Classification

Machine Learning

Ignacio Sarmiento-Barbieri

Universidad de La Plata

Agenda

Classification: Motivation

- Many predictive questions are about classification
  - Email should go to the spam folder or not
  - A household is bellow the poverty line
  - Accept someone to a graduate program or no
- Aim is to classify y based on $X's$

Classification: Motivation

- Main difference is that y represents membership in a category: $y \in \{1, 2, ..., n\}$
  - Qualitative (e.g., spam, personal, social)
  - Not necessarily ordered

The prediction question is, given a new X,
what is our best guess at the response category $\hat{y}$

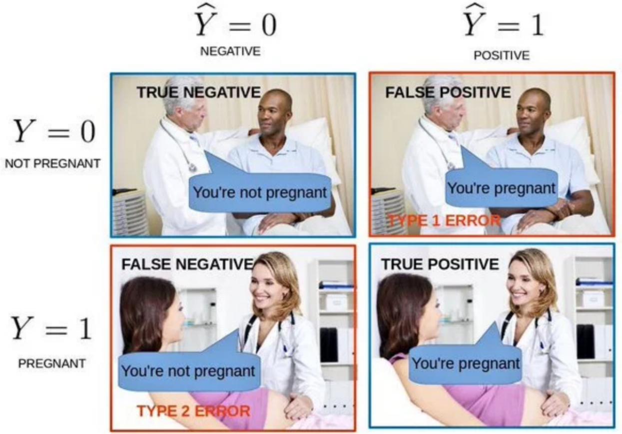
Agenda

Risk, Probability, and Classification

- Two states of nature $Y \rightarrow i \in \{0, 1\}$
- Two actions $(\hat{Y}) \rightarrow j \in \{0, 1\}$

		$\hat{Y}$ 0	$\hat{Y}$ 1
Y	0	True Negative	False Positive
	1	False Negative	True Positive

Risk, Probability, and Classification



Source: https://dzone.com/articles/understanding-the-confusion-matrix

Risk, Probability, and Classification

- Two actions $\hat{Y} \to j \in \{0, 1\}$
- Two states of nature $Y \to i \in \{0, 1\}$
- Probabilities
  - $p = Pr(Y = 1|X)$
  - $1 - p = Pr(Y = 0|X)$

Risk, Probability, and Classification

- Actions have costs associated to them
- Loss: $L(i,j)$, penalizes being in bin i,j
  - We define $L(i,j)$

$$L(i,j) = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases} \tag{1}$$

Risk, Probability, and Classification

- Risk: expected loss of taking action j

$$E[L(i,j)] = \sum_i p_j L(i,j) \tag{2}$$

$$R(j) = (1-p)L(0,j) + pL(1,j)$$

- The objective is to minimize the risk

Agenda

Bayes classifier

$$R(1) \quad < R(0) \tag{3}$$

Bayes classifier

- Under a 0-1 penalty the problem boils down to finding

$$p = Pr(Y = 1|X) \tag{4}$$

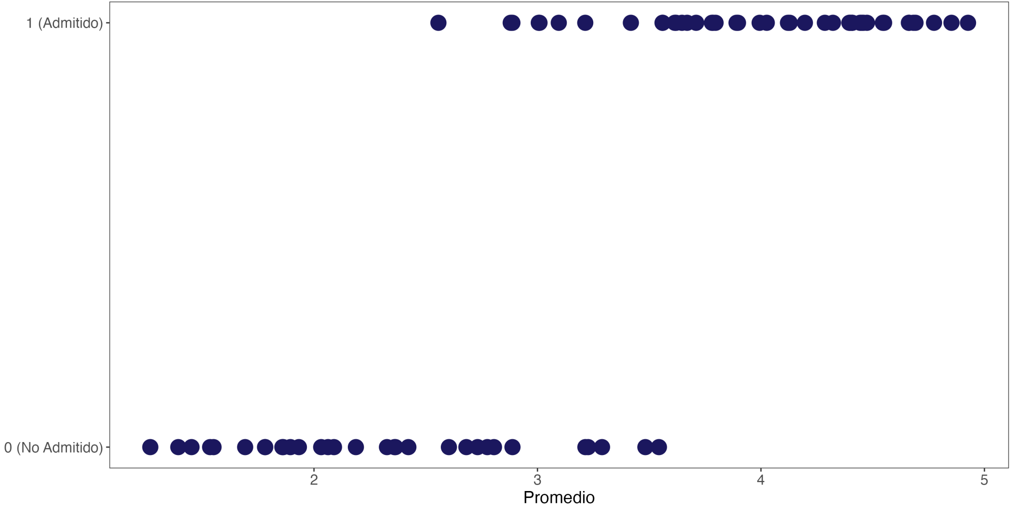
- We then predict 1 if $p > 0.5$ and 0 otherwise (Bayes classifier)
- Many ways of finding this probability in binary cases

