

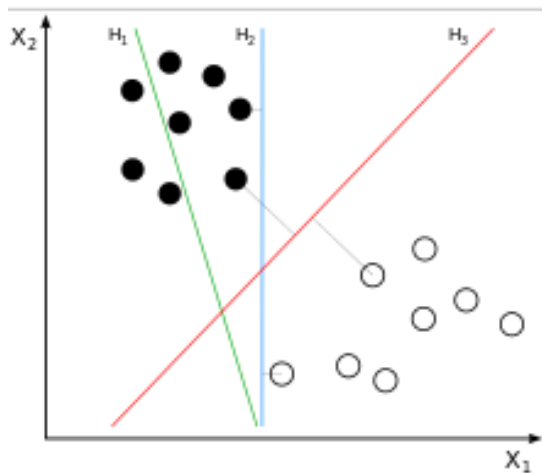
# Support Vector Machines

# Support Vector Machines

Se trata de plantear el problema de separación lineal de clases con métodos algebraicos (buscar el hiperplano de separación). Se basa en tres ideas importantes.

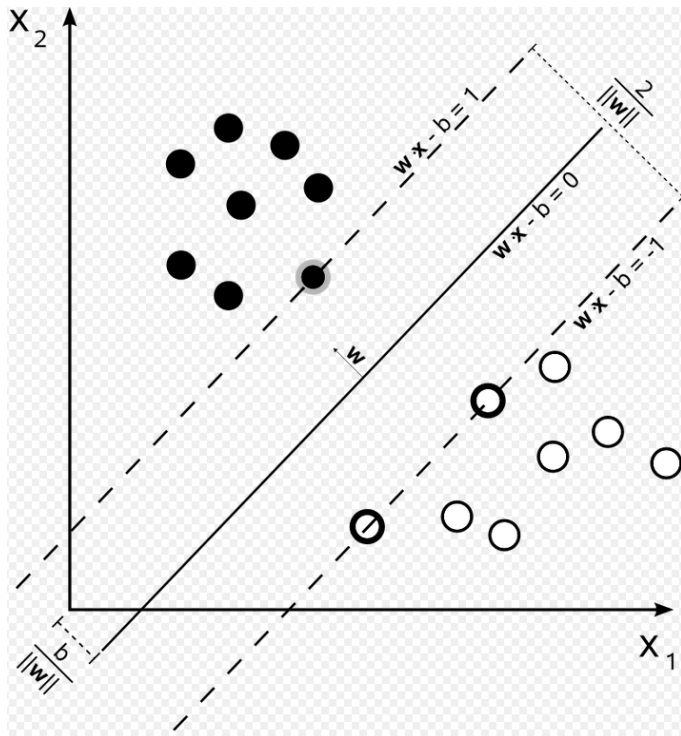
## 1) Primera idea importante: el “maximal margin” (Vapnik, 1963)

Se trata, no solamente de separar las clases por un hiperplano (función lineal), sino de incluir en la decisión de la construcción del separador, el concepto de separador con máximo margen. Esto a menudo mejora tanto el sesgo como la varianza de los resultados.



$H_1$  does not separate the classes.  
 $H_2$  does, but only with a small margin.  
 $H_3$  separates them with the maximum margin.

Se trata de hallar el vector de parámetros  $\mathbf{w}$  que maximice el margen. Las ecuaciones de los hiperplanos que delimitan el margen son  $\mathbf{w}\mathbf{x}=1$  y  $\mathbf{w}\mathbf{x}=-1$ . Denotando  $y$  como  $(-1,1)$  en el problema de clasificación, hay que maximizar la distancia entre los dos hiperplanos de separación ( $2/\text{norma de } \mathbf{w}$ ). Se suele hacer con métodos clásicos de optimización (Lagrange, Kuhn Tucker).



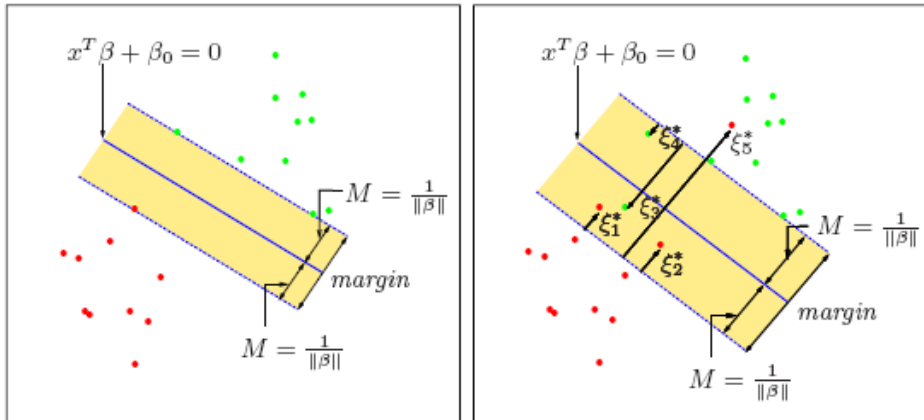
$$\arg \min_{(\mathbf{w}, b)} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to (for any  $i = 1, \dots, n$ )

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$$

## 2) Segunda idea importante: el “soft margin” (Cortes, Vapnik, 1995)

La separación perfecta no suele existir, y es necesario permitir observaciones mal clasificadas por los separadores para no incurrir en sobreajustes.



Esto implica añadir una variable  $\xi$  de “residuo” y constante  $C$  de regularización del margen que está relacionada inversamente con la anchura del margen y el “permiso para fallar” que vamos a permitir en la construcción del separador. A mayor  $C$  (menores “residuos”  $\xi_i$ ), menor margen. A menor  $C$  (permitimos mayores residuos  $\xi_i$ ), más permiso para fallar y más margen.

