Estadística y computación para metagenómica

Victor Muñiz Sánchez

victor_m@cimat.mx

Centro de Investigación en Matemáticas.
Unidad Monterrey.

Junio 2023

Métodos de clasificación

Conceptos de regularización y selección de modelos

Selección de modelos

Selección de modelos

Recuerda (en algún momento lo vimos) que el error esperado de **cualquier** modelo de predicción en una observación de prueba $\mathbf{x}_{\text{new}}$ puede descomponerse mediante:

$$
\begin{aligned}
\text{error}(\mathbf{x}_{\text{new}}) &= E((y - \hat{f}(\mathbf{x}_{\text{new}}))^2 \\
&= \sigma_\epsilon^2 + \text{Bias}^2(\hat{f}(\mathbf{x}_{\text{new}})) + \text{Var}(\hat{f}(\mathbf{x}_{\text{new}})) \\
&= \text{Error} + \text{Bias}^2 + \text{Variance}.
\end{aligned}
$$

Selección de modelos

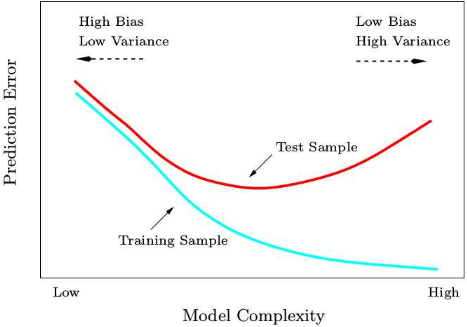
Recuerda (en algún momento lo vimos) que el error esperado de **cualquier** modelo de predicción en una observación de prueba $\mathbf{x}_{\text{new}}$ puede descomponerse mediante:

$$\begin{aligned} \text{error}(\mathbf{x}_{\text{new}}) &= E((y-\hat{f}(\mathbf{x}_{\text{new}}))^2 \\ &= \sigma_\epsilon^2 + \text{Bias}^2(\hat{f}(\mathbf{x}_{\text{new}})) + \text{Var}(\hat{f}(\mathbf{x}_{\text{new}})) \\ &= \text{Error} + \text{Bias}^2 + \text{Variance}. \end{aligned}$$

En base a estos conceptos, ¿Cómo podemos escoger el modelo "adecuado"?

Selección de modelos

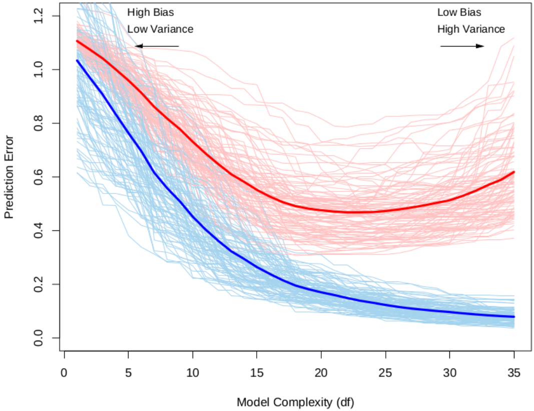
Bias-Variance tradeoff



Hastie, et al. 2nd. Ed.

Selección de modelos

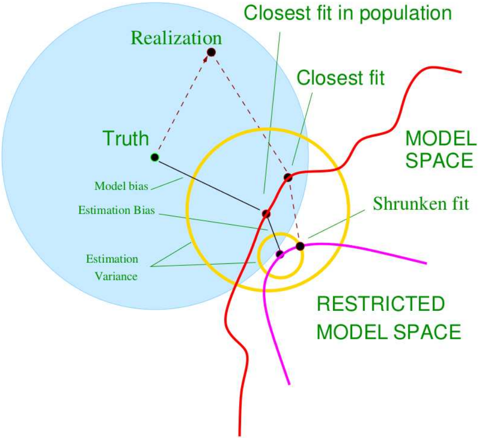
Bias-Variance tradeoff



Hastie, et al. 2nd. Ed.

Selección de modelos

Bias-Variance tradeoff esquemáticamente



Hastie, et al. 2nd. Ed.

Selección de modelos

Nuestros objetivos.

- Selección del modelo: estimación del desempeño de diferentes modelos para escoger el mejor.
- Evaluación del modelo: una vez escogido un modelo (final), estimar su error de predicción (error de generalización) en un nuevo conjunto de datos.

Selección de modelos

Nuestros objetivos.

- Selección del modelo: estimación del desempeño de diferentes modelos para escoger el mejor.
- Evaluación del modelo: una vez escogido un modelo (final), estimar su error de predicción (error de generalización) en un nuevo conjunto de datos.

Teniendo una suficiente cantidad de datos, podriamos alcanzar ambos objetivos dividiendo nuestro conjunto de datos en:

1. Entrenamiento (ajustar el modelo)
2. Validación (error de predicción del modelo ajustado)
3. Prueba (error de generalización del modelo final)

Sin embargo, muchas veces no tenemos suficiente cantidad de datos.

Selección de modelos

¿Qué necesitamos para escoger nuestro modelo?

- Una medida de su complejidad
- Una medida del error de generalización asociado

Hay varias formas para escoger el mejor modelo:

Selección de modelos

¿Qué necesitamos para escoger nuestro modelo?

- Una medida de su complejidad
- Una medida del error de generalización asociado

Hay varias formas para escoger el mejor modelo:

- AIC y BIC (Akaike Information Criterion, Bayesian IC). Cuando usamos log-verosimilitudes como función de costo

Selección de modelos

¿Qué necesitamos para escoger nuestro modelo?

- Una medida de su complejidad
- Una medida del error de generalización asociado

Hay varias formas para escoger el mejor modelo:

- AIC y BIC (Akaike Information Criterion, Bayesian IC). Cuando usamos log-verosimilitudes como función de costo
- Dimensión VC: *Dada una clase de funciones $\{f(\mathbf{x}, \theta)\}$, la dimensión VC se define como el número más grande de puntos (en alguna configuración) que pueden ser separados (shattered) por miembros de $\{f(\mathbf{x}, \theta)\}$* .

  Por ejemplo, la familia de clasificadores lineales tiene dimensión VC igual a $d + 1$, con d la dimensión de los datos.

  A partir de la dimensión VC, pueden definirse cotas de error de predicción, y entonces, se elige el modelo con la menor cota.

Selección de modelos

¿Qué necesitamos para escoger nuestro modelo?

- Una medida de su complejidad
- Una medida del error de generalización asociado

Hay varias formas para escoger el mejor modelo:

- AIC y BIC (Akaike Information Criterion, Bayesian IC). Cuando usamos log-verosimilitudes como función de costo
- Dimensión VC: *Dada una clase de funciones* $\{f(\mathbf{x}, \theta)\}$*, la dimensión VC se define como el número más grande de puntos (en alguna configuración) que pueden ser separados (shattered) por miembros de* $\{f(\mathbf{x}, \theta)\}$.