Agenda

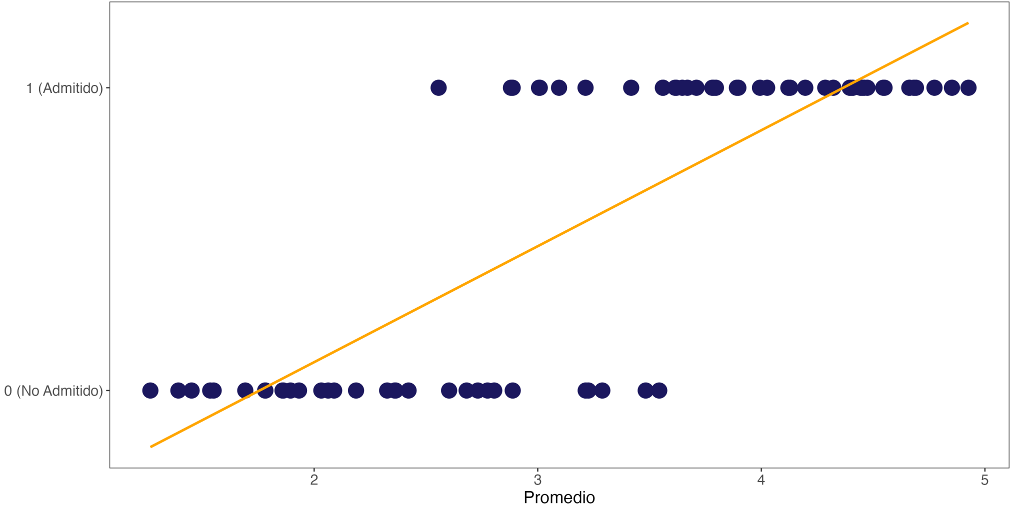
Setup

- Y is a binary random variable$\{0, 1\}$
- X is a vector of K predictors
- $p = Pr(Y = 1|X)$

