hart, 1967), es un método de clasificación supervisada basado en una idea muy simple e intuitiva: cada elemento es asignado a la misma clase que sus vecinos más cercanos. Se trata de un método no paramétrico, en el que no se hace ningún supuesto a priori sobre la distribución.

Los casos de entrenamiento son vectores en un espacio de dimensión m , descrito en términos de m atributos o variables explicativas, suponiendo conocida la clase a la que pertenece cada uno de los casos. Para la asignación de cada caso (punto o vector en ese espacio m -dimensional) se consideran los k elementos de la muestra más próximos a ese punto -los k vecinos más cercanos- y se asigna el caso a la clase C más frecuente entre ellos. Generalmente se utiliza la distancia euclídea.

Si el número de atributos es grande puede ocurrir que muchos de ellos sean irrelevantes, pero predominan en la asignación, ya que son mayoría, sobre los atributos importantes. Para corregir este efecto se identifican y eliminan previamente las variables superfluas, o bien se asigna un peso diferente a cada una en el cálculo de las distancias, reajustando los pesos durante el entrenamiento. Para ello se puede utilizar como ponderación la información mutua media $I(X_i, C)$ entre la variable i y la clase C , que mide la información que comparten X_i y C , e indica hasta que punto el conocimiento de una de ellas reduce nuestra incertidumbre sobre la otra. Si X_i y C son independientes, en cuyo caso X_i es irrelevante, la información mutua será cero; cuanto mayor sea la relación entre ambas, mayor será la información mutua.

También es posible atribuir un mayor peso en el cálculo de las frecuencias a los casos más próximos, utilizando por ejemplo el inverso del cuadrado de la distancia como factor de ponderación.

La elección de k , el número de vecinos que deben ser considerados para encontrar la clase más probable, depende de la aplicación. Los valores grandes –muchos vecinos– reducen el efecto de ruido en la clasificación, pero tienden a separar clases relativamente próximas. Generalmente se consideran adecuados valores de k entre 3 y 7.

El algoritmo es el siguiente:

- 1) Para cada caso i que debe ser asignado se calcula su distancia a cada uno de los casos ya asignados
- 2) Se seleccionan los k casos más próximos a i
- 3) Se asigna i a la clase más frecuente dentro del conjunto de k casos más próximos.

Cuando existe empate en las frecuencias de dos o más clases se utiliza alguna regla heurística, como elegir entre ellas la clase del vecino más próximo (es decir $k=1$), o bien la clase cuya distancia media a ese punto es menor dentro del conjunto.

Para la aplicación con R utilizaremos la función `knn` del paquete `class`, que no es necesario instalar (debe estar ya con la instalación básica de R), que emplea como criterio la distancia euclídea, y decide la asignación por mayoría entre los k vecinos más próximos (al azar en caso de empate). Si hay varios vecinos a la misma distancia que el k -ésimo se incluyen todos ellos. La sintaxis básica es:

```
knn(train, test, cl, k)
```

donde `train` es el conjunto de prueba, `test` el de validación, `cl` la variable que contiene la clase, y `k` el número de vecinos a considerar (1 por defecto).

Utilizaremos para el ejemplo el conjunto `algas`, ya conocido, con 31 casos de 6 tipos diferentes de algas, y 19 variables con la concentración relativa de diferentes pigmentos. El objetivo es identificar la clase de alga a partir de la composición relativa de pigmentos.

```
setwd("C:/CURSO DM")  
load("algas.RData") # lee el conjunto "algas"
```

Dividimos el conjunto de datos en dos partes; el primero con los 20 primeros casos, y el segundo con los 11 restantes, excluyendo en ambos la variable "clase", que obviamente no será utilizada como predictor.

```
entrenamiento <- subset(algas[1:20, ], select = -clase) # casos de entrenamiento  
prueba <- subset(algas[21:31, ], select = -clase) # casos de validación
```

```
{ construcción aleatoria de las dos submuestras, siempre recomendable:  
set.seed(123) # fija el arranque aleatorio, lo que permite reproducir la misma muestra  
muestra <- sample(1:nrow(algas), 20) # muestra de 20 números de caso entre 1 y 31  
entrenamiento <- subset(algas[muestra, ], select = -clase) # casos de entrenamiento  
prueba <- subset(algas[-muestra, ], select = -clase) # casos de validación }
```

