

# Clasificador de Máximo Margen

Consideremos un espacio de dimensión  $p$ .

Queremos encontrar el **hiperplano** óptimo para separar dos clases

## Definition

Un hiperplano es un espacio afín de codimensión 1. Se describe mediante una ecuación lineal, no degenerada

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0,$$

es decir que no todos los  $\beta_i = 0$ .

Los hiperplanos parten el espacio en dos "mitades"

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p > 0,$$

y

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p < 0.$$

# Clasificador de Máximo Margen

Supongamos que tenemos vectores  $p$  dimensionales que pertenecen a dos categorías  $\{-1, 1\}$ .

Tenemos una muestra de entrenamiento  $x_1, \dots, x_n$

Supongamos que se puede encontrar un hiperplano que separe las dos clases en forma perfecta.

Del siguiente modo

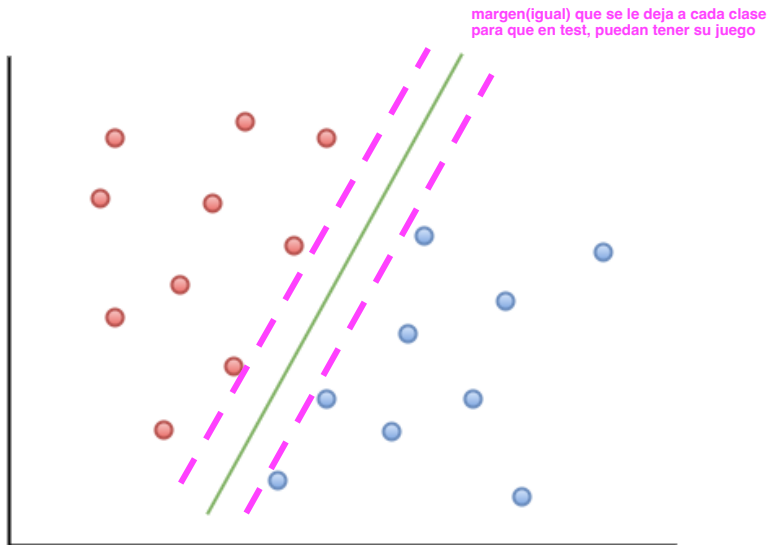
$$\begin{aligned}\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} &> 0 && \text{si } y_i = 1 \\ \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} &< 0 && \text{si } y_i = -1\end{aligned}$$

Luego, el **hiperplano separador** satisface

$$(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) y_i > 0,$$

para  $i = 1, \dots, n$ .

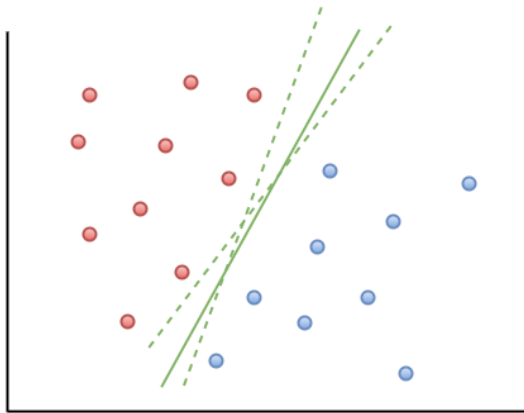
# Clasificador de Máximo Margen



# Clasificador de Máximo Margen

Si el hiperplano existe entonces, sea  $x^*$  una observación en la muestra de clasificación, luego si  $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$  es positivo entonces le asignamos la etiqueta 1 y si es negativo  $-1$ .  
**Información adicional:** si  $|f(x^*)|$  está alejado de cero la clasificación es más certera que si es próximo a cero.

# Clasificador de Máximo Margen



Hay muchos hiperplanos posibles!!! Cuál elegimos?





# Clasificador de Máximo Margen

Sea  $(x_1, y_1), \dots, (x_n, y_n)$  una muestra de entrenamiento, donde  $x_i \in \mathbf{R}^p$  y  $y_i = \pm 1$ .