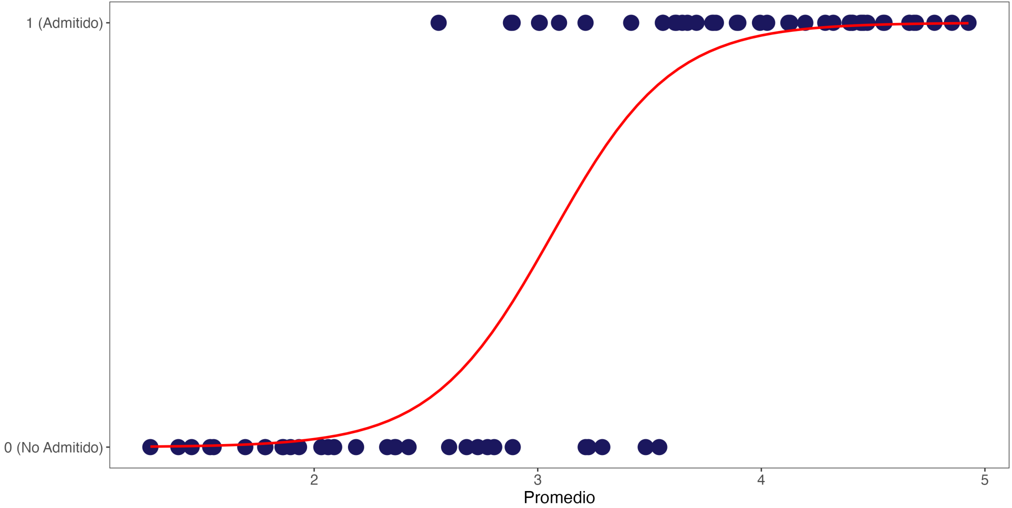
Logit



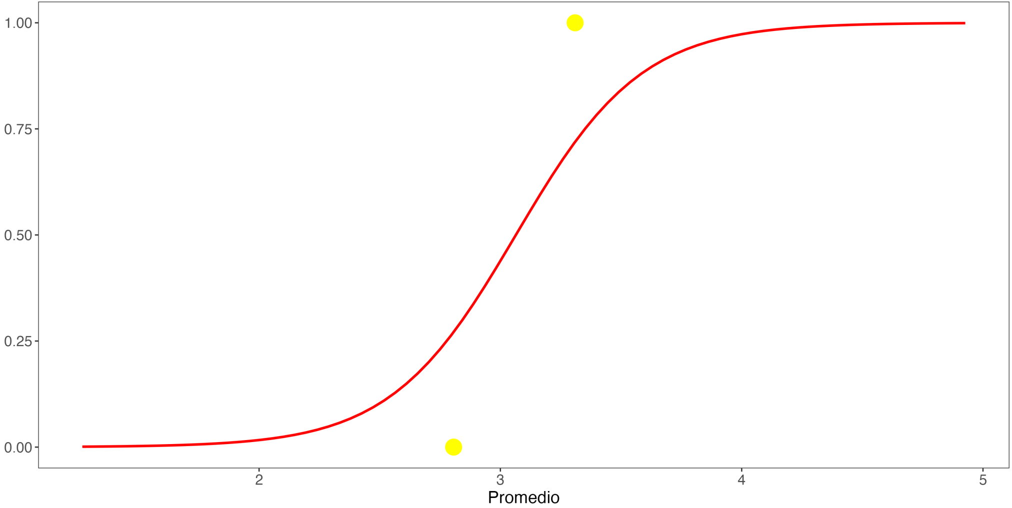
Logit



Logit



Logit



Logit

- Logit

$$
\begin{aligned}
p &= \frac{e^{X\beta}}{1+e^{X\beta}} \\
&= \frac{exp(\beta_0+\beta_1 x_1+\cdots+\beta_k x_k)}{1+exp(\beta_0+\beta_1 x_1+\cdots+\beta_k x_k)}
\end{aligned} \tag{5}
$$

Logit

- Logit

$$p = \frac{e^{X\beta}}{1+e^{X\beta}} \tag{5}$$
$$= \frac{exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}{1 + exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}$$

- Odds ratio

$$ln\left(\frac{p}{1-p}\right) = X\beta \tag{6}$$
$$= \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

Agenda

Aside: Maximum Likelihood Estimation

- Developed by Ronald A. Fisher (1890-1962)
- "If Fisher had lived in the era of "apps," maximum likelihood estimation might have made him a billionaire" (Efron and Tibshiriani, 2016)
- Why? MLE gives "automatically"
  - Consistent
  - Asymptotically normal
  - Asymptotically efficient

Aside: Maximum Likelihood Estimation

$$Pr(Y = y|X) = f(y;\theta) \tag{7}$$

- $f()$ known
- θ unknown
- Example:

$$Y|X \sim Poisson(\lambda) \tag{8}$$

$$f(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!} \tag{9}$$

Aside: Maximum Likelihood Estimation

- $Y_1, \ldots, Y_n \sim_{iid} f(Y;\theta)$

$$Pr(Y_i = y_i | X_i) = f(y_i;\theta) \tag{10}$$

- Likelihood

$$L(\theta; y_i) = f(y_i;\theta) \tag{11}$$

Aside: Maximum Likelihood Estimation

- For a random sample $Y_1, \ldots, Y_n \sim_{iid} f(Y;\theta)$
- The likelihood function is

$$L(\theta|y_1,\ldots,y_n) = \Pi_{i=1}^n L(\theta;y_i) \quad (12)$$
$$= \Pi_{i=1}^n f(x_i;\theta)$$