  Por ejemplo, la familia de clasificadores lineales tiene dimensión VC igual a $d+1$, con d la dimensión de los datos.

  A partir de la dimensión VC, pueden definirse cotas de error de predicción, y entonces, se elige el modelo con la menor cota.
- Validación Cruzada (CV).

Selección de modelos

En términos prácticos, la opción más usada es sin duda, Validación Cruzada (K-Fold CV). Con esto, se puede asegurar (al menos), de que todos los datos son usados tanto para el ajuste como para la estimación de los modelos.

Ejemplo: 5-Fold CV

1	2	3	4	5
Train	Train	Validation	Train	Train

Selección de modelos

Sea $f^{-K(i)}(\mathbf{x}_i, \theta)$ el modelo ajustado con el $K(i)$ conjunto de datos (Fold) removido.

El error de predicción estimado está dado por:

$$\text{Err}_{CV}(f, \theta) = \frac{1}{n} \sum_{i=1}^{n} L(y_i - f^{-K(i)}(\mathbf{x}_i, \theta))$$

Y escogemos el modelo con el error mínimo

Máquinas de soporte vectorial

Máquinas de Soporte Vectorial

Los dos artículos fundamentales.

A Training Algorithm for Optimal Margin Classifiers

Bernhard E. Boser*
EECS Department
University of California
Berkeley, CA 94720
boser@eecs.berkeley.edu

Isabelle M. Guyon
AT&T Bell Laboratories
50 Fremont Street, 6th Floor
San Francisco, CA 94105
isabelle@neural.att.com

Vladimir N. Vapnik
AT&T Bell Laboratories
Crawford Corner Road
Holmdel, NJ 07733
vlad@neural.att.com

Machine Learning, 20, 273–297 (1995)

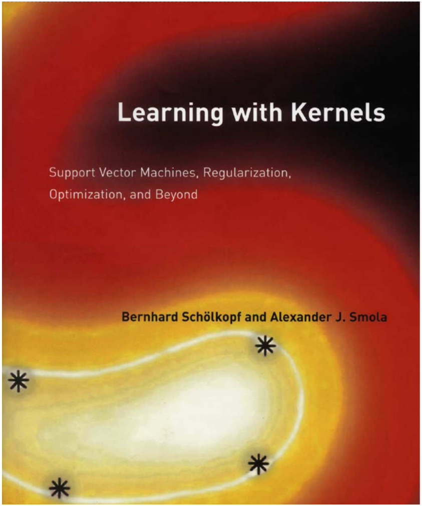
Support-Vector Networks

CORINNA CORTES
VLADIMIR VAPNIK
AT&T Bell Labs., Holmdel, NJ 07733, USA

corinna@neural.att.com
vlad@neural.att.com

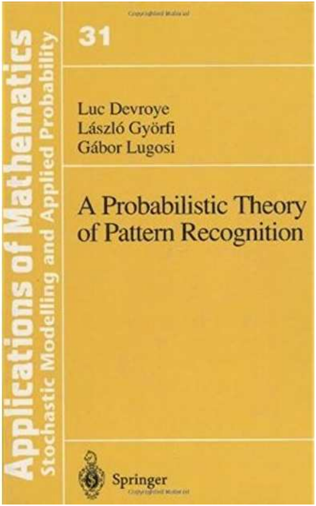
Máquinas de Soporte Vectorial

También



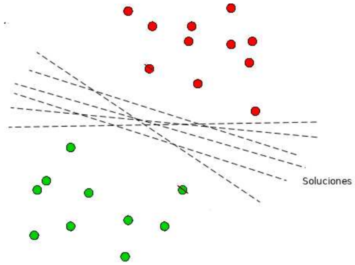
Máquinas de Soporte Vectorial

Y si quieres, también



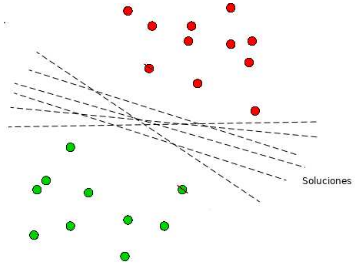
Máquinas de Soporte Vectorial

Recuerda el problema que teníamos anteriormente con clasificadores lineales



Máquinas de Soporte Vectorial

Recuerda el problema que teníamos anteriormente con clasificadores lineales



¿Cómo tener una solución única (óptima)?
Debemos definir un criterio.
Para SVM, éste criterio está dado por la solución óptima que involucra la distancia entre el hiperplano obtenido y un **subconjunto** de los datos de entrenamiento, llamados **vectores soporte**.

Máquinas de Soporte Vectorial

Veamos el caso más sencillo: dos categorías separables linealmente.

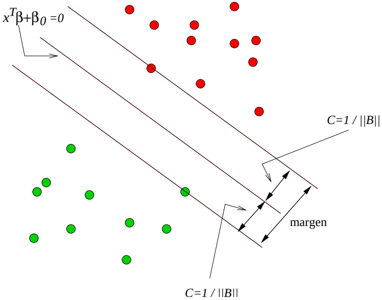
Sean $\{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathbb{R}^d$ y $y \in \{-1, 1\}$, nuestros datos de entrenamiento.
Define un hiperplano separador

$$f(\mathbf{x}) = \beta_0 + \mathbf{x}'\boldsymbol{\beta} = 0.$$

Sea C_+ la distancia más corta de $f(\mathbf{x})$ al punto con clase 1 $(\mathbf{x}_+)$ más cercano.
Sea C_- la distancia más corta de $f(\mathbf{x})$ al punto con clase -1 $(\mathbf{x}_-)$ más cercano.
Definimos el **márgen** como

$$C_+ + C_-$$

Máquinas de Soporte Vectorial



Observa que:

$$\begin{aligned} \beta_0 + \mathbf{x}_i'\boldsymbol{\beta} &\geq 1 \quad \text{si } y_i = 1 \\ \beta_0 + \mathbf{x}_i'\boldsymbol{\beta} &\leq -1 \quad \text{si } y_i = -1. \end{aligned}$$

Por lo tanto,

$$\beta_0 + \mathbf{x}_+'\boldsymbol{\beta} = 1 \tag{1}$$
$$\beta_0 + \mathbf{x}_-'\boldsymbol{\beta} = -1 \tag{2}$$

para los puntos $\mathbf{x}_+$ y $\mathbf{x}_-$ que están en los hiperplanos H_+ y H_-, respectívamente.