Ahora aplicamos el método, indicando el conjunto de entrenamiento y el de validación, así como la variable que contiene la clase de alga. No configuramos el valor de k , el número de vecinos, por lo que será el valor por defecto $k=1$ (puede ser adecuado para conjuntos pequeños de datos como el nuestro):

```
library(class)  
resultados <- knn(entrenamiento, prueba, cl = algas[1:20,"clase"])
```

```
table(resultados, algas[21:31,"clase"])
```

	cianofíc.	Clorofíc.	criptofíc	diatomeas	dinofíceas	eustigmat
cianofíceas	3	0	0	0	0	0
clorofíceas	0	3	0	0	0	0
criptofíceas	0	0	1	0	0	0
diatomeas	0	0	0	2	0	0
dinofíceas	0	0	0	0	0	0
eustigmatofíceas	0	0	0	0	0	2

Todos los casos del conjunto de validación han sido identificados correctamente. Podemos aplicar el método al conjunto completo (que utilizamos como muestra de entrenamiento y de validación):

```
predictores <- subset(algas, select = -clase) # toda la muestra, excluyendo la variable clase
resultados <- knn(predictores,predictores, cl = algas$clase)
table(resultados, algas$clase)
```

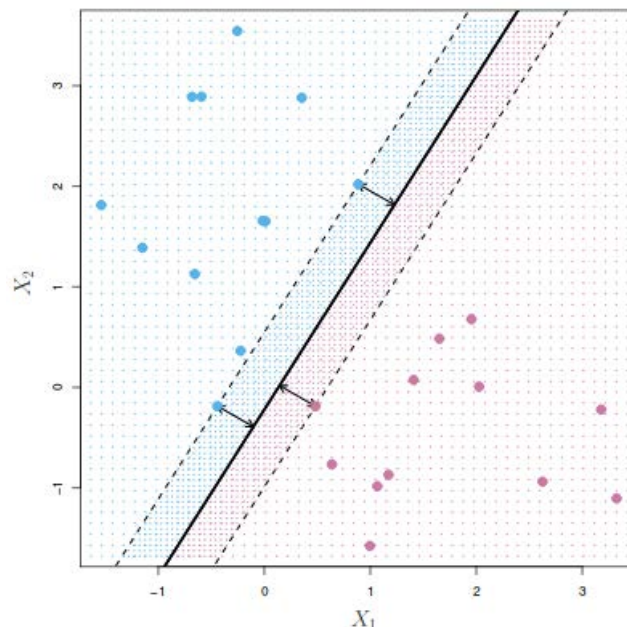
	cianofíc.	clorofíc.	criptofíc.	diatomeas	dinofíc.	eustigmat.
cianofíceas	4	0	0	0	0	0
clorofíceas	0	8	0	0	0	0
criptofíceas	0	0	4	0	0	0
diatomeas	0	0	0	6	0	0
dinofíceas	0	0	0	0	5	0
eustigmatofíceas	0	0	0	0	0	4

De nuevo comprobamos que todos los casos son asignados correctamente.

SVM (Support Vector Machines, Máquinas de Vector Soporte) (Vladimir Vapnik, 1963)

Cuando se dispone de un conjunto de n observaciones con k predictores y se desea predecir o identificar una variable binaria, con dos niveles, se pueden utilizar hiperplanos para predecir el grupo de pertenencia. Un hiperplano separa el espacio de los predictores en dos partes; en el caso más simple, con solamente dos predictores, el espacio completo es un plano y el hiperplano buscado es una recta: asignaremos el caso a una de las dos clases dependiendo del lado en que se encuentre el punto. La distancia de cada caso al hiperplano es un indicador de fiabilidad, cuanto más alejado más fiable es la asignación (si está sobre la recta o muy cerca podría pertenecer a cualquiera de las dos clases).