- A maximum likelihood estimator of the parameter θ:

$$\hat{\theta}^{MLE} = \underset{\theta\in\Theta}{argmax}\, L(\theta,x) \quad (13)$$

Aside: Maximum Likelihood Estimation

- Note that maximizing (12) is the same as maximizing

$$l(\theta; y_1, \dots, y_n) = \ln L(\theta; y_1, \dots, y_n) = \sum_{i=1}^{n} l(\theta; y_i) \tag{14}$$

- Advantages of (14)
  - Contribution of observation i: $l_i(x|\theta) = \ln f(y_i; \theta)$
  - Eq. (12) is prone to underflow.

MLE Logit

- Imagine that we have a sample of iid observations $(y_i, x_i); i = 1, \ldots, n$, where $y_i \in \{0, 1\}$
- Under logit we have

$$p_i = \frac{e^{x_i\beta}}{1 + e^{x_i\beta}} \tag{15}$$

- Then the likelihood

$$\begin{aligned} L(\theta; y_1, \ldots, y_n) &= \Pi_{y_i=1} p_i \Pi_{y_i \neq 1} (1 - p_i) && (16) \\ &= \Pi_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i} && (17) \\ &= \Pi_{i=1}^n \left(\frac{p_i}{1 - p_i} \right)^{y_i} (1 - p_i) && (18) \end{aligned}$$

MLE Logit

- The log likelihood is then

$$l(\theta; y_1, \ldots, y_n) = \sum_{i=1}^{n} log \left(\frac{p_i}{1-p_i} \right)^{y_i} + \sum_{i=1}^{n} log(1-p_i) \tag{19}$$

- FOC

$$\frac{\partial l}{\partial \beta_j} = \sum_{i=1}^{n} \frac{y_i}{p_i(1-p_i)} \frac{\partial p_i}{\partial \beta_j} - \sum_{i=1}^{n} \frac{1}{(1-p_i)} \frac{\partial p_i}{\partial \beta_j} \tag{20}$$

$$= \sum_{i=1}^{n} \frac{y_i - p_i}{p_i(1-p_i)} \frac{\partial p_i}{\partial \beta_j} \tag{21}$$

- Note:
  - This is a system of K non linear equations with K unknown parameters.
  - We cannot explicitly solve for $\hat{\beta}$
  - It's important to check SOC

Agenda

Newton's Method

- Suppose that we wish to minimize a function $Q(\beta)$, where β is a k-vector
- $Q(\beta)$ is assumed to be twice continuously differentiable.
- Given any initial value of β, say $\beta_{(0)}$, we can perform a second-order Taylor expansion of $Q(\beta)$ around $\beta_{(0)}$ in order to obtain an approximation $(Q^*(\beta))$ to $Q(\beta)$:

$$Q^*(\beta) = Q(\beta_{(0)}) + g'_{(0)}(\beta - \beta_{(0)}) + \frac{1}{2}(\beta - \beta_{(0)})' H_{(0)}(\beta - \beta_{(0)}) \tag{22}$$

Newton's Method

- FOC

$$g_{(0)} + H_{(0)}(\beta - \beta_{(0)}) = 0 \tag{23}$$

- Solving these yields a new value of β, which we will call $\beta_{(1)}$:

$$\beta_{(1)} = \beta_{(0)} - H_{(0)}^{-1} g_{(0)} \tag{24}$$

Newton's Method

- FOC

$$g_{(0)} + H_{(0)}(\beta - \beta_{(0)}) = 0 \tag{23}$$

- Solving these yields a new value of β, which we will call $\beta_{(1)}$:

$$\beta_{(1)} = \beta_{(0)} - H_{(0)}^{-1} g_{(0)} \tag{24}$$

- If the quadratic approximation $Q^*(\beta))$ is a strictly convex function, which it will be if and only if the Hessian $H_{(0)}$ is positive definite, $\beta_{(1)}$ will be the global minimum of $Q^*(\beta))$.