Solución del problema de optimización

$$\begin{aligned} & \max_{\beta_0, \dots, \beta_p, M} M \\ & \text{Sujeto a que } \sum_{i=1}^p \beta_i^2 = 1, \\ & (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) y_i > M, \text{ para } i = 1, \dots, n. \end{aligned}$$

M: Ancho del margen



# Clasificador de Máximo Margen

- ▶ Si las observaciones están del lado correcto del hiperplano, entonces  $M > 0$ .
- ▶ Podemos juntar las dos restricciones,

$$\frac{1}{\|\beta\|} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) y_i > M.$$

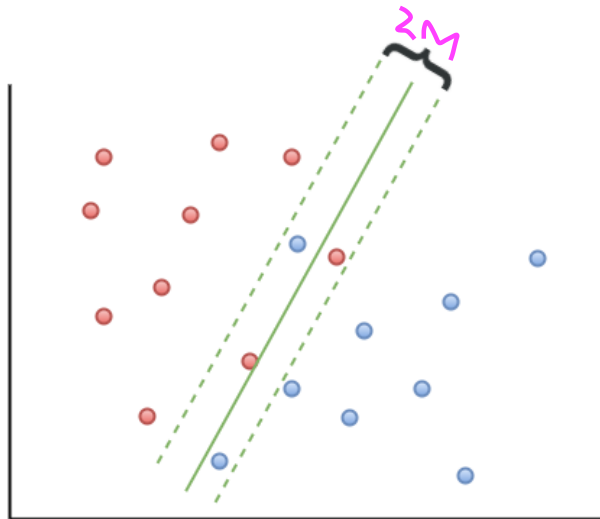
- ▶  $M$  es el margen del hiperplano.
- ▶ Al imponer la restricción  $\|\beta\| = 1$  se puede ver que la distancia perpendicular de la observación  $i$  al hiperplano está dada por

$$(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) y_i$$

Este problema de optimización se puede resolver en forma eficiente.

# Clasificador de Máximo Margen

En muchos casos no existe el hiperplano separador óptimo.



# Support Vector Classifier

Hay casos donde

- ▶ no hay un hiperplano separador.
- ▶ hay un hiperplano separador, pero son muy sensibles a algunas observaciones específicas.
- ▶ agregar una observación provoca grandes cambios en el hiperplano, dejando márgenes muy angostos, *sobreajuste*.

**Idea:** relajar la exigencia de que clasifique a las observaciones de la muestra de entrenamiento en forma perfecta:

- ▶ ganar robustez en relación a observaciones particulares.
- ▶ clasificar bien la *mayoría* de las observaciones.

# Support Vector Classifier

El **Support Vector Classifier** o **Clasificador de Márgenes Suaves** permite clasificar algunas observaciones del lado incorrecto del margen o del hiperplano.

Solución del problema de optimización

variable de holgura de cada observacion

$$\begin{aligned} & \max_{\beta_0, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M} M \\ & \text{Sujeto a que } \sum_{i=1}^p \beta_i^2 = 1, \\ & (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) y_i > M(1 - \epsilon_i), \text{ para } i = 1, \dots, n. \\ & 0 \leq \epsilon_i, \sum_{i=1}^n \epsilon_i \leq C. \end{aligned}$$

Costo maximo dispuesto a aceptar para la holgura.

# Support Vector Classifier

- ▶  $C \geq 0$ .
- ▶  $\epsilon_1, \dots, \epsilon_n$  son las **variables de holgura** o **slack variables**. Dan la posición relativa de  $x_i$  al hiperplano y al margen
  - ▶  $\epsilon_i = 0$  sobre el margen.
  - ▶  $\epsilon_i > 0$  lado incorrecto del margen.
  - ▶  $\epsilon_i > 1$  lado incorrecto del hiperplano.
- ▶  $M$  es el ancho del margen que se busca maximizar.

La clasificación se realiza exactamente igual que en el caso anterior,

$$\begin{aligned} \text{Si } f(x^*) > 0 \quad y^* &= 1 \\ \text{Si } f(x^*) < 0 \quad y^* &= -1 \end{aligned}$$

# Support Vector Classifier

Cómo determinar  $C$ ?