Máquinas de Soporte Vectorial

La diferencia es

$$\mathbf{x}_+'\boldsymbol{\beta} - \mathbf{x}_-'\boldsymbol{\beta} = 2,$$

y la suma

$$\beta_0 = -\frac{1}{2}(\mathbf{x}_+'\boldsymbol{\beta} + \mathbf{x}_-'\boldsymbol{\beta}).$$

También:

$$\begin{aligned} C_+ &= \frac{|\beta_0 + \mathbf{x}_+'\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} &= \frac{1}{\|\boldsymbol{\beta}\|} \\ C_- &= \frac{|\beta_0 + \mathbf{x}_-'\boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} &= \frac{1}{\|\boldsymbol{\beta}\|}. \end{aligned}$$

Por lo tanto, el márgen es

$$M = \frac{2}{\|\boldsymbol{\beta}\|}.$$

Máquinas de Soporte Vectorial

Ahora, observa que, según (1) y (2), decimos que $\mathbf{x}_i$ **es un vector de soporte** si

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) = 1,$$

entonces, lo que queremos es:

$$\max_{\beta_0, \boldsymbol{\beta}, \|\boldsymbol{\beta}\|=1} M,$$

sujeto a

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq M, \quad i = 1, 2, \ldots, n.$$

La restricción en $\boldsymbol{\beta}$ es para que no crezca arbitrariamente.

Máquinas de Soporte Vectorial

Ahora, observa que, según (1) y (2), decimos que $\mathbf{x}_i$ **es un vector de soporte** si

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) = 1,$$

entonces, lo que queremos es:

$$\max_{\beta_0, \boldsymbol{\beta}, \|\boldsymbol{\beta}\|=1} M,$$

sujeto a

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq M, \quad i = 1, 2, \ldots, n.$$

La restricción en $\boldsymbol{\beta}$ es para que no crezca arbitrariamente. Otra forma de considerar ésta restricción es modificando las condiciones:

$$\frac{1}{\|\boldsymbol{\beta}\|} y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq M$$

Máquinas de Soporte Vectorial

O también:

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq \|\boldsymbol{\beta}\|M,$$

y ésta desigualdad se cumple para cualquier escalamiento positivo de $\boldsymbol{\beta}$.
Podemos definir arbitrariamente

$$\|\boldsymbol{\beta}\| = \frac{1}{M},$$

que es un valor positivo.

Máquinas de Soporte Vectorial

O también:

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq \|\boldsymbol{\beta}\|M,$$

y ésta desigualdad se cumple para cualquier escalamiento positivo de $\boldsymbol{\beta}$.
Podemos definir arbitrariamente

$$\|\boldsymbol{\beta}\| = \frac{1}{M},$$

que es un valor positivo.
Observa que, si M es grande, $\|\boldsymbol{\beta}\|$ será pequeño.

Máquinas de Soporte Vectorial

Entonces, podemos redefinir el problema de optimización como

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{\beta}\|^2$$

s.a.

$$y_i(\beta_0 + \mathbf{x}_i' \boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \ldots, n.$$

Máquinas de Soporte Vectorial

Entonces, podemos redefinir el problema de optimización como

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2$$

s.a.

$$y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \ldots, n.$$

Tenemos entonces un problema de optimización muy agradable:

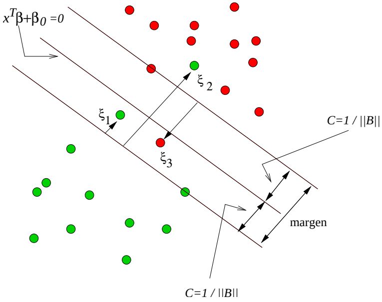
- cuadrático, por lo tanto es convexo y solución única
- restricciones lineales
- hay métodos muy eficientes para resolverlo.

(ver notas a mano...)

Máquinas de Soporte Vectorial

Caso no separable (soft margin).

Aquí, permitimos que cada dato pueda aparecer en el lado "equivocado" del hiperplano separador.



Máquinas de Soporte Vectorial

Definimos las variables de holgura

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)' \geq 0.$$

El problema de optimización es

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2$$

s.a.

$$\begin{aligned} y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) + \xi_i &\geq 1 \\ \xi_i &\geq 0 \\ \sum \xi_i &\leq \text{constante}, \end{aligned}$$

para $i = 1, 2, \ldots, n$.

Máquinas de Soporte Vectorial

El "soft margin" lo formulamos como un problema de regularización sobre los pesos $\boldsymbol{\xi}$ del margen:$\lambda\|\boldsymbol{\xi}\|_L$, $\lambda \geq 0$.

Máquinas de Soporte Vectorial

El "soft margin" lo formulamos como un problema de regularización sobre los pesos $\boldsymbol{\xi}$ del margen:$\lambda\|\boldsymbol{\xi}\|_L$, $\lambda \geq 0$. Generalmente usamos la norma L_1 para "activar" o permitir que solo algunos datos estén del lado incorrecto de $f(\mathbf{x})$. Entonces, el problema de optimización queda:

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$

s.a.

$$\begin{array}{rcl} y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) & \geq & 1 - \xi_i \\ \xi_i & \geq & 0, \end{array}$$

para $i = 1, 2, \ldots, n$.

Máquinas de Soporte Vectorial

El Lagrangiano primal es

$$\begin{aligned} L_P &= \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\ &\quad - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \right) - \sum_{i=1}^{n} \eta_i \xi_i, \end{aligned}$$

con $\boldsymbol{\alpha}, \boldsymbol{\eta} \geq 0$ los multiplicadores de Lagrange.

Máquinas de Soporte Vectorial

El Lagrangiano primal es

$$
\begin{aligned}
L_P &= \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\
&\quad - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \right) - \sum_{i=1}^{n} \eta_i \xi_i,
\end{aligned}
$$

con $\boldsymbol{\alpha}, \boldsymbol{\eta} \geq 0$ los multiplicadores de Lagrange.

Sus derivadas:

$$
\begin{aligned}
\frac{\partial L_P}{\partial \beta_0} &= \sum_{i=1}^{n} \alpha_i y_i \\
\frac{\partial L_P}{\partial \boldsymbol{\beta}} &= \boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\
\frac{\partial L_P}{\partial \xi_i} &= \lambda - \alpha_i - \eta_i
\end{aligned}
$$

Máquinas de Soporte Vectorial

Resolviendo:

$$
\begin{aligned}
\sum_{i=1}^{n} \alpha_i y_i &= 0 \\
\boldsymbol{\beta}^* &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\
\alpha_i &= \lambda - \eta_i
\end{aligned}
$$

Máquinas de Soporte Vectorial

Resolviendo:

$$
\begin{aligned}
\sum_{i=1}^{n} \alpha_i y_i &= 0 \\
\boldsymbol{\beta}^* &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\
\alpha_i &= \lambda - \eta_i
\end{aligned}
$$

Como en el caso separable, sustituimos éstos valores en el Lagrangiano primal, y luego de un poco de álgebra, obtenemos su correspondiente problema dual:

$$
L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i' \mathbf{x}_j)
$$

Máquinas de Soporte Vectorial

Resolviendo:

$$\begin{aligned}\sum_{i=1}^{n} \alpha_i y_i &= 0 \\ \boldsymbol{\beta}^* &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ \alpha_i &= \lambda - \eta_i\end{aligned}$$

Como en el caso separable, sustituimos éstos valores en el Lagrangiano primal, y luego de un poco de álgebra, obtenemos su correspondiente problema dual:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i' \mathbf{x}_j)$$

¡Que es el mismo que el caso separable!