El mejor clasificador, el hiperplano más adecuado o hiperplano óptimo de separación (MMH, maximal margin hyperplane) es aquel que se encuentra más alejado de todas las observaciones de entrenamiento. Para ello se determina la distancia perpendicular de cada observación a un determinado hiperplano, y la menor de todas ellas es el “margen”; el mmh es el hiperplano que consigue un mayor margen. En la imagen que sigue (tomada del libro *Support Vector Machines Succinctly*, de Alexandre Kowalczyk), se muestra el hiperplano óptimo (la línea continua) para un ejemplo con dos clases identificadas con distinto color, y el margen -distancia mínima de los puntos de cada clase al hiperplano- indicado con líneas discontinuas.



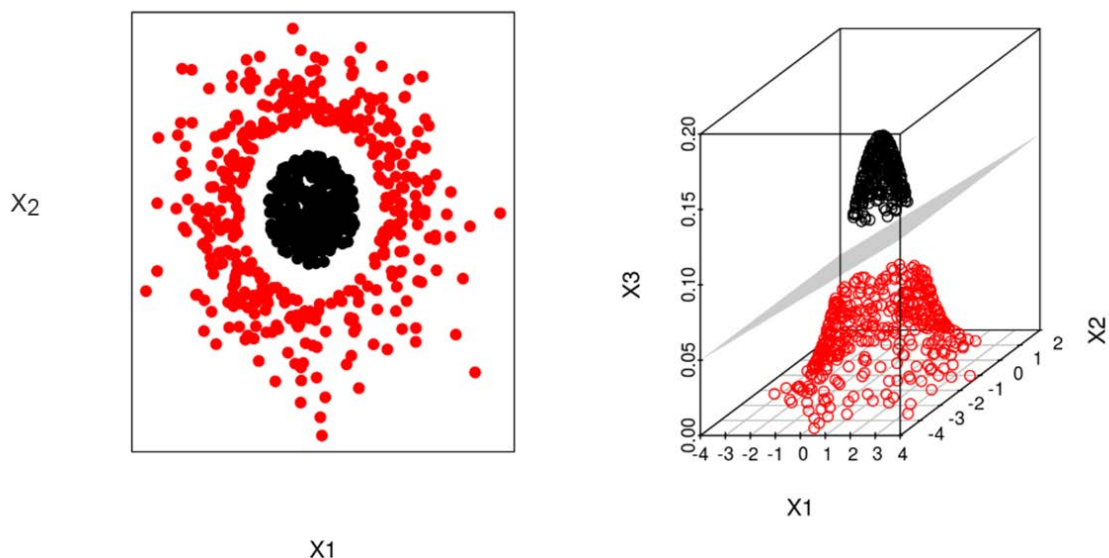
Cada observación es un vector de dimensión k (en el ejemplo dimensión 2). Las observaciones situadas a la distancia mínima (es decir el margen) se denominan “vectores soporte”, ya que permiten definir por completo el hiperplano óptimo; las restantes observaciones no tienen influencia alguna.

En la mayoría de las aplicaciones las observaciones están mezcladas y no permiten una separación completa, y por lo tanto no existe un hiperplano de separación MMH. Sin embargo se puede extender el concepto de margen (margen adaptado o soft margin) para obtener un hiperplano que casi separe las clases, permitiendo algunos errores de clasificación. Los vectores soporte son ahora los que se encuentran justo en el margen o los que lo incumplen (los que están más cerca del hiperplano).

Para ello se introduce un parámetro de ajuste adicional que supone una penalización de los casos de entrenamiento mal clasificados. El valor de ese parámetro se obtiene mediante validación cruzada, de tal modo que se consigue finalmente el clasificador que logra la mayor proporción de elementos correctamente clasificados cuando cada elemento de la muestra ha sido asignado sin emplear los datos de ese propio caso en la definición del criterio de asignación, es decir en la determinación del MMH.

Con este procedimiento se consiguen además dos ventajas adicionales: el método es más robusto (ya que no depende excesivamente de un número muy escaso de vectores soporte), y la validación cruzada reduce el problema de sobreajuste, común a muchas técnicas de minería de datos, según el cual el método funciona muy bien con los casos de la muestra, pero predice mal los casos nuevos no utilizados en el análisis.