- ▶  $C$  determina la severidad de las violaciones.
- ▶  $C$  se puede ver como un *presupuesto* por el cual las  $n$  observaciones pueden violar el margen y hay que determinar como invertirlo.
- ▶  $C = 0$  es el caso del *hiperplano separador óptimo*.
- ▶  $C > 0$ , a lo sumo  $\lfloor C \rfloor$  observaciones están del lado incorrecto del margen.

**Típicamente  $C$  se determina por cross-validation.**

# Support Vector Classifier

- ▶ El hiperplano separador queda determinado por las observaciones que yacen sobre los márgenes o que los violan, **support vectors**.
- ▶ Relación entre  $C$  y relación *sesgo-varianza*.
  - ▶  $C$  grande  $\leftrightarrow$  muchos support-vectors  $\leftrightarrow$  varianza baja, sesgo alto.
  - ▶  $C$  chico  $\leftrightarrow$  pocos support-vectors  $\leftrightarrow$  varianza alta, sesgo bajo.
- ▶ El hiperplano separador está determinado por unas pocas observaciones, esto lo diferencia de LDA, donde quedaba determinado por las medias y matrices de covarianza estimadas con todo los puntos de la muestra. Más resistente a outliers. (que estan BIEN clasificados)

# Support Vector Machine

Qué hacemos si la **frontera** entre los grupos es **no lineal**?

**Propuesta:** Agrandar el espacio de variables regresoras, *features*, con variables no lineales, agregar términos cuadráticos, cúbicos, de órdenes superiores, interacciones, etc. Considerar una familia de funciones

$$\{h_1(x_i), \dots, h_N(x_i)\}$$


**Problema:** Si el espacio de variables es muy grande, se vuelve computacionalmente inmanejable.



# Support Vector Machine

**Agrandar el espacio de variables para acomodar la frontera no lineal entre las clases.**

Sin entrar en detalles técnicos:

- ▶ La solución del problema de optimización de Support Vector Classifier, depende  $\langle x_i, x_{i'} \rangle$ , no depende de las observaciones  $x_i$ .  
 Producto interno entre observaciones cruzadas, que es aproximadamente, la distancia entre ellas, las observaciones
- ▶ El hiperplano separados se puede expresar,

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x_i, x \rangle$$

con  $\alpha_i \ i = 1, \dots, n$ .

- ▶ Para estimar los parámetros  $\beta_0, \alpha_1, \dots, \alpha_n$  hay que calcular los  $\binom{n}{2}$  productos internos  $\langle x_i, x_{i'} \rangle$ .
- ▶  $\alpha_i \neq 0 \Leftrightarrow x_i$  es un support vector.

# Support Vector Machine

Luego, sea  $\mathcal{S} = \{i : x_i \text{ es support vector}\}$ .

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x_i, x \rangle$$

← Solo sobre las obs que caen en la franja M

que tienen en general mucho menos términos.

En segundo lugar, se reemplaza  $\langle x_i, x_{i'} \rangle$  por  $k(x_i, x_{i'})$  en las estimaciones de  $\alpha_i$  para  $i \in \mathcal{S}$ .

# Support Vector Machine

Kernels usuales,  $k(x_i, x_{i'})$

- ▶ Lineal:  $\sum_{j=1}^p x_{ij}x_{i'j}$ .
- ▶ Polinómico de orden  $d$ :  $\left(1 + \sum_{j=1}^p x_{ij}x_{i'j}\right)^d$ .
- ▶ Radial:  $\exp\left(-\delta \sum_{j=1}^p (x_{ij} - x_{i'j})^2\right)$ , donde  $\delta > 0$ . “mirar localmente”

Una ventaja del kernel radial es que al considerar una nueva observación  $x^*$  le da poca importancia a las observaciones  $x$  que están alejadas de ella, es más local.