Máquinas de Soporte Vectorial

Como las restricciones requieren que $\lambda - \alpha_i - \eta_i = 0$ y $\eta_i \geq 0$, tenemos que, $0 \leq \alpha_i \leq \lambda$.

Máquinas de Soporte Vectorial

Como las restricciones requieren que $\lambda - \alpha_i - \eta_i = 0$ y $\eta_i \geq 0$, tenemos que, $0 \leq \alpha_i \leq \lambda$.
Las condiciones KKT para éste caso son:

$$\begin{aligned}
y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) &\geq 0 && (3)\\
\xi_i &\geq 0 && (4)\\
\alpha_i &\geq 0 && (5)\\
\eta_i &\geq 0 && (6)\\
\alpha_i\left(y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i)\right) &= 0 && (7)\\
\xi_i(\alpha_i - \lambda) &= 0. && (8)
\end{aligned}$$

Máquinas de Soporte Vectorial

Como las restricciones requieren que $\lambda - \alpha_i - \eta_i = 0$ y $\eta_i \geq 0$, tenemos que, $0 \leq \alpha_i \leq \lambda$.
Las condiciones KKT para éste caso son:

$$
\begin{aligned}
y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) &\geq 0 && (3)\\
\xi_i &\geq 0 && (4)\\
\alpha_i &\geq 0 && (5)\\
\eta_i &\geq 0 && (6)\\
\alpha_i \left(y_i(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0) - (1 - \xi_i)\right) &= 0 && (7)\\
\xi_i(\alpha_i - \lambda) &= 0. && (8)
\end{aligned}
$$

Por las condiciones de complementariedad (7) y (8), tenemos que, la variable de holgura ξ_i será > 0 solo si $\alpha_i = \lambda$.

Máquinas de Soporte Vectorial

El problema de optimización dual es entonces

$$\max \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}'\mathbf{M}\boldsymbol{\alpha}, \tag{9}$$

s.a.

$$\boldsymbol{\alpha}'\mathbf{y} = 0, \qquad 0 \leq \boldsymbol{\alpha} \leq \lambda\mathbf{1},$$

con $\mathbf{M}$ una matriz cuadrada, simétrica, semidefinida positiva y entradas $M_{ij} = y_i y_j (\mathbf{x}_i'\mathbf{x}_j)$.

Máquinas de Soporte Vectorial

En el óptimo,

$$\boldsymbol{\beta}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i,$$

pero, por las condiciones de complementariedad (7) y (8):

$$\boldsymbol{\beta}^* = \sum_{i \in SV} \alpha_i^* y_i \mathbf{x}_i,$$

donde SV es el conjunto de vectores soporte, es decir, aquellos donde

$$0 < \alpha_i^* \quad \text{y} \quad \xi_i = 0.$$

Generalmente, usamos un promedio de los α_i^* que son SV.

Máquinas de Soporte Vectorial

También, obtenemos

$$\beta_0^* = -\frac{1}{2}\left(\mathbf{x}_+'\boldsymbol{\beta}^* + \mathbf{x}_-'\boldsymbol{\beta}^*\right),$$

con $\mathbf{x}_+$ y $\mathbf{x}_-$ cualesquiera vectores de soporte.

Máquinas de Soporte Vectorial

También, obtenemos

$$\beta_0^* = -\frac{1}{2}\left(\mathbf{x}_+'\boldsymbol{\beta}^* + \mathbf{x}_-'\boldsymbol{\beta}^*\right),$$

con $\mathbf{x}_+$ y $\mathbf{x}_-$ cualesquiera vectores de soporte.

Finalmente, nuestro clasificador queda entonces como

$$f^*(\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}^* + \beta_0^*,$$

y la función de decisión será:

$$G(\mathbf{x}) = \text{signo}(f^*(\mathbf{x})).$$

Máquinas de Soporte Vectorial

SVM no lineal.

Considera nuevamente el problema de optimización para el caso general (9).
Para generar fronteras de clasificación no-lineales, se utiliza el truco del kernel, considerando que la formulación de la solución de SVM está dada en términos de productos punto.
En este caso:

$$\begin{aligned} M_{ij} &= y_i y_j (\mathbf{x}_i' \mathbf{x}_j) \\ &= y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \\ &= y_i y_j k(\mathbf{x}_i' \mathbf{x}_j), \end{aligned}$$

y la solución está dada por:

$$\begin{aligned} f^*(\mathbf{x}) &= \boldsymbol{\beta}^{*T} \mathbf{x} + \beta_0^* \\ &= \sum_{i \in SV} \alpha_i^* y_i k(\mathbf{x}, \mathbf{x}_i) + \beta_0^*, \end{aligned}$$

con $k(\cdot, \cdot)$, un kernel válido.

Máquinas de Soporte Vectorial

SVM multiclase.

Generalmente hay dos opciones:

- One vs All.
  Resuelve K problemas binarios de clasificación, y toma $\hat{y} = f_k(\mathbf{x})$ con el valor más grande positivo, donde $f_k(\mathbf{x})$ es la solución óptima para el problema de clasificación binaria de la clase k contra el resto.
- One vs One.
  Construye $\binom{K}{2}$ clasificadores binarios, y $f_k(\mathbf{x})$ es la solución k que recibe más votaciones.

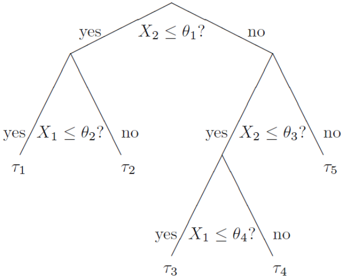
Máquinas de Soporte Vectorial

`SVM.ipynb`

Arboles de Clasificación y regresión

Arboles de clasificación

- Es un método de clasificación y regresión no paramétrico.
- Es el resultado de preguntar una secuencia ordenada de preguntas
- El tipo de pregunta que se contesta en cada paso de la secuencia, depende de las respuestas de las preguntas previas en la secuencia
- La secuencia termina en la predicción de la clase.



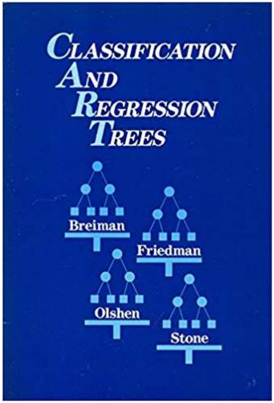
Arboles de clasificación

Algoritmos más conocidos relacionados con árboles:

- CART, Breiman et al. 1984
- ID3, Quinlan, 1986
- C4.5, Quinlan, 1993
- Bayesian CART, Chipman et al. 1998
- Random Forest, Breiman, 2001
- BART, Chipman et al. 2010

Arboles de clasificación

Nosotros, nos enfocaremos en CART (Breiman, Friedman, Olshen, Stone, 1984)



Arboles de clasificación

CART:

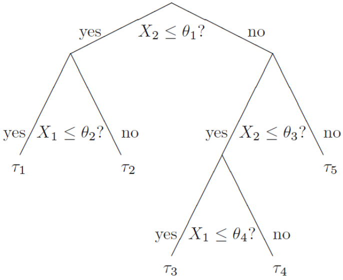
- Sin supuestos distribucionales, pero cuenta con una sólida justificación teórica
- Maneja mezclas de variables (contínuas, categóricas, etc...)
- Un buen modelo explicativo, al menos de inicio
- Puede afrontar problemas de (relativa) alta dimensionalidad, aunque computacionalmente puede ser costoso
- Incluye un mecanismo de regularización a través del procedimiento de "crecimiento" y "poda" del árbol
- Permite datos pesados, aprioris $P(y)$ desiguales y **costos diferentes** de mala clasificación.