El clasificador descrito funcionará razonablemente bien cuando el límite de separación entre las clases es aproximadamente lineal. Cuando no lo es, y los grupos no son linealmente separables, se utiliza una estrategia que consiste en expandir las dimensiones del espacio original. En la figura siguiente se muestran dos clases no separables linealmente (gráfico de la izquierda, en el que no es posible dibujar una recta que separe las dos clases), pero añadiendo una dimensión extra (x_3 en el gráfico de la derecha) es posible esa separación mediante un plano (en color gris).



El método de Máquinas de Vector Soporte (SVM) es una extensión del clasificador lineal descrito anteriormente aumentando la dimensión de los datos. Se aplica a los datos una transformación no lineal, y se construye un espacio de mayor dimensión provisto de un producto escalar (kernel o núcleo) en el cual aplicamos el mismo algoritmo obteniendo así una versión no lineal del clasificador. Existen diferentes posibilidades para ello, siendo los más habituales el kernel polinómico y el gaussiano (base radial).

Cuando el número de clases es mayor que dos se generaliza el procedimiento aplicando alguna de las siguientes estrategias: one-versus-one, one-versus-all y DAGSVM.

La primera, one-versus-one, consiste en aplicar el método a todos los posibles pares de las k clases, generando por lo tanto $k(k-1)/2$ predicciones, y asignando finalmente cada elemento a la clase que ha sido asignada con mayor frecuencia en el conjunto de predicciones. Esta es la opción por defecto de la función `svm` de R, aunque puede ser no adecuada –por el volumen de operaciones necesarias– cuando el número de clases es muy elevado.

La segunda, one-versus-all, compara cada clase con todas las restantes, obteniendo así k clasificadores, es decir un hiperplano para cada clase, que permite predecir la clase correspondiente a cada elemento. Sin embargo puede ocurrir que exista más de un clasificador positivo, en cuyo caso algunos elementos serán asignados a más de una clase.

La tercera, DAGSVM (Directed Acyclic Graph SVM) es una variante de la primera, one-versus-one, eliminando las comparaciones innecesarias: si en la comparación entre las clases i y j se obtiene que un elemento pertenece a la primera, podemos excluir la clase j de todas las comparaciones o pares de clases subsiguientes. De este modo en la asignación de cada elemento se excluyen en general la mayoría de las combinaciones o pares de clases, mejorando así el rendimiento del algoritmo.

Para su aplicación con R utilizaremos la función `svm` del paquete `e1071` que ya ha sido instalado en una unidad anterior.

La sintaxis básica de la función es:

```
svm(y ~ x, data)
```

Donde “ y ” es la variable de clasificación, o variable dependiente, y “ x ” el conjunto de predictores, que pueden ser cuantitativos o cualitativos. El argumento `data`, opcional, permite indicar el conjunto de datos al que pertenecen las variables de la fórmula.

Utilizaremos para el ejemplo el conjunto de datos “empresas”, contenido en el archivo “empresas.RData”, con una muestra de 1194 empresas de las cuales 165 son empresas culturales, y con 25 variables, relacionadas en la tabla siguiente, obtenidas del balance y cuenta de resultados de cada empresa. El objetivo es establecer si las variables contables están relacionadas con el carácter de empresa cultural, y en ese caso si es posible identificar si una empresa es cultural o no a partir de sus cuentas anuales (información libremente disponible en el Registro Mercantil).