# Support Vector Machine

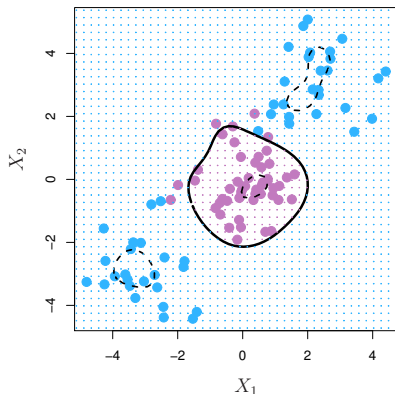
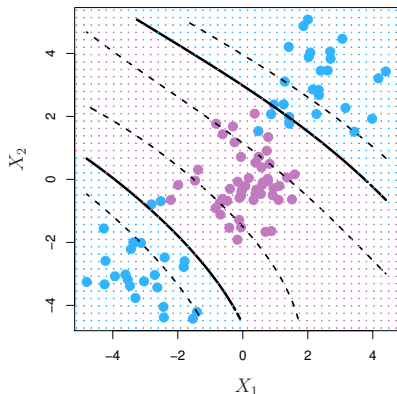
## Clasificador SVM

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i k(x_i, x)$$

### **Ventaja:**

- ▶ Solo depende de calcular  $k(x_i, x_{i'})$
- ▶ Tiene carácter local.

# Support Vector Machine



La figura de la derecha muestra clasificación con un kernel cúbico, mientras que la de la izquierda con un kernel radial.

# Support Vector Machine

Para dar mayor flexibilidad se puede considerar

Clasificador SVM

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i k(h(x_i), h(x))$$

donde  $h$  sean transformaciones no lineales.

# Support Vector Machine

## Ejemplo: Sonar

- ▶ El conjunto de datos busca discriminar señales de un *sonar* provenientes de un metal cilíndrico o de una roca aproximadamente cilíndrica.
- ▶ Cada dato consiste en 60 variables que representan la energía dentro de una banda de frecuencia específica, integrada en un lapso de tiempo.
- ▶ La etiqueta que cada dato trae asociada es "R" si el objeto es una piedra o "M" si es un metal cilíndrico.
- ▶ El conjunto consta de 208 datos, con el objetivo de clasificarlos separaremos la muestra en entrenamiento (75%) y testeo (25%).

# Support Vector Machine

```
library(mlbench)
library(library(e1071))
data("Sonar")
training_sample=sample(c(TRUE, FALSE), nrow(Sonar),
replace = T, prob = c(0.75,0.25))
train=Sonar[training_sample, ]
test =Sonar[!training_sample, ]
```



# Support Vector Machine

Clasificamos, comenzamos con el kernel lineal y cross validamos el costo

```
tune.out = tune(svm, Class~., data = train, kernel =  
"linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1,  
5, 10, 100)))
```

```
bestmod = tune.out$best.model
```

Parameters:

*SVM – Type : C – classification*

*SVM – Kernel : linear*

*cost : 0.01*

*Number of Support Vectors : 100*

# Support Vector Machine

Hacemos las predicciones

```
Class.pred=predict(bestmod,test)
```

```
misclass =table(predict = Class.pred, truth =  
test$Class)
```

	M	R
pred M	23	4
pred R	12	19

# Support Vector Machine

Ahora vamos a clasificar nuevamente usando un nucleo *radial*, para esto no solo tenemos que crossvalidar el costo, también tenemos que crossvalidar el parámetro  $\gamma$ .

```
tune.out = tune(svm, Class~., data = train, kernel =  
"radial", ranges = list(cost = c(0.001, 0.01, 0.1, 1,  
5, 10, 100), gamma = c(0.5,1,2,3,4)))  
bestmod = tune.out$best.model
```

Parameters:

*SVM – Type* : *C – classification*

*SVM – Kernel* : *radial*

*cost* : 5

Number of Support Vectors : 150

# Support Vector Machine

Hacemos las predicciones

```
Class.pred=predict(bestmod,test)
```

```
misclass =table(predict = Class.pred, truth =  
test$Class)
```

	M	R
pred M	35	19
pred R	0	4

# Support Vector Machine

## Qué hacemos si tenemos más de dos grupos?

Asumimos que tenemos  $K$  categorías

- ▶ Dividir el conjunto de datos en  $K(K - 1)/2$  conjuntos, donde cada uno de ellos tenga 2 categorías.
- ▶ Resolver el problema binario de clasificación.
- ▶ Con todas las asignaciones hechas asignar las observaciones a la clase que más votada.
- ▶ Importante: este método no es perfecto, puede tener empates en algunas zonas.