Arboles de clasificación

Consideremos un conjunto de n datos de entrenamiento:

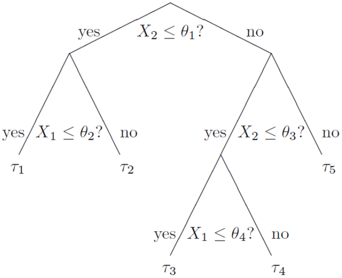
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n); \quad \mathbf{x} \in \mathcal{R}^d, y \in \{-1, 1\}$$

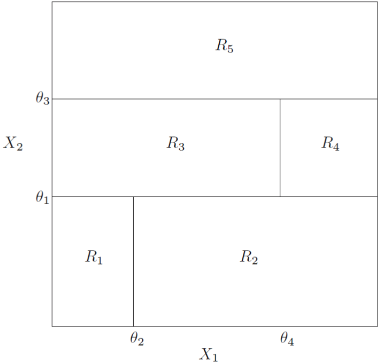
- Un **Nodo** es un subconjunto del conjunto de variables
- Un **Nodo no terminal** (o nodo padre) es un nodo que se divide en dos nodos hijos.
- Un **Nodo terminal** es un nodo que no se divide y asigna la clase a un objeto.
- Puede haber más de un nodo terminal con la misma clase. Por ejemplo, $\tau_1 = \tau_4 = 1$ y $\tau_2 = \tau_2 = \tau_5 = -1$



Arboles de clasificación

Geométricamente, el espacio de entrada $\mathcal{R}^d$ se particiona en un número de rectángulos $(d = 2)$ o hipercubos $(d > 2)$ sin superponerse.

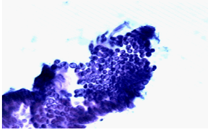
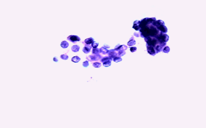
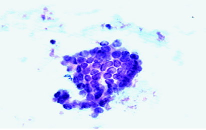




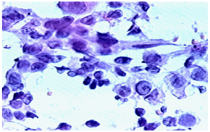
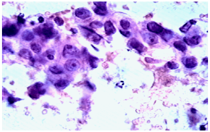
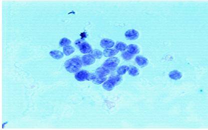
Arboles de clasificación

Ejemplo. Breast Cancer Wisconsin (Diagnostic) Data Set.

Benigno

Maligno

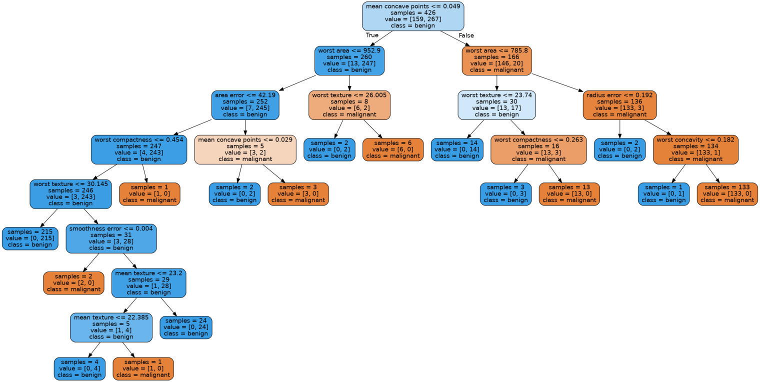
Arboles de clasificación

Ejemplo. Breast Cancer Wisconsin (Diagnostic) Data Set.

```
Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------
**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, field
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign
```

Arboles de clasificación

Ejemplo. Breast Cancer Wisconsin (Diagnostic) Data Set.



Arboles de clasificación

Cómo construir el árbol?

Básicamente se tienen que resolver estos puntos:

1. Escoger las condiciones booleanas para dividir cada nodo
2. Elegir el criterio para dividir el nodo padre en sus dos nodos hijos
3. Decidir si un nodo se convierte en nodo terminal
4. Asignar la clase a los nodos terminales

Arboles de clasificación

Estrategias para dividir un nodo:

- En cada nodo, se debe decidir cuál variable es **la mejor** para realizar el split. En esta metodología, se consideran todos los posibles splits sobre todas las variables presentes en el nodo

Arboles de clasificación

Estrategias para dividir un nodo:

- En cada nodo, se debe decidir cuál variable es **la mejor** para realizar el split. En esta metodología, se consideran todos los posibles splits sobre todas las variables presentes en el nodo
- Para **variables contínuas**: el número de splits posibles es el número de valores observados menos uno.

Arboles de clasificación

Estrategias para dividir un nodo:

- En cada nodo, se debe decidir cuál variable es **la mejor** para realizar el split. En esta metodología, se consideran todos los posibles splits sobre todas las variables presentes en el nodo
- Para **variables contínuas**: el número de splits posibles es el número de valores observados menos uno.
- Para **variables categóricas** con M categorías: el número de splits posibles es $2^{M-1} - 1$

Arboles de clasificación

Estrategias para dividir un nodo:

- En cada nodo, se debe decidir cuál variable es **la mejor** para realizar el split. En esta metodología, se consideran todos los posibles splits sobre todas las variables presentes en el nodo
- Para **variables contínuas**: el número de splits posibles es el número de valores observados menos uno.
- Para **variables categóricas** con M categorías: el número de splits posibles es $2^{M-1} - 1$
- Para algún nodo, definimos r_i como el número de splits posibles para una variable contínua u ordinal x_i, y s_j como el número de splits para una variable categórica x_j, entonces el número de splits posibles en un nodo es $\sum_i r_i + \sum_j s_j$.

Arboles de clasificación

Estrategias para dividir un nodo:

- En cada nodo, se debe decidir cuál variable es **la mejor** para realizar el split. En esta metodología, se consideran todos los posibles splits sobre todas las variables presentes en el nodo
- Para **variables contínuas**: el número de splits posibles es el número de valores observados menos uno.
- Para **variables categóricas** con M categorías: el número de splits posibles es $2^{M-1}-1$
- Para algún nodo, definimos r_i como el número de splits posibles para una variable contínua u ordinal x_i, y s_j como el número de splits para una variable categórica x_j, entonces el número de splits posibles en un nodo es $\sum_i r_i + \sum_j s_j$.