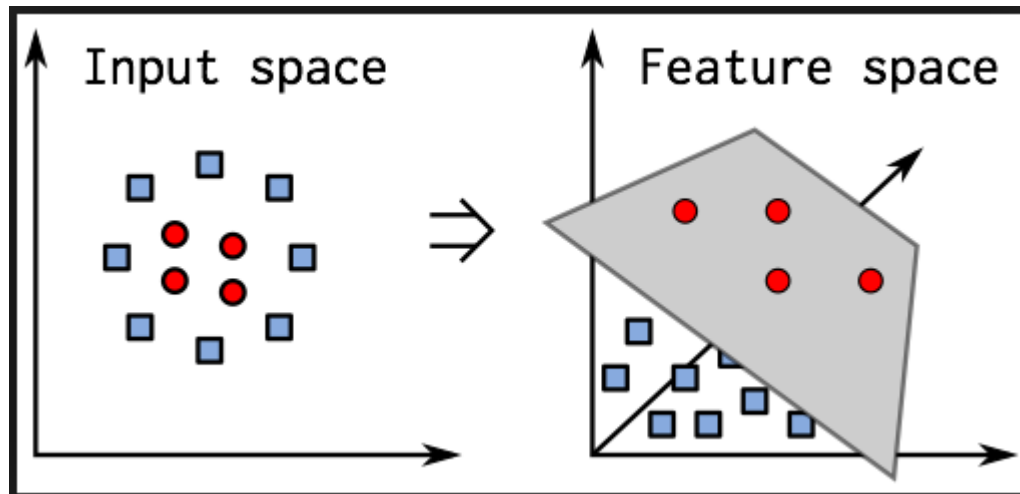
$$\arg \min_{\mathbf{w}, \xi, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \right\}$$

subject to (for any  $i = 1, \dots, n$ )

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

### 3) Tercera idea importante: el Kernel (Boser, Guyon, Vapnik, 1992)

La separación entre clases en muchos problemas no es lineal. Una idea para aplicar a pesar de todo un algoritmo de separación lineal es trabajar en un **espacio de dimensión superior donde sí tenga sentido la separación lineal.**



(Por ejemplo, supongamos que a la tabla  $x_2^2$ , calculada directamente:

y	x1	x2	$x_2^2$
-1	3	4	16
1	4	6	36
...	...	...	

y	x1	x2
-1	3	4
1	4	6
...	...	...

añadimos una variable nueva,

Entonces en ese espacio de 3 dimensiones sí se podría separar linealmente como en el ejemplo de la figura).

Para ello, se pueden introducir nuevas funciones de las variables ( $x^2$ ,  $x^3$ ,  $x_1x_2$ , etc.) lo que aumenta la dimensión del vector de variables independientes. Entonces sería más fácil para el algoritmo encontrar separaciones lineales y buen tamaño de margen.

La función  $\phi(\mathbf{x})$  representa una función que extrapola de la dimensión original a una superior. Por ejemplo, de 3 dimensiones pasamos a 9:

$$\phi(x_1, x_2, x_3) = (x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2, x_3x1, x_3x2, x_3x3)$$

El problema es que este aumento de dimensión hace a menudo impracticables los cálculos, pero aquí interviene el “truco Kernel” (“the Kernel trick”): cualquier algoritmo que dependa solo de los productos escalares (como es el caso del SVM) permite trabajar computacionalmente en una dimensión controlada a través de una función llamada Kernel, que tiene que cumplir:

$$K(x,y) = \langle \phi(x), \phi(y) \rangle.$$

## Ejemplo

En el caso anterior, el Kernel asociado a  $\varphi(x_1, x_2, x_3)$  es la función  $K(x,y)=(\langle x,y \rangle)**2$  pues se cumple la simetría  $K(x,y)=\langle \varphi(x), \varphi(y) \rangle$ .

$$\varphi(x_1, x_2, x_3)=(x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$$

Supongamos  $x = (1, 2, 3)$  y  $y = (4, 5, 6)$ .

Si queremos calcular  $\langle \varphi(x), \varphi(y) \rangle$ :

$$\varphi(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$$

$$\varphi(y) = (16, 20, 24, 20, 25, 36, 24, 30, 36)$$

$$\langle \varphi(x), \varphi(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

Ahora si en lugar de calcular ese producto haciendo todas esas operaciones utilizamos el Kernel, todo sería más sencillo:

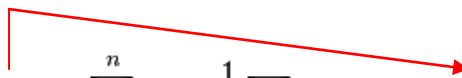
$$K(x, y) = (\langle x,y \rangle)**2 = (4 + 10 + 18) **2 = 32**2 = 1024$$

Por lo tanto no ha sido necesario crear nuevas variables, ni utilizarlas como tal , ahorrando problemas de todo tipo. El problema de optimización puede plantearse en su forma dual, como función de productos escalares  $\mathbf{x}_i^T \mathbf{x}_j$  (ver formulación más abajo).

Sustituyendo en el problema de optimización  $\mathbf{x}_i^T \mathbf{x}_j$  por su Kernel  $k(\mathbf{x}_i, \mathbf{x}_j)$ , implícitamente estamos aumentando la dimensión del espacio de variables de cara a la construcción de hiperplanos de separación, **sin pasar realmente por la creación de nuevas variables.**

Using the fact that  $\|\mathbf{w}\|^2 = \mathbf{w}^T \cdot \mathbf{w}$  and substituting  $\mathbf{w} = \sum_{i=1} \alpha_i y_i \mathbf{x}_i$ , one can show that the dual of the SVM reduces to the following optimization problem:

Maximize (in  $\alpha_i$  )

$$\tilde{L}(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$


subject to (for any  $i = 1, \dots, n$ )

$$\alpha_i \geq 0,$$

and to the constraint from the minimization in  $b$

$$\sum_{i=1}^n \alpha_i y_i = 0.$$

Here the kernel is defined by  $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$ .