# Support Vector Machine

## Dimensión alta

**Problema:** Datos en dimensión alta, separación en un subespacio de dimensión menor.

Al kernel le costaría buscar sobre todas las dimensiones para encontrar el hiperplano separador.

## $k$ Vecinos más cercanos ( $k$ -NN)

En este caso no vamos a ajustar un modelo.

Tenemos  $(x_1, y_1), \dots, (x_n, y_n)$  una muestra de entrenamiento.

Se quiere clasificar  $x$ .

Se consideran los  $K$  datos más cercanos a  $x$ , en la muestra de entrenamiento

Se lo asigna a la **población más votada** donde votan únicamente los  $k$  vecinos más cercanos.

# $k$ -NN

Necesitamos tener una noción de **cercanía**, necesitamos definir distancias...

Datos reales:

- ▶ Euclidea
- ▶ Mahalanobis
- ▶ Correlación de Pearson.



## Similitudes para datos categóricos (binarios)

Supongamos que tengo dos individuos  $x_i$  y  $x_j$  para los que se observa presencia/ausencia de  $p$  variables binarias.

Observación	$x_1$	$x_2$	$\dots$	$x_p$
$x_i$	0	1	$\dots$	1
$x_j$	1	1	$\dots$	0

- $a$  es el número de veces en las  $p$  variables que ambas observaciones valen 1 al mismo tiempo.
- $b$  es el número de veces en las  $p$  variables una vale 0 y otra vale 1.
- $c$  es el número de veces en las  $p$  variables una vale 1 y otra vale 0.
- $d$  es el número de veces en las  $p$  variables que ambas observaciones valen 0 al mismo tiempo.

- ▶ Coeficiente de concordancia simple

$$s_{ij} = \frac{a + d}{p}.$$

- ▶ Coeficiente de Jaccard

$$s_{ij} = \frac{a}{a + b + c}.$$

# $k$ -NN

## Similitud entre datos **cualitativos**

Para variables cualitativas con más de dos categorías la medida de similitud más utilizada es una generalización del coeficiente de concordancia simple.

$$s_{ij} = \frac{a_1 + \cdots + a_k + d}{p},$$

donde

$a_k$  es el número de veces en que ambas observaciones valen  $k$ ,  
para  $k = 1, \dots, p$ .

$d$  es el número de veces en que ambas observaciones valen 0.

## $k$ -NN

Antes de comenzar el proceso, es conveniente estandarizar las variables, de forma tal que todas tengan media 0 y varianza 1, para evitar problemas de diferentes magnitudes.

Sean  $(X_i, Y_i)_{1 \leq i \leq n}$  muestra de entrenamiento perteneciente a  $M$  categorías, queremos clasificar  $x$

Ordenamos los datos *según* su distancia a  $x$ ,  $(X_{(i)}, Y_{(i)})_{1 \leq i \leq n}$

$$\|x - X_{(1)}\| \leq \|x - X_{(2)}\| \leq \dots \leq \|x - X_{(n)}\|$$

Luego,  $k(x) = \{X_{(i)} : i \leq k\}$ .

## $k$ -NN

La regla de clasificación  $g_n(x)$ ,

$$(g_{n1}(x), \dots, g_{nM}(x)),$$

donde

$$g_{nm}(x) = \frac{\# \{Y_{(i)} = m, i = 1, \dots, k\}}{k}$$

El grupo asignado es

$$\hat{m}(x) = \arg \max \{g_{nm}(x), m = 1, \dots, M\}$$

## Consistencia Universal

Si

▶  $k_n \rightarrow \infty$

▶  $\frac{k_n}{n} \rightarrow 0$

la regla de clasificación es universalmente consistente.