  Por ejemplo, para el primer nodo del Cleveland heart disease dataset, hay 391 posibles splits.

  ¿Cuál es el mejor?

Arboles de clasificación

Estrategias para dividir un nodo:

- Función de impureza de un nodo. Para algún nodo τ y $k = 1, 2, \ldots, K$ clases:

$$i(\tau) = \phi(p(y = 1|\tau), p(y = 2|\tau), \ldots, p(y = K|\tau)).$$

  Requerimos que esta función sea
  - simétrica
  - definida para todas las probabilidades $p(k|\tau)$
  - sume 1
  - minimizada en $(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$
  - maximizada en $(1/K, \ldots, 1/K)$

Arboles de clasificación

Estrategias para dividir un nodo.

Hay varias funciones que cumplen con esos requisitos, pero las dos opciones más comúnes son la función de entropía:

$$i(\tau) = -\sum_{k=1}^{K} P(k|\tau) \log P(k|\tau),$$

que para 2 clases, $\{-1, 1\}$, se reduce a

$$i(\tau) = -\left(P(y=-1|\tau) \log P(y=-1|\tau) + P(y=1|\tau) \log P(y=1|\tau)\right).$$

Arboles de clasificación

Estrategias para dividir un nodo.

La otra función es el índice de diversidad de Gini:

$$-\sum_{k\neq k'}^{K} P(k|\tau)P(k'|\tau) = 1 - \sum_{k}[P(k|\tau)]^2,$$

de igual forma para el caso de clasificación binaria, Gini se reduce a

$$i(\tau) = 2P(1-P),$$

donde $P = P(y=-1|\tau)$. El índice de Gini es la opción por default en la mayoría de los módulos de software en R y Python.

Arboles de clasificación

Estrategias para dividir un nodo.

La otra función es el índice de diversidad de Gini:

$$-\sum_{k\neq k'}^{K} P(k|\tau)P(k'|\tau) = 1 - \sum_{k}[P(k|\tau)]^2,$$

de igual forma para el caso de clasificación binaria, Gini se reduce a

$$i(\tau) = 2P(1-P),$$

donde $P = P(y=-1|\tau)$. El índice de Gini es la opción por default en la mayoría de los módulos de software en R y Python.

En ambos casos, la impureza será máxima cuando las clases estén mezcladas, i.e, $P = 1/2$ para dos clases.

Arboles de clasificación

Estrategias para dividir un nodo: Queremos

- Nodos terminales más "puros" (reduce incertidumbre al calcular $\hat{y}$).
- Equivalente a reducir la impureza de los nodos en cada split que los forman.

Arboles de clasificación

Estrategias para dividir un nodo: Queremos

- Nodos terminales más "puros" (reduce incertidumbre al calcular $\hat{y}$).
- Equivalente a reducir la impureza de los nodos en cada split que los forman.

Definimos la **bondad del split** $\delta i(h, \tau)$ de un nodo τ como la reducción de impureza obtenida al dividir el nodo padre τ en sus nodos hijos τ_R y τ_L:

$$\delta i(h, \tau) = i(\tau) - p_L i(\tau_L) - p_R i(\tau_R),$$

donde p_L, p_R es la proporción de observaciones que van al nodo izquierdo y derecho, respectívamente.
Entonces, para una variable X_j **elegimos el split** $h \in \mathcal{H}$ **que maximice** $\delta i(h, \tau)$, donde $\mathcal{H}$ es el conjunto de todos los posibles splits para X_j.

Arboles de clasificación

Estrategias para crear el árbol.

- Esta estrategia se repite para cada nodo (recursive partitioning), hasta formar un árbol de clasificación. ¿Hasta dónde puede crecer un nodo?
- Un criterio para detener el árbol de clasificación es restringir su tamaño de antemano. Por ejemplo, declarar **nodos terminales** cuando el número de observaciones en tales nodos es menor o igual a cierto valor.
- Otro criterio es hacer que el árbol crezca a su tamaño máximo (modelo saturado) y luego "podarlo".

Arboles de clasificación

Estrategias para asignar nodos terminales a clases.

- Una vez que declaramos un nodo como nodo terminal, debemos asignar la clase correspondiente.
- La estrategia más usada está relacionada con el clasificador óptimo Bayesiano: asignar la clase más probable:
$$\hat{y} = \max_k p(y = k|\tau),$$
donde $k \in -1, 1$.
Una forma de estimar $p(y = k|\tau) = n_k(\tau)/n(\tau)$.
- Si existe un costo asociado a **Clasificar mal un dato de la clase** i, entonces puede adaptarse la regla de Bayes introduciendo costos de mala clasificación. En este caso se escogerá la clase que minimice el costo de mala clasificación.

Arboles de clasificación

Selección del árbol óptimo.

En CART, el procedimiento estándar es crear un árbol grande y después "podarlo" de abajo hacia arriba, hasta obtener el tamaño correcto según un criterio basado en una medida del error obtenido. En CART, se utiliza la tasa de datos mal clasificados $R(T)$.

Arboles de clasificación

Selección del árbol óptimo.

En CART, el procedimiento estándar es crear un árbol grande y después "podarlo" de abajo hacia arriba, hasta obtener el tamaño correcto según un criterio basado en una medida del error obtenido. En CART, se utiliza la tasa de datos mal clasificados $R(T)$.
Sea T un árbol de clasificación y $\{\tau_1, \tau_2, \ldots, \tau_L\}$ el conjunto de nodos terminales de T. La estimación de $R(T)$ es

$$R(T) = \sum_{l=1}^{L} R(\tau_l) P(\tau_l),$$

donde $P(\tau_l)$ es la probabilidad de que una observación caiga en el nodo τ_l.

Arboles de clasificación

Selección del árbol óptimo.

En la práctica, se usa resubstitution estimate de $R(T)$:

$$R(T) = \sum_{l=1}^{L} r(\tau_l) p(\tau_l),$$

donde

$$r(\tau_l) = 1 - \max_k p(y = k|\tau_l),$$

con $p(y = k|\tau_l)$ estimada como lo hicimos en la asignación de clases a los nodos terminales (Bayes), y $p(\tau_l) = n(\tau_l)/n$.

Arboles de clasificación

Selección del árbol óptimo.

Una vez definida la medida de error, el procedimiento para podar el árbol es el siguiente.

1. Crear un árbol grande $T_{\text{máx}}$, por ejemplo, poniendo un criterio de paro basado en una cantidad mínima de observaciones, es decir, seguir dividiendo los nodos hasta que contengan menos de $n_{\text{mín}}$ observaciones.
2. Calcular la estimación de $R(\tau_l)$ para cada nodo terminal $\tau_l \in T_{\text{máx}}$.
3. Podar $T_{\text{máx}}$ desde abajo hacia arriba (nodo raíz) de tal forma que en cada etapa del proceso de poda, se minimice una versión regularizada de $R(T)$.

Arboles de clasificación

Selección del árbol óptimo.