$\mathbf{w}$  can be computed thanks to the  $\alpha$  terms:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$



Obviamente hay que utilizar Kernels estándar que cumplen la propiedad. El más frecuente de los no lineales es el RBF Gaussiano (necesita un parámetro sigma o gamma). El Kernel polinomial necesita un parámetro de grado del polinomio y el sigmoide suele necesitar un parámetro de posición y otro de escala.

Linear function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
Polynomial function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r)^d$
RBF function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\sigma^2}\right)$ $= \exp(-\gamma \ \mathbf{x}_i - \mathbf{x}_j\ ^2), \gamma = \frac{1}{2\sigma^2}$
Sigmoid function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r)$ $\tanh(x) = \frac{e^x}{e^x + 1}$

## Parámetros a considerar en SVM

1) Parámetro C. Aumentar C implica menor sesgo y mayor sobreajuste. Su rango depende mucho de los datos.

2) La función Kernel (no siempre es necesaria) y su(s) parámetro(s).

RBF: Aumentar gamma en la función RBF implica menor sesgo y mayor sobreajuste.

Polinomial. Aumentar el grado del polinomio implica menor sesgo y mayor sobreajuste.

Hay interdependencia entre ambos parámetros. Ver página web

[http://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html)

## Ventajas de SVM

(ver <http://www.svms.org/> para más detalles sobre SVM)

- Muy flexible, sobre todo por el truco Kernel. Hay versión para regresión y para clasificación multinomial. Ambas tienen buenas propiedades también.
- Puede competir en datos separables linealmente con la regresión logística.
- Buena performance en clasificación de imágenes.

## Desventajas de SVM

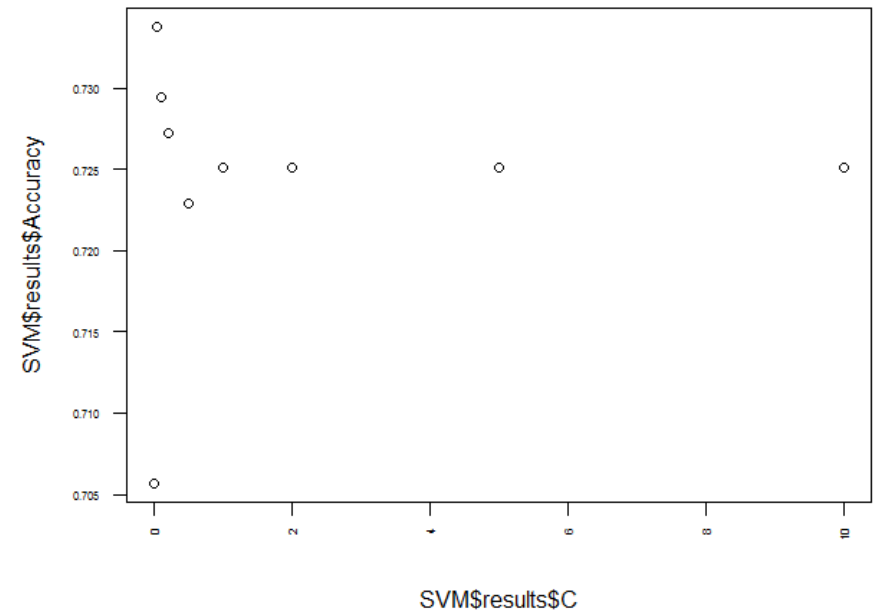
- Lento a veces en la optimización
- Dificultad en seleccionar la función Kernel y sus parámetros asociados
- Valores missings, categorías poco representadas, variables irrelevantes, etc. son un problema como en los algoritmos clásicos.

# Tuneado con caret y pruebas

```
# *****  
# TUNEADO SVM BINARIA  
# *****  
  
load ("c:/saheartbis.Rda")  
  
# SVM LINEAL: SOLO PARÁMETRO C  
  
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))  
  
control<-trainControl(method = "cv",number=4,savePredictions = "all")  
  
SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+  
  typea+famhist.Absent,  
  method="svmLinear",trControl=control,  
  tuneGrid=SVMgrid,verbose=FALSE)  
SVM  
SVM$results  
plot(SVM$results$C,SVM$results$Accuracy)
```

C	Accuracy	Kappa
0.01	0.7056784	0.2413527
0.05	0.7337706	0.3773863
0.10	0.7294228	0.3675741
0.20	0.7272489	0.3730833
0.50	0.7229198	0.3655104
1.00	0.7250937	0.3712509
2.00	0.7250937	0.3712509
5.00	0.7250937	0.3712509
10.00	0.7250937	0.3712509

Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was C = 0.05.



# El tuneado puede tardar mucho pero es necesario

```
# SVM Polinomial: PARÁMETROS C, degree, scale
```

```
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10),  
  degree=c(2,3),scale=c(0.1,0.5,1,2,5))
```

```
control<-trainControl(method = "cv",  
  number=4,savePredictions = "all")
```

```
SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+  
  typea+famhist.Absent,  
  method="svmPoly",trControl=control,  
  tuneGrid=SVMgrid,verbose=FALSE)  
SVM
```

Accuracy was used to select the optimal model using the largest value.