**Cómo se elige  $k_n$  en la práctica?**

Vía **cross-validation**, minimizando el error de predicción.

# $k$ -NN

Este procedimiento suele ser muy útil especialmente cuando las poblaciones no son normales.

1. Definir una distancia entre puntos.
2. Calcular la distancia del punto a clasificar  $\mathbf{x}_0$  a todos los puntos de la muestra de entrenamiento
3. Seleccionar los  $k$  puntos más próximos.
4. Calcular la proporción de estos  $k$  puntos que pertenece a cada población.
5. Clasificar a  $\mathbf{x}_0$  en la población con mayor frecuencia dentro de  $k$ -vecinos más próximos.

# $k$ -NN

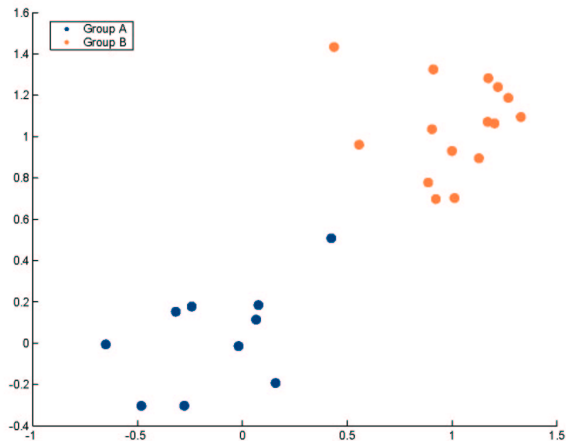


Figure: Consideramos 3 vecinos



# $k$ -NN

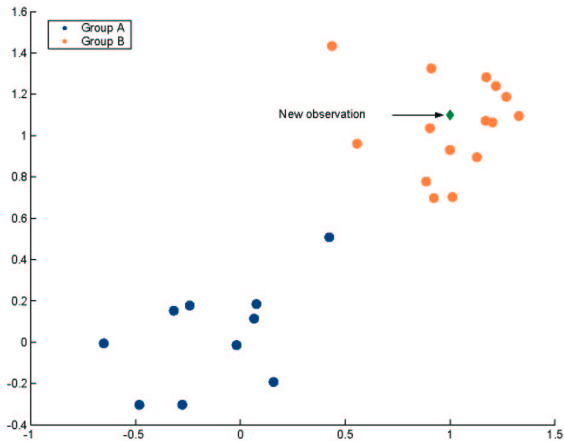


Figure: Consideramos 3 vecinos

# $k$ -NN

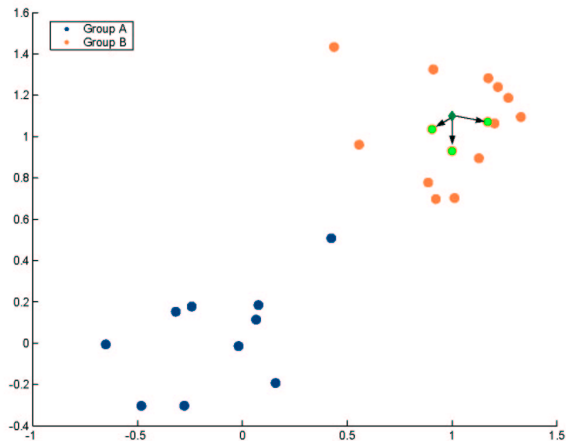
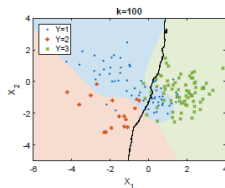
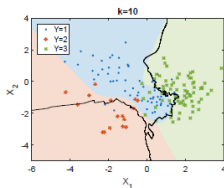
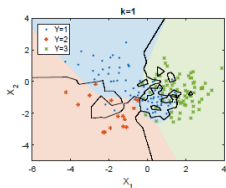