VARIABLE	SIGNIFICADO
CULTURAL	Empresa cultural (1) o no cultural (0)
npac	ACTIVO CORRIENTE
npacc	Acreedores comerciales
npactt	ACTIVO TOTAL
npbaim	RESULTADO ANTES DE IMPUESTOS
npcaps	Capital Suscrito
npcone	Aprovisionamientos
npdfcp	Deudas a cp
npextc	Existencias
npfpro	Fondos Propios
npgger	Gastos de Personal
npimps	Impuesto sobre beneficios
npincn	Importe neto de la cifra de negocios
npingf	Ingresos Financieros
npmarv	Margen
npogex	Otros gastos de explotación
npoinf	Otros ingresos
nppatn	PATRIMONIO NETO
nppc	PASIVO CORRIENTE
npreje	RESULTADO DEL EJERCICIO
nprene	Rentabilidad Económica
nprenf	Rentabilidad Financiera
nprevs	Reservas
npexp	RESULTADO DE EXPLOTACIÓN
npes	Efectivo y otros activos líquidos

```
setwd("C:/CURSO DM")
load("empresas.RData")
names(empresas)
summary(empresas)
```

```
> names(empresas)
[1] "CULTURAL" "npac"      "npacc"     "npactt"    "npbaim"    "npcaps"
[7] "npcone"   "npdfcp"    "npextc"    "npfpro"    "npgger"    "npimps"
[13] "npincn"   "npingf"    "npmarv"    "npogex"    "npoinf"    "nppatn"
[19] "nppc"     "npreje"    "nprene"    "nprenf"    "nprevs"    "npexp"
[25] "npes"

> summary(empresas)

CULTURAL      npac      npacc      npactt      npbaim
0:1029 Min.   :    16615 Min.   :    229 Min.   :   40879 Min.   : -4132944
1: 165 1st Qu.:   255194 1st Qu.:   98685 1st Qu.:   432775 1st Qu.:   -8442
      Median :   658630 Median :   245198 Median :   1196432 Median :    7258
      Mean   :   5693995 Mean   :   2892705 Mean   :   13630905 Mean   :   1612010
      3rd Qu.:   1919394 3rd Qu.:   752458 3rd Qu.:   3135912 3rd Qu.:    55804
      Max.   :2421517000 Max.   :1467502000 Max.   :5501603000 Max.   :1485515000

      npcaps      npcone      npdfcp      npextc
Min.   :    2971 Min.   : -3736509000 Min.   :    0 Min.   :   -8241
1st Qu.:   12000 1st Qu.:  -1881934 1st Qu.:   21030 1st Qu.:   40000
Median :    54090 Median :   -509448 Median :    89893 Median :   167886
Mean   :   638138 Mean   :  -8317870 Mean   :   896782 Mean   :  1500006
3rd Qu.:   162632 3rd Qu.:  -167099 3rd Qu.:   347338 3rd Qu.:   530968
Max.   :  93500000 Max.   :    780862 Max.   :116743000 Max.   :416970000

      npfpro      npgger      npimps      npincn
Min.   : -3180689 Min.   : -130811000 Min.   : -70053000 Min.   :    52
1st Qu.:   132143 1st Qu.:  -594990 1st Qu.:  -13609 1st Qu.:   421716
Median :    411654 Median :  -235288 Median :   -1870 Median :   1065598
```

Mean : 7421842	Mean : -985653	Mean : -137112	Mean : 12013710
3rd Qu.: 1232870	3rd Qu.: -111509	3rd Qu.: 1714	3rd Qu.: 3265299
Max. : 3227989000	Max. : -3359	Max. : 2848000	Max. : 5363837000

npingf	npmarv	npogex	npoinf
Min. : -47	Min. : -46.94530	Min. : -634379000	Min. : -42
1st Qu.: 53	1st Qu.: 0.00035	1st Qu.: -443658	1st Qu.: 50
Median : 652	Median : 0.02225	Median : -148974	Median : 632
Mean : 63971	Mean : 0.84694	Mean : -1497239	Mean : 63869
3rd Qu.: 6076	3rd Qu.: 0.06705	3rd Qu.: -59719	3rd Qu.: 5732
Max. : 27458749	Max. : 54.94870	Max. : -2463	Max. : 27458749

nppatn	nppc	npreje	nprene
Min. : 1521	Min. : 7212	Min. : -5723801	Min. : -44.97920
1st Qu.: 133538	1st Qu.: 159101	1st Qu.: -6983	1st Qu.: 0.00032
Median : 425298	Median : 399854	Median : 5182	Median : 0.02655
Mean : 7465947	Mean : 4628631	Mean : 1474897	Mean : 0.96188
3rd Qu.: 1300917	3rd Qu.: 1211212	3rd Qu.: 43680	3rd Qu.: 0.06782
Max. : 3204901000	Max. : 2272580000	Max. : 1415462000	Max. : 56.03620

nprenf	nprevs	nprexp	npres
Min. : -78.50060	Min. : 1	Min. : -2708285	Min. : -105060
1st Qu.: -0.01738	1st Qu.: 49475	1st Qu.: -2676	1st Qu.: 13911
Median : 0.02530	Median : 240272	Median : 20260	Median : 45949
Mean : 0.86629	Mean : 4430496	Mean : 1666783	Mean : 741334
3rd Qu.: 0.10110	3rd Qu.: 825920	3rd Qu.: 90318	3rd Qu.: 155702
Max. : 53.00130	Max. : 2288286000	Max. : 1496275000	Max. : 612926000