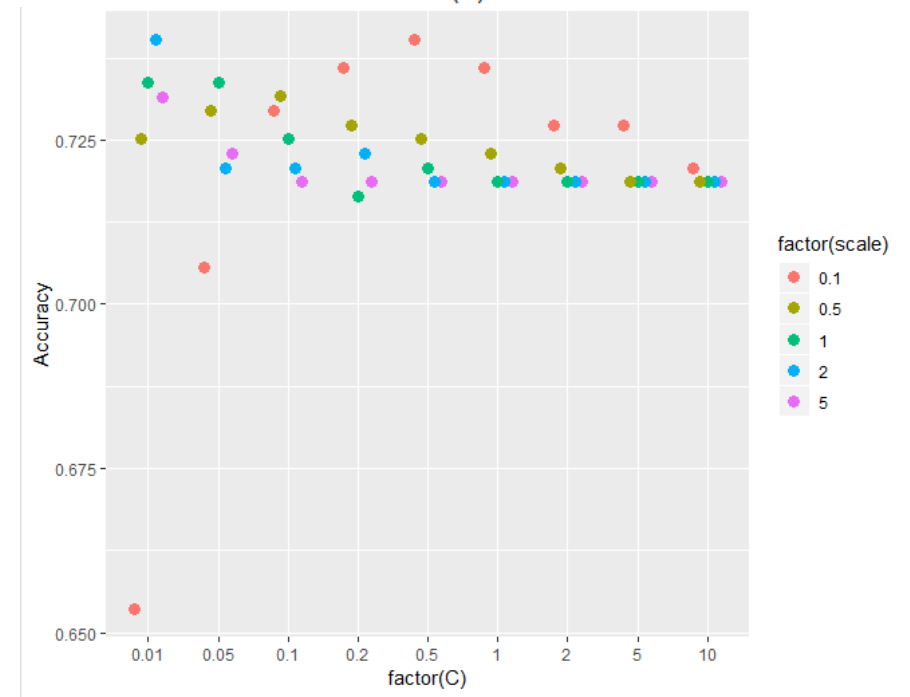
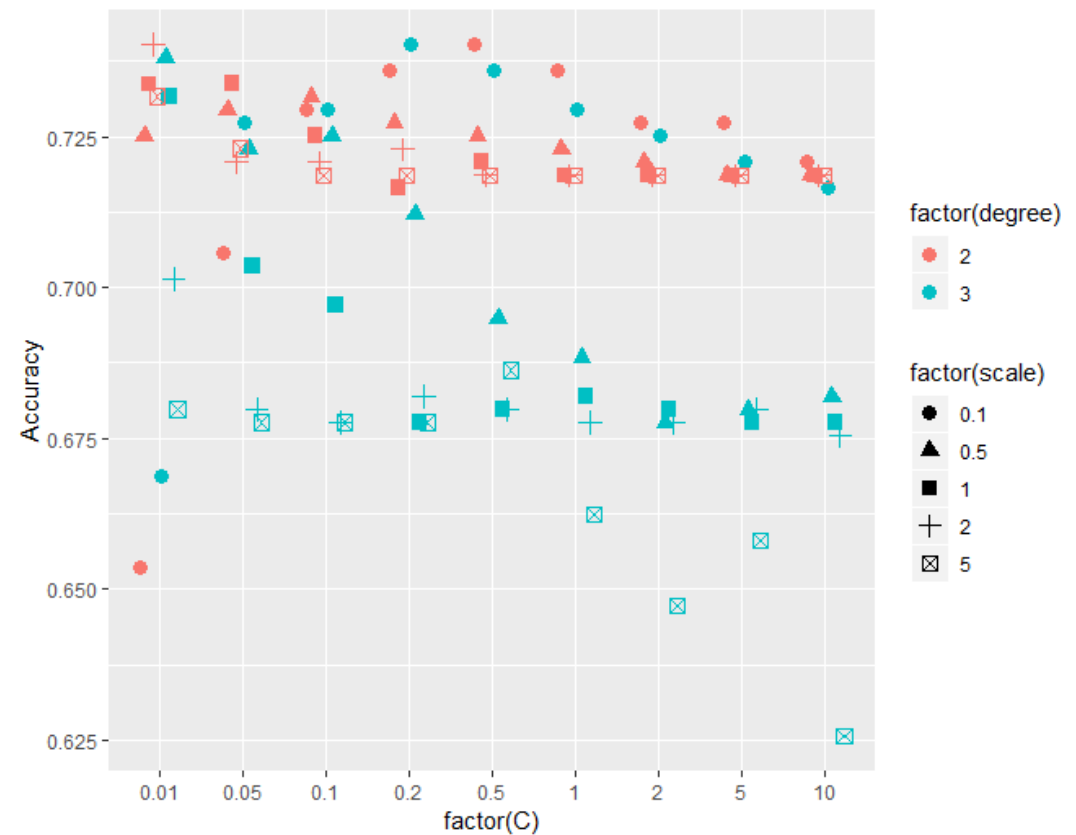
The final values used for the model were  
degree = 2, scale = 2 and C  
= 0.01.

```
SVM$results
```

```
# LOS GRÁFICOS DOS A DOS NO SIRVEN  
# plot(SVM$results$C,SVM$results$Accuracy)  
# plot(SVM$results$degree,SVM$results$Accuracy)  
# plot(SVM$results$scale,SVM$results$Accuracy)  
dat<-as.data.frame(SVM$results)  
library(ggplot2)  
# PLOT DE DOS VARIABLES CATEGÓRICAS, UNA CONTINUA  
ggplot(dat, aes(x=factor(C), y=Accuracy,  
  color=factor(degree),pch=factor(scale))) +  
  geom_point(position=position_dodge(width=0.5),size=3)
```

```
# SOLO DEGREE=2  
dat2<-dat[dat$degree==2,]
```

```
ggplot(dat2, aes(x=factor(C), y=Accuracy,  
  colour=factor(scale))) +  
  geom_point(position=position_dodge(width=0.5),size=3)
```



```
# SVM RBF: PARÁMETROS C, sigma
```

```
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),
  sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))
```

```
control<-trainControl(method = "cv",
  number=4,savePredictions = "all")
```