Figure: Consideramos 3 vecinos

# $k$ -NN

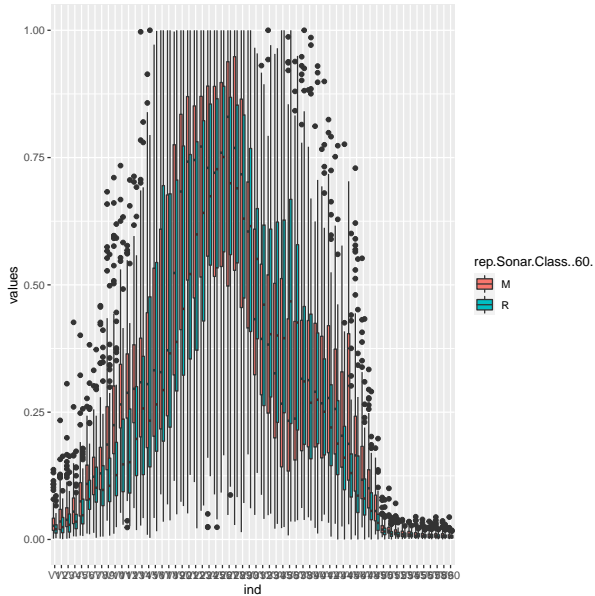
La línea negra, es el clasificador, como divide las categorías



- ▶  $k$  chico  $\Rightarrow$  bajo sesgo, alta varianza.
- ▶  $k$  grande  $\Rightarrow$  alto sesgo, baja varianza.

# k-NN

Volvemos con los datos Sonar



## $k$ -NN

**Aclaración:** en principio este caso no hace falta estandarizar los datos porque están todos en la misma escala. Todas son medidas entre 0.1 y 1, que representan la energía en una banda de frecuencia específica, integradas a lo largo de un lapso de tiempo. Sin embargo vemos que hay diferencias importantes para diferente bandas de frecuencias.

Hacemos 5 fold cv para determinar el número de vecinos más cercanos en la muestra de entrenamiento

```
library(caret)
trControl = trainControl(method = "cv", number = 5)
fit = train(Class ~ ., method = "knn", tuneGrid =
expand.grid(k = 1:10), trControl = trControl, metric
= "Accuracy", data = train)
```

## $k$ -NN

Predecimos con el número óptimo de vecinos que en este caso es 1.

```
pred.knn=knn(train[,1:60],test[,1:60],cl=train$Class,  
k=fit$bestTune)
```

	Pred:M	Pred:R
Actual:M	25	1
Actual:R	5	16

## k-NN

Repetimos el análisis, ahora **estandarizando** (por separado la muestra de entrenamiento y la de validación. Obteniendo los siguientes resultados.

	Pred:M	Pred:R
Actual:M	26	0
Actual:R	2	19

Como vemos sube el accuracy.

## $k$ -NN

Se puede ver que si  $p$  aumenta la tasa de convergencia de  $k$ -NN disminuye. Sufre la *maldición de la dimensionalidad*, al aumentar la dimensión del espacio el volumen crece en forma exponencial, luego los datos pasan a ser esparsos.

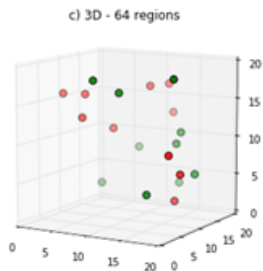
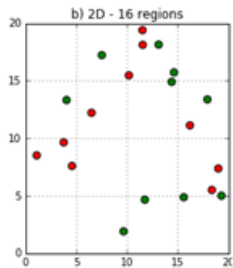
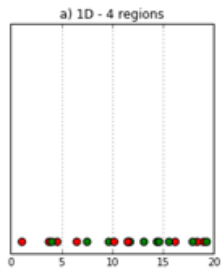
- ▶ Volumen del cubo con aristas  $2r$   $p$  dimensional  $\rightsquigarrow (2r)^p$ .
- ▶ Volumen de la bola de radio  $r$   $p$  dimensional  $\rightsquigarrow \frac{2r^p \pi^{p/2}}{\Gamma(p/2)p}$ .