Construimos los conjuntos de datos de entrenamiento (las 1000 primeras empresas) y de validación (las 194 restantes):

```
entrenamiento <- empresas[1:1000, ]
prueba <- empresas[1001:1194, ]
```

```
{ construcción aleatoria de ambos subconjuntos, siempre recomendable:
  set.seed(123)
  muestra <- sample(1:nrow(empresas), 1000)
  entrenamiento <- empresas[muestra, ]
  prueba <- empresas[-muestra, ] }
```

Ahora aplicamos el método SVM para construir el modelo, con la variable CULTURAL dependiente de todas las restantes:

```
library(e1071)
modelo <- svm(CULTURAL ~ ., data = entrenamiento)
summary(modelo)
```

La orden summary nos permite observar los resultados, el contenido del objeto “modelo”:

```
Call: svm(formula = CULTURAL ~ ., data = entrenamiento)
```

```
Parameters:
  SVM-Type: C-classification
  SVM-Kernel: radial
  cost: 1
  gamma: 0.04166667
```

```
Number of Support Vectors: 113
(53 60)
```

```
Number of Classes: 2
Levels: 0 1
```


El tipo de svm es “clasificación”, opción por defecto cuando la variable dependiente es un factor; svm también se puede utilizar como método de regresión si la variable dependiente es continua.

Por defecto el núcleo o kernel es radial, con la transformación $\exp(-\gamma \|u-v\|^2)$, donde γ es una constante. Podrían haberse utilizado otros: "linear", "polynomial", "sigmoid", mediante el argumento kernel (kernel = "linear") dentro de la función svm. Generalmente los mejores resultados se consiguen con la opción por defecto.

Cost = 1 indica el parámetro de ajuste adicional citado anteriormente para definir el margen adaptado: todas las observaciones dentro del margen de \pm Cost se consideran también vectores soporte, y no solamente las observaciones que están en el margen.

Gamma = 0.04166667 es el valor de la constante utilizada en la transformación núcleo; por defecto se utiliza el inverso del número de variables explicativas ($1/24=0.0416667$).

El número de vectores soporte encontrados es 113, divididos en dos partes con 53 y 60 puntos o vectores a cada lado del hiperplano. Si queremos saber (no es relevante) cuales son esos 113 casos (empresas) del conjunto de datos podemos ejecutar `modelo$index`.

A continuación, una vez construido el modelo, lo utilizamos para predecir la clase (empresa cultural o no). Observamos los resultados, elaborando la matriz de confusión, o tabla de doble entrada con la predicción y la clase real:

```
resultados.entrenamiento <- predict(modelo, newdata = entrenamiento, type = "class")
table(resultados.entrenamiento, entrenamiento$CULTURAL)
```

```
> table(resultados.entrenamiento, entrenamiento$CULTURAL)
```

```
resultados.entrenamiento    0    1
                           0 860   20
                           1   0  120
```

Observamos que la mayoría de las empresas han sido identificadas correctamente: todas las no culturales (860), y 120 de las 140 culturales. ¿Será capaz nuestro clasificador svm de identificar también correctamente las empresas no utilizadas para su construcción? Aplicamos el modelo al conjunto de prueba:

```
resultados.prueba <- predict(modelo, newdata = prueba, type = "class")
t <- table(resultados.prueba, prueba$ CULTURAL)
t ; 100 * sum(diag(t)) / sum(t)
```

```
resultados.prueba    0    1
                   0 165    3
                   1   4   22
[1] 96.39175
```

El 96,39% de las empresas de la muestra de validación son identificadas correctamente.