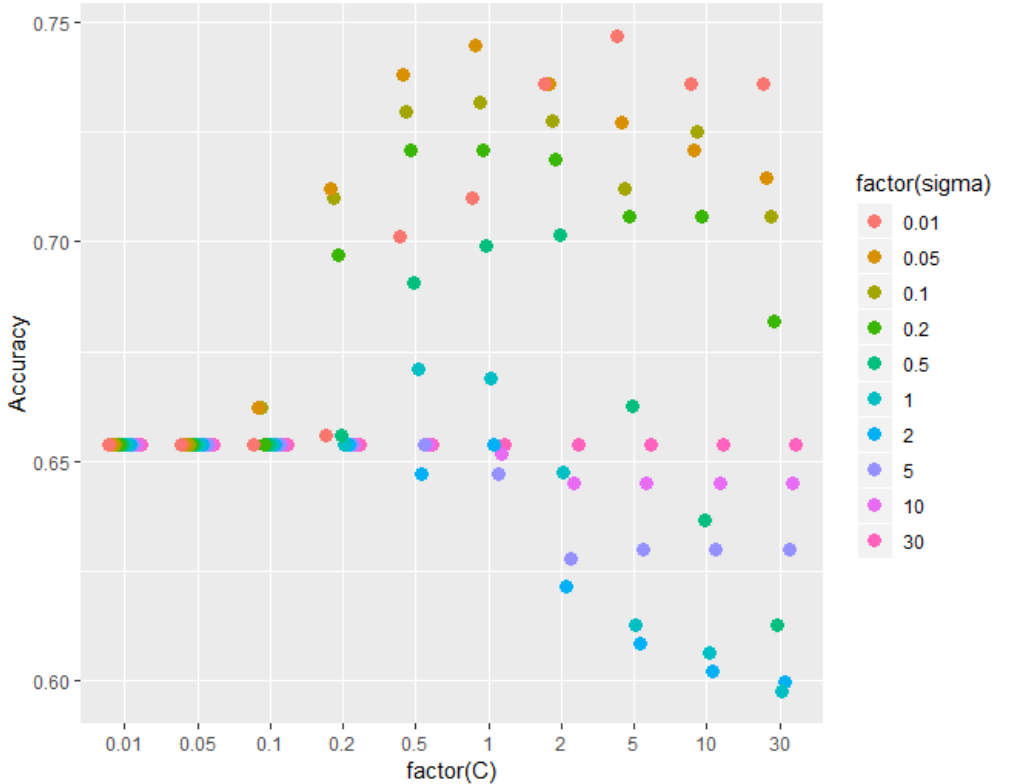
```
SVM<- train(data=saheartbis, factor(chd)~sbp+tobacco+ldl+age+
  typea+famhist.Absent,
  method="svmRadial", trControl=control,
  tuneGrid=SVMgrid, verbose=FALSE)
```

SVM

```
dat<-as.data.frame(SVM$results)
```

```
ggplot(dat, aes(x=factor(C), y=Accuracy,
  color=factor(sigma))) +
  geom_point(position=position_dodge(width=0.5), size=3)
```

Accuracy was used to select the optimal model using the largest value.  
The final values used for the model were  $\sigma = 0.01$  and  $C = 5$ .

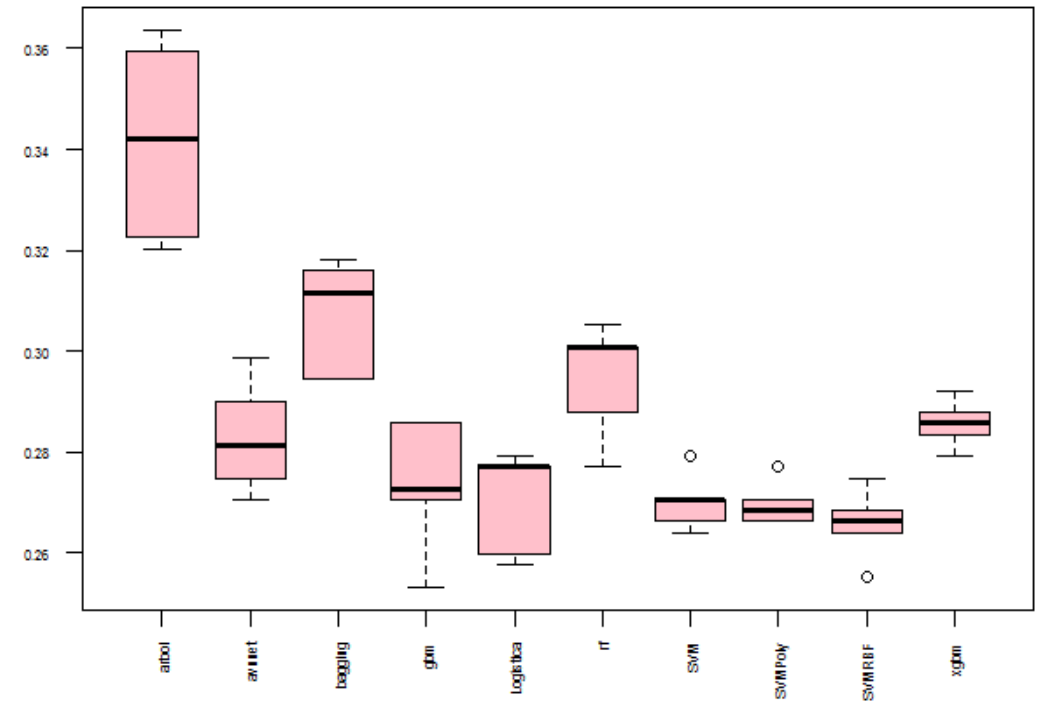


# Comparación con otros modelos vía validación cruzada repetida

...

```
medias8<-cruzadaSVMbin(data=saheartbis, vardep="chd",  
  listconti=c("sbp", "tobacco",  
    "ldl","age", "typea","famhist.Absent"),  
  listclass=c(""),  
  grupos=4,sinicio=1234, repe=5,  
  C=0.05)  
  
medias8$modelo="SVM"  
  
medias9<-cruzadaSVMbinPoly(data=saheartbis, vardep="chd",  
  listconti=c("sbp", "tobacco",  
    "ldl","age", "typea","famhist.Absent"),  
  listclass=c(""),  
  grupos=4,sinicio=1234, repe=5,  
  C=0.01,degree=2,scale=2)  
medias9$modelo="SVMPoly"  
  
medias10<-cruzadaSVMbinRBF(data=saheartbis, vardep="chd",  
  listconti=c("sbp", "tobacco",  
    "ldl","age", "typea","famhist.Absent"),  
  listclass=c(""),  
  grupos=4,sinicio=1234, repe=5,  
  C=5,sigma=0.01)  
  
medias10$modelo="SVMRBF"  
  
union1<-rbind(medias1,medias2,medias3,medias4,medias5,  
  medias6,medias7,medias8,medias9,medias10)  
  
par(cex.axis=0.5)  
boxplot(data=union1,tasa~modelo,main="TASA FALLOS",col="pink")
```