Luego,

$$\begin{aligned}\frac{\text{VolBola}_r}{\text{VolHipercub}_{2r}} &= \frac{2r^p \pi^{p/2}}{(2r)^p \Gamma(p/2)p} = \\ &= \frac{\pi^{p/2}}{2^{p-1} \Gamma(p/2)p} \xrightarrow[p \rightarrow \infty]{} 0.\end{aligned}$$



# $k$ -NN



## $k$ -NN

Supongamos que tenemos observaciones uniformemente distribuidas en un hipercubo  $p$  dimensional. Dado un punto  $x$  se quiere determinar el entorno de los  $k$  vecinos más cercanos.

La longitud de cada lado esperada es  $k^{1/p}$ . Luego,

Dimensión	Porcentaje de la población	Proporción del área
1	0.01	0.01
	0.10	0.10
2	0.01	$(0.01)^{1/2} = 0.10$
	0.10	$(0.10)^{1/2} = 0.3162$
$\vdots$	$\vdots$	$\vdots$
10	0.01	$(0.01)^{1/10} = 0.6309$
	0.10	$(0.10)^{1/10} = 0.7942$

# $k$ -NN

- ▶ Los datos suelen estar cerca de los bordes, donde es más difícil aprender. La distancia del origen al punto más cercano en dimensión  $p$  considerando que hay  $N$  puntos

$$d(N, p) = \left(1 - \frac{1}{2^{1/N}}\right)^{1/p}$$

- ▶ Los tamaños muestrales necesarios aumentan exponencialmente, la densidad muestral es proporcional a  $N^{1/p}$ .

## $k$ -NN

Cómo solucionarlo?

$k$ -NN asume que la densidad poblacional en entornos de los puntos es aproximadamente constante.

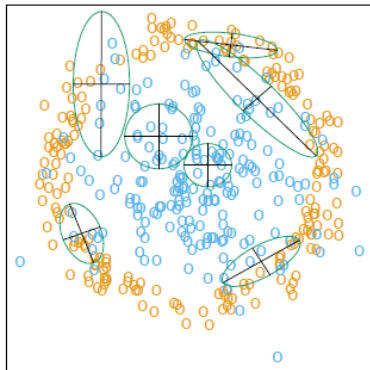
En dimensión alta, típicamente hay direcciones *informativas* y otras *no informativas*. La idea es adaptar las métricas al clasificador NN, de forma tal de alargarlas en las direcciones que añadan varianza y estrecharlas en las direcciones con menor varianza.

Este procedimiento es adaptivo a cada punto del espacio.

Hay una propuesta **DANN**, *discriminant analysis nearest neighbours*, la idea es considerar una distancia de *estilo de Mahalanobis* donde en lugar de considerar la matriz de covarianza se considera una matriz que tiene en cuenta la covarianza de cada clase en el entorno del punto que se quiere clasificar.

Para su implementación ver <https://cran.r-project.org/web/packages/dann/vignettes/dann.html>

# $k$ -NN



**Observación:** procedimientos como  $k$ -NN no suelen mejorar con clasificadores agregados (bagging).

# $k$ -NN




## Ventajas:

- ▶ Funciona bien cuando cada clase tiene diferentes prototipos.
- ▶ Cuando las clases tiene formas irregulares.
- ▶ No hace supuestos sobre las distribuciones de los datos.

## Desventajas:

- ▶ Costo computacional alto, en cómputo y uso de memoria. Se necesitan  $n \times p$  consultas por consulta. Hay algoritmos para bajar el costo computacional y otros para bajar el espacio de guardado (quedarse con datos cercanos a las fronteras.)
- ▶ Se pierde toda posibilidad de interpretar los resultados.
- ▶ Problemas en dimensión alta.



-  T. Hastie, R. Tibshirani, J. Friedman.  
Elements of Statistical Learning, 2nd Ed.  
Springer-Verlag, 2009.  
Capítulo 4 y 12.
-  J. Garret, D. Witten, T. Hastie, R. Tibshirani.  
An Introduction to Statistical Learning.  
Springer-Verlag, 2013.  
Capítulo 9.
-  M. Kuhn, k. Johnson.  
Applied Predicting Modelling.  
Springer-Verlag, 2015.  
Capítulo 13.