Regularización y poda mediante Minimal Cost-Complexity Pruning (Breiman, 1984).
Para algún árbol T, la medida de Cost-Complexity se define como

$$R_\alpha(T) = R(T) + \alpha|T|,$$

donde $|T|$ es el número de nodos terminales en T y $R(T)$ es la tasa de error.
$\alpha \geq 0$ es el parámetro de regularización que penaliza la complejidad del modelo (en éste caso el tamaño el árbol).

Arboles de clasificación

Selección del árbol óptimo.

Considera algún nodo t de un árbol, y T_t el subárbol que se genera teniendo como padre el nodo t.
Para un α dado, se puede construir un árbol mediante un proceso de poda, haciendo crecer un árbol grande (e.g., a su tamaño máximo) y eliminando ramas del árbol de forma secuencial, hasta que el árbol final podado tenga valores

$$\min \alpha^*(t) \geq \alpha, \qquad \text{para todo } t,$$

donde

$$\alpha^*(t) = \frac{R(t) - R(T_t)}{|T| - 1},$$

El valor α óptimo puede obtenerse mediante una búsqueda sobre los árboles resultantes del proceso de poda o con un esquema de validación cruzada.

Arboles de clasificación

Volviendo al ejemplo anterior:

`notebooks/CART.ipynb`

Bagging y random forest (RF)

Bagging

- Bagging (Breiman, L. Bagging predictors. Mach Learn 24, 123–140 (1996)) es un acrónimo de bootstrap aggregating.

Bagging

- Bagging (Breiman, L. Bagging predictors. Mach Learn 24, 123–140 (1996)) es un acrónimo de bootstrap aggregating.
- Bagging fue el primer procedimiento que combinó exitósamente un ensamble de algoritmos de aprendizaje para mejorar el desempeño de uno solo de los algoritmos usados.

Bagging

- Bagging (Breiman, L. Bagging predictors. Mach Learn 24, 123–140 (1996)) es un acrónimo de bootstrap aggregating.
- Bagging fue el primer procedimiento que combinó exitósamente un ensamble de algoritmos de aprendizaje para mejorar el desempeño de uno solo de los algoritmos usados.
- En términos generales, bootstrap es un método de remuestreo, donde a partir de un conjunto de datos de entrenamiento de tamaño n, se selecciona una muestra (bootstrap sample) del mismo tamaño n, aleatoriamente con reemplazo.

Bagging

- En machine learning, bootstrap nos proporciona una forma computacional de **estimar el error de generalización** a partir de un conjunto de datos de entrenamiento, tomando un número determinado de muestras bootstrap donde se ajusta un modelo de predicción.

Bagging

- En machine learning, bootstrap nos proporciona una forma computacional de **estimar el error de generalización** a partir de un conjunto de datos de entrenamiento, tomando un número determinado de muestras bootstrap donde se ajusta un modelo de predicción.
- Bagging es más eficiente si el predictor usado es inestable (con mucha varianza). Si el predictor es estable, el predictor obtenido mediante bagging será muy parecido al modelo individual.

Bagging

Considera un conjunto de datos de entrenamiento:

$$\mathcal{Z} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n},$$

con $\mathbf{x}_i \in \mathbb{R}^d$ y y_i la variable de respuesta, ya sea contínua (regresión) o categórica (clasificación).

Bagging forma un ensamble de k modelos entrenados en conjuntos de datos de entrenamiento $\{\mathcal{Z}_k\}$ obtenido mediante bootstrap, los cuales se combinan posteriormente para obtener el clasificador final.

Bagging

Considera un conjunto de datos de entrenamiento:

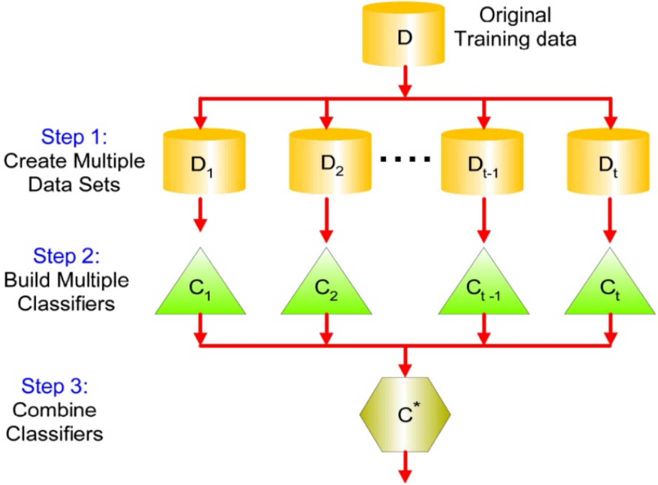
$$\mathcal{Z} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n},$$

con $\mathbf{x}_i \in \mathbb{R}^d$ y y_i la variable de respuesta, ya sea contínua (regresión) o categórica (clasificación).

Bagging forma un ensamble de k modelos entrenados en conjuntos de datos de entrenamiento $\{\mathcal{Z}_k\}$ obtenido mediante bootstrap, los cuales se combinan posteriormente para obtener el clasificador final.

Si las muestras son tomadas <u>sin reemplazo</u>, el método se llama <u>pasting</u> Breiman, L. Pasting Small Votes for Classification in Large Databases and On-Line. Machine Learning 36, 85–103 (1999).

Bagging


Original Training data
D
Step 1:
Create Multiple Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2:
Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3:
Combine Classifiers
C^*

Bagging

Evaluación OOB

- Para estimar el error de generalización, se requiere un conjunto de datos de prueba independiente. En vez de usar conjunto(s) de validación **elegidos apriori**, la evaluación Out Of Bag crea conjuntos de validación con cada muestra boostrap.

Bagging

Evaluación OOB

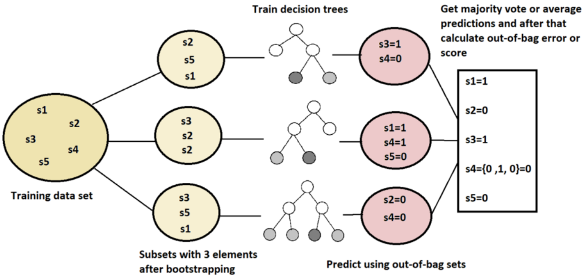
- Para estimar el error de generalización, se requiere un conjunto de datos de prueba independiente. En vez de usar conjunto(s) de validación **elegidos apriori**, la evaluación Out Of Bag crea conjuntos de validación con cada muestra boostrap.
- En bagging, hay cierta probabilidad de que un conjunto de datos nunca sea seleccionado, y que otros sean seleccionados muchas veces. Puede mostrarse (ver Breiman, 1996 ó Izenman, 2008) que aproximádamente 37 % de los datos no serán seleccionados en el procedimiento de Bagging.