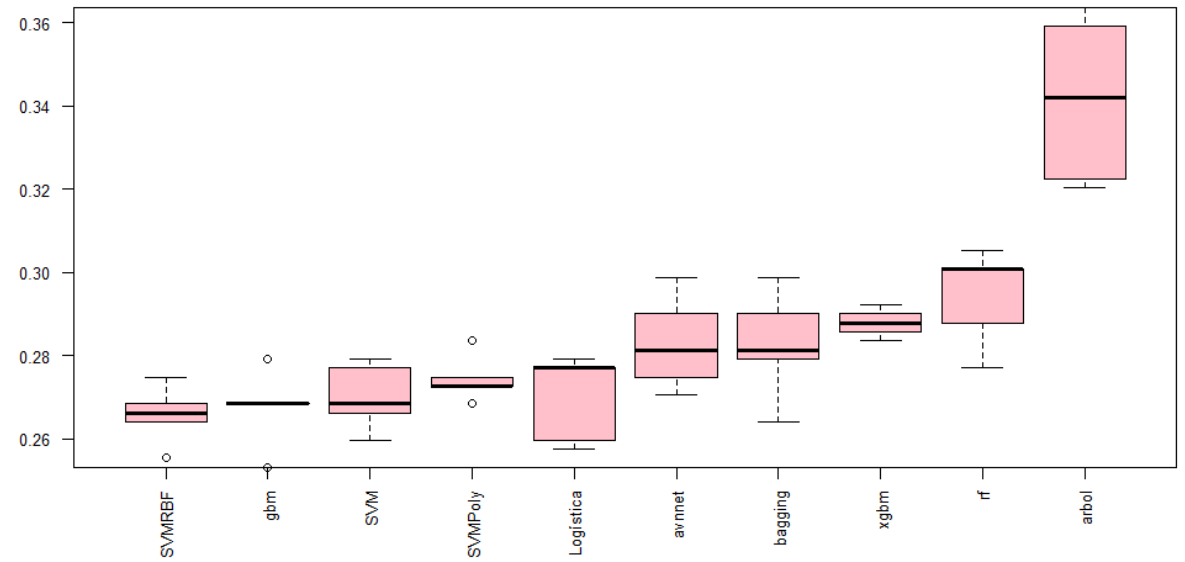
**TASA FALLOS**



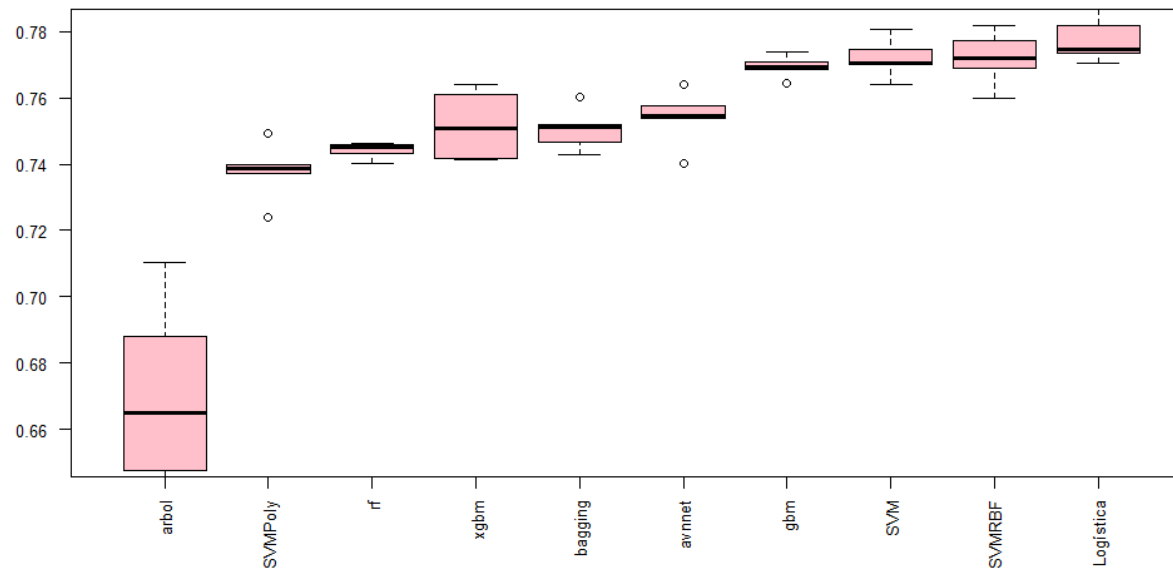
```
uni<-union1
uni$modelo <- with(uni,
  reorder(modelo,tasa, median))
par(cex.axis=0.8,las=2)
boxplot(data=uni,tasa~modelo,col="pink",main="TASA FALLOS")
```

```
uni<-union1
uni$modelo <- with(uni,
  reorder(modelo,auc, median))
par(cex.axis=0.8,las=2)
boxplot(data=uni,auc~modelo,col="pink",main="AUC")
```

**TASA FALLOS**



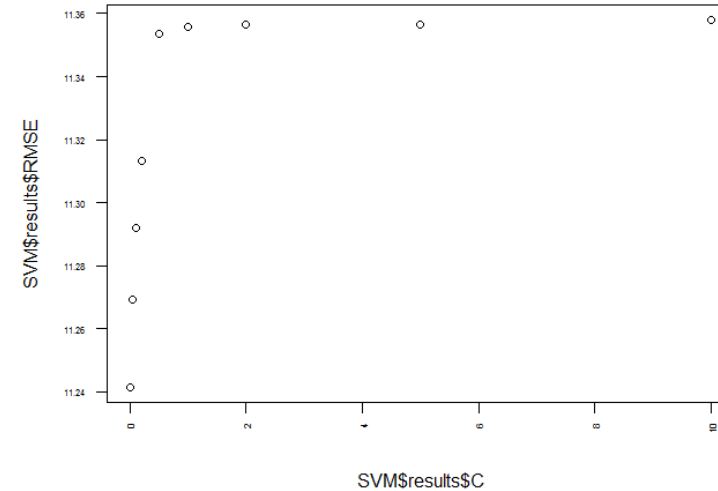
**AUC**





## Ejemplo variable dependiente continua (criterio: RMSE (cuanto más bajo mejor))

```
# *****  
# TUNEADO SVM CONTINUA  
# *****  
load ("c:/compressbien.Rda")  
# SVM LINEAL: SOLO PARÁMETRO C  
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))  
control<-trainControl(method = "cv",number=4,  
  savePredictions = "all")  
SVM<- train(data=compressbien,cstrength~age+water+cement+blast,  
  method="svmLinear",trControl=control,  
  tuneGrid=SVMgrid,verbose=FALSE)  
  
SVM$results  
plot(SVM$results$C,SVM$results$RMSE)
```



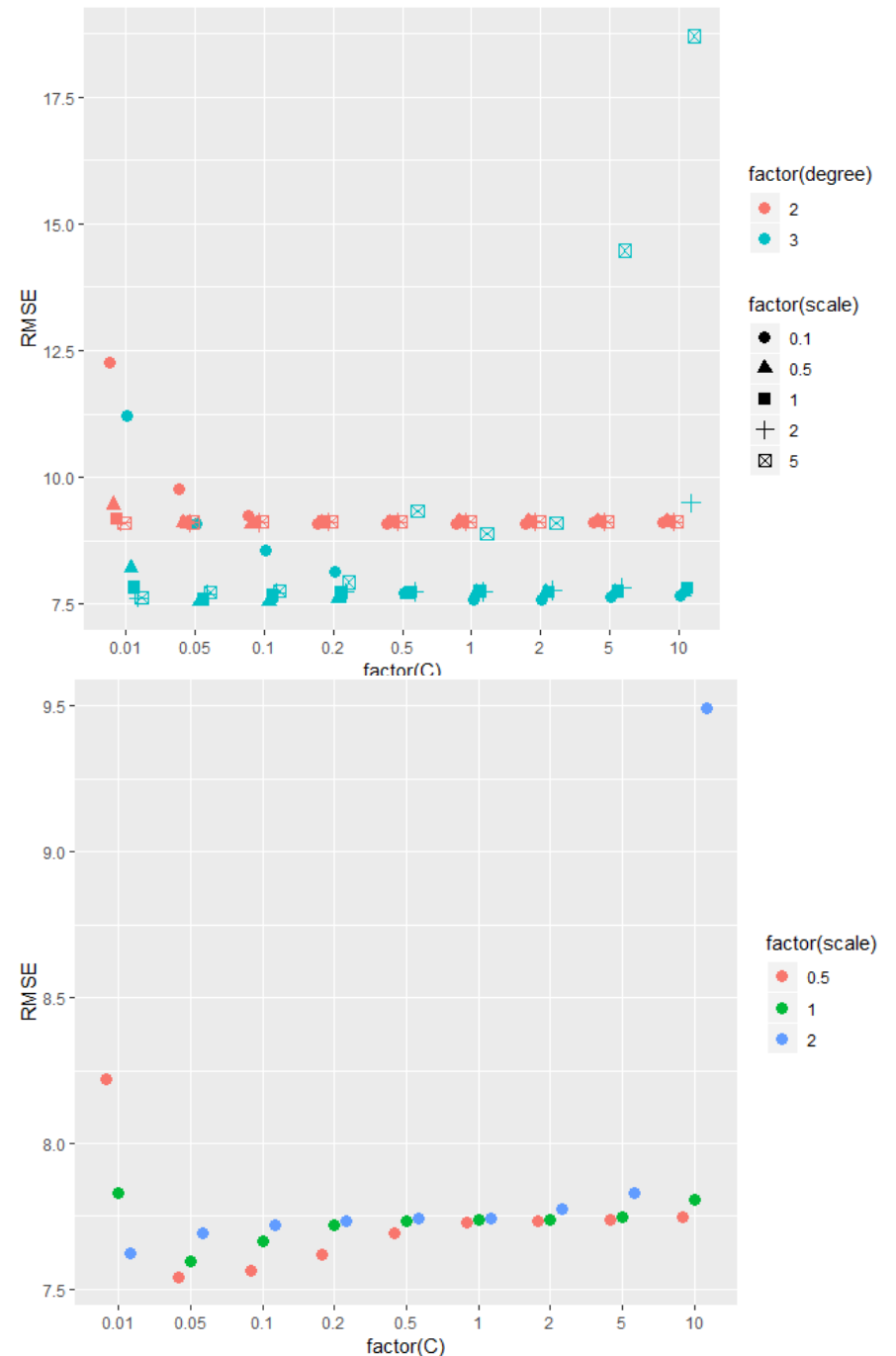
```
# SVM Polinomial: PARÁMETROS C, degree, scale
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10),
  degree=c(2,3),scale=c(0.1,0.5,1,2,5))
control<-trainControl(method = "cv",
  number=4,savePredictions = "all")
(estó tarda mucho)
SVM<- train(data=compressbien,cstrength~age+water+cement+blast,
  method="svmPoly",trControl=control,
  tuneGrid=SVMgrid,verbose=FALSE)
SVM
SVM$results
dat<-as.data.frame(SVM$results)
```