Bagging

Evaluación OOB

- Para estimar el error de generalización, se requiere un conjunto de datos de prueba independiente. En vez de usar conjunto(s) de validación **elegidos apriori**, la evaluación Out Of Bag crea conjuntos de validación con cada muestra boostrap.
- En bagging, hay cierta probabilidad de que un conjunto de datos nunca sea seleccionado, y que otros sean seleccionados muchas veces. Puede mostrarse (ver Breiman, 1996 ó Izenman, 2008) que aproximádamente 37 % de los datos no serán seleccionados en el procedimiento de Bagging.
- Al conjunto de datos no seleccionados se les llama observaciones out-of-bag (OOB), y pueden usarse como un conjunto de datos de prueba para evaluar el modelo obtenido con bagging.

Bagging

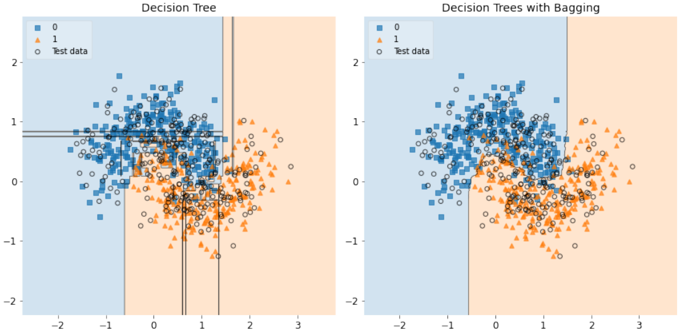


Observa que, a diferencia del error de validación tradicional (por ejemplo, de k-FOLD CV), el error OOB se calcula usando **un subconjunto** de los modelos ajustados: aquellos que no contenían los datos OOB en su conjunto de datos de entrenamiento.

Bagging

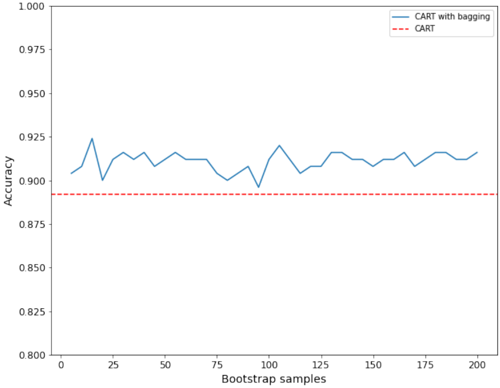
Ejemplo: bagging con árboles de decisión (CART).

Recuerda que CART tiene sobreajuste cuando el modelo es muy grande o no se usa regularización.



Bagging

Ejemplo: bagging con árboles de decisión (CART).



Random forests

- Bagging es una técnica para reducir la varianza de una función de predicción estimada. Este método de ensamble funciona muy bien para algoritmos de estimación sobreajustados, es decir, con alta varianza y poco sesgo (e.g. CART sin regularización).

Random forests

- Bagging es una técnica para reducir la varianza de una función de predicción estimada. Este método de ensamble funciona muy bien para algoritmos de estimación sobreajustados, es decir, con alta varianza y poco sesgo (e.g. CART sin regularización).
- Cuando se usa CART + bagging, cada árbol es **idénticamente distribuido**, y el valor esperado de un promedio de B árboles es el mismo que el valor esperado de cualquier árbol individual, entonces la única forma de mejorar el resultado del ensamble de CARTs es reducir su varianza.

Random forests

- Bagging es una técnica para reducir la varianza de una función de predicción estimada. Este método de ensamble funciona muy bien para algoritmos de estimación sobreajustados, es decir, con alta varianza y poco sesgo (e.g. CART sin regularización).
- Cuando se usa CART + bagging, cada árbol es **idénticamente distribuido**, y el valor esperado de un promedio de B árboles es el mismo que el valor esperado de cualquier árbol individual, entonces la única forma de mejorar el resultado del ensamble de CARTs es reducir su varianza.
- La idea esencial de bagging es promediar muchos modelos de predicción "ruidosos" pero con poco sesgo, reduciendo de ésta forma la varianza.

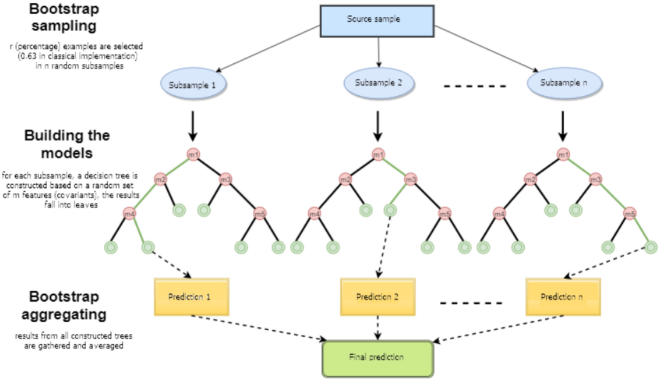
Random forests

- Bagging es una técnica para reducir la varianza de una función de predicción estimada. Este método de ensamble funciona muy bien para algoritmos de estimación sobreajustados, es decir, con alta varianza y poco sesgo (e.g. CART sin regularización).
- Cuando se usa CART + bagging, cada árbol es **idénticamente distribuido**, y el valor esperado de un promedio de B árboles es el mismo que el valor esperado de cualquier árbol individual, entonces la única forma de mejorar el resultado del ensamble de CARTs es reducir su varianza.
- La idea esencial de bagging es promediar muchos modelos de predicción "ruidosos" pero con poco sesgo, reduciendo de ésta forma la varianza.
- RF $\neq$ Bagging+CART

Random forests

- La idea principal en RF es mejorar la reducción de varianza de bagging disminuyendo la correlación entre los árboles, sin incrementar demasiado la varianza.
- Esto se logra en el proceso de ajuste de los árboles mediante la selección aleatoria de las variables de entrada.
- Específicamente, cuando se ajusta un modelo CART en la muestra bootstrap, **antes de cada split, se seleccionan aleatoriamente $m < d$ de las variables de entrada como candidatas para el split**.

Random forests



Random forests

Algoritmo RF

for $b = 1$ to B **do**

Genera una muestra bootstrap de tamaño n del conjunto de datos de entrenamiento.

Ajusta un árbol T_b a la muestra bootstrap repitiendo los siguientes pasos de manera recursiva para cada nodo terminal del árbol, hasta que el número de datos mínimo $n_{\text{mín}}$ en el nodo se haya obtenido.

1. Selecciona m variables de manera aleatoria del conjunto original de d variables de los datos
2. Selecciona el mejor split entre ésas m variables
3. Divide el nodo en dos nodos hijos

end for

Obtén el ensamble de árboles $\{T_b\}_1^B$

La predicción para un nuevo dato $\mathbf{x}$ se realiza mediante:

- Regresión: $\hat{f}_{rf}^B(\mathbf{x}) = \dfrac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$
- Clasificación: $\hat{y}_{rf}^B(\mathbf{x}) = \text{votación mayoritaria}\{\hat{y}_b(\mathbf{x})\}_1^B$.

Adaboost

Ejemplos:

`notebooks/metodos_ensamble.ipynb`

`notebooks/metodos_ensamble_practica.ipynb`