RMSE was used to select the optimal model using the smallest value.  
The final values used for the model were degree = 3, scale = 0.5 and C = 0.05.

```
# PLOT DE DOS VARIABLES CATEGÓRICAS, UNA CONTINUA
ggplot(dat, aes(x=factor(C), y=RMSE,
  color=factor(degree),pch=factor(scale)))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```

```
# AFINO MÁS
dat2<-dat[dat$degree==3&dat$scale<5&dat$scale>0.1,]

ggplot(dat2, aes(x=factor(C), y=RMSE,
  colour=factor(scale)))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```



```
# SVM RBF: PARÁMETROS C, sigma
```

```
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),  
  sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))
```

```
control<-trainControl(method = "cv",  
  number=4,savePredictions = "all")
```

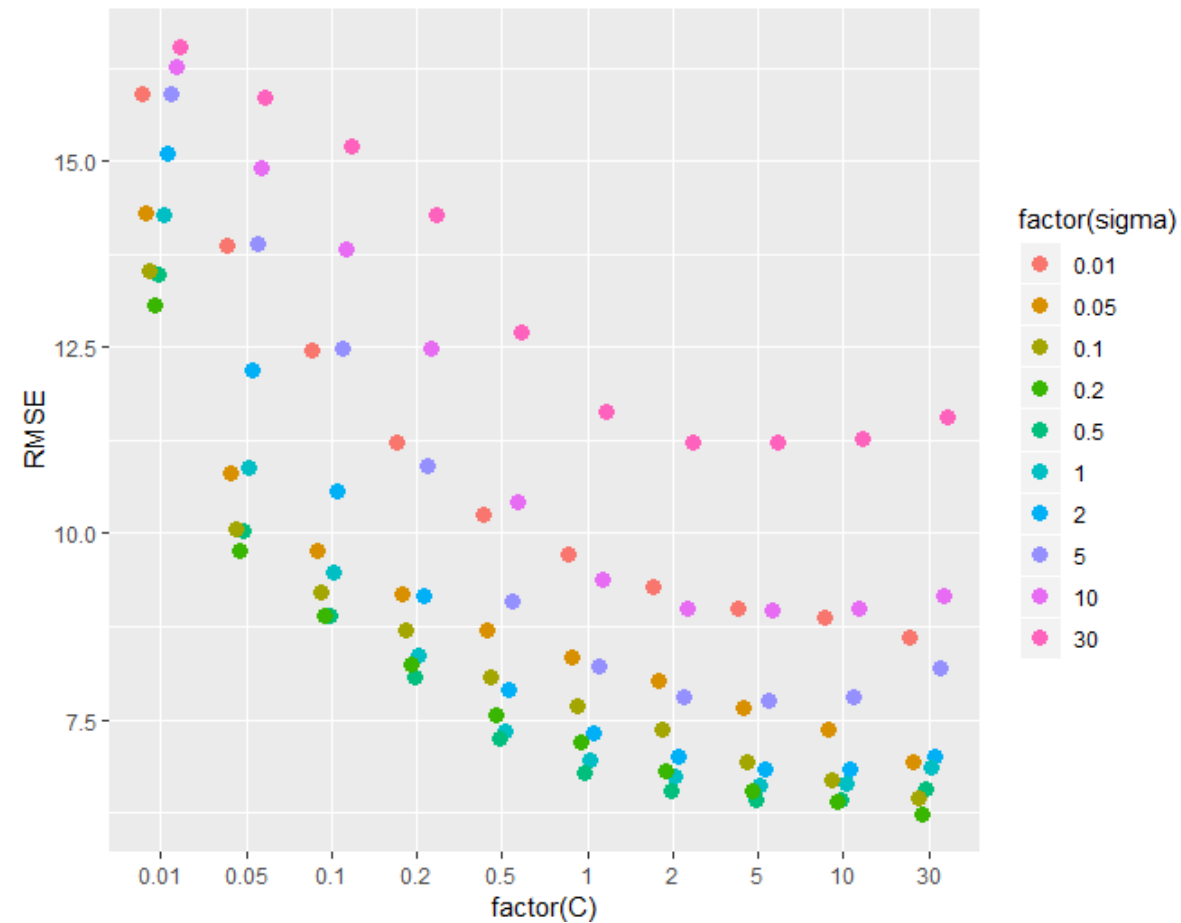
```
SVM<- train(data=compressbien,cstrength~age+water+cement  
+blast, method="svmRadial",trControl=control,  
  tuneGrid=SVMgrid,verbose=FALSE)
```

SVM

RMSE was used to select the optimal model  
using the smallest value.  
The final values used for the  
model were sigma = 0.2 and C = 30.

```
dat<-as.data.frame(SVM$results)
```

```
ggplot(dat, aes(x=factor(C), y=RMSE,  
  color=factor(sigma)))+  
  geom_point(position=position_dodge(width=0.5),size=3)
```



# Comparación con otros modelos vía validación cruzada repetida

...

```
medias8<-cruzadaSVM(data=data,  
  vardep="cstrength",listconti=c("age","water","cement","blast"),  
  listclass=c(""),  
  grupos=4,sinicio=1234,repe=5,C=0.01)
```

```
medias8$modelo="SVM"
```

```
medias9<-cruzadaSVMpoly(data=data,  
  vardep="cstrength",listconti=c("age","water","cement","blast"),  
  listclass=c(""),  
  grupos=4,sinicio=1234,repe=5,C=0.05,degree=3,scale=0.5)
```

```
medias9$modelo="SVMpoly"
```

```
medias10<-cruzadaSVMRBF(data=data,  
  vardep="cstrength",listconti=c("age","water","cement","blast"),  
  listclass=c(""),  
  grupos=4,sinicio=1234,repe=5,C=30,sigma=0.2)
```

```
medias10$modelo="SVMRBF"
```

```
union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias6,  
  medias7,medias8,medias9,medias10)
```

```
par(cex.axis=0.5)
```

```
boxplot(data=union1,error~modelo,col="pink")
```