quasi-Newton's Method

- Because the loglikelihood function is to be maximized, the Hessian should be negative definite
- Newton's Method will usually not work well, and will often not work at all, when the Hessian is not negative definite.
- In such cases, one popular way to obtain the MLE is to use some sort of quasi-Newton method:

$$\beta_{(j+1)} = \beta_{(j)} + \alpha_j D_{(j)}^{-1} g_{(j)} \tag{25}$$

- where $\alpha_{(j)}$ is a scalar which is determined at each step
- $D_{(j)}$ is a matrix which approximates $-H_{(j)}$ near the maximum but is constructed so that it is always positive definite.

Agenda

Summary

- We observe (y_i, X_i) $i = 1, \ldots, n$
- Logit

$$p_i = \frac{e^{X_i\beta}}{1 + e^{X_i\beta}} \tag{26}$$

- Prediction

$$\hat{p}_i = \frac{e^{X_i\hat{\beta}}}{1 + e^{X_i\hat{\beta}}} \tag{27}$$

- Classification

$$\hat{Y}_i = 1[\hat{p}_i > 0.5] \tag{28}$$

Example

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/

Agenda

Agenda

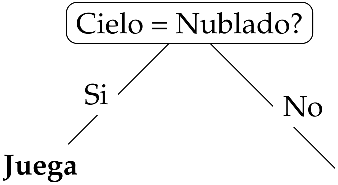
Árboles: Problema

- Jugamos al tenis?

Cielo	Humedad	Tenis?
Sol	Alta	No
Sol	Alta	No
Nublado	Alta	Sí
Sol	Alta	No
Sol	Normal	Sí
Nublado	Alta	Sí
Nublado	Normal	Sí

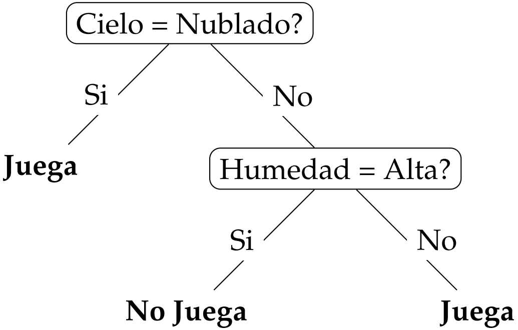
Árboles: Problema

Cielo	Humedad	Tenis?
Sol	Alta	No
Sol	Alta	No
Nublado	Alta	Sí
Sol	Alta	No
Sol	Normal	Sí
Nublado	Alta	Sí
Nublado	Normal	Sí



Árboles: Problema

Cielo	Humedad	Tenis?
Sol	Alta	No
Sol	Alta	No
Nublado	Alta	Sí
Sol	Alta	No
Sol	Normal	Sí
Nublado	Alta	Sí
Nublado	Normal	Sí



¿Cómo construimos un árbol de decisión?

- Regiones lo más “puras” posibles
  - **Regresión**: minima varianza
  - **Clasificación**: ?

¿Cómo construimos un árbol de decisión?

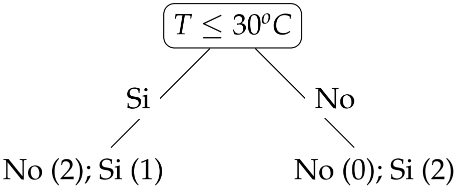
Problemas de clasificación

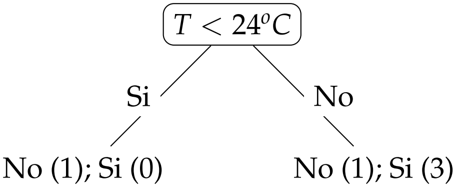
Temperatura °C	Llovió
23	NO
24	NO
29	SI
31	SI
33	SI

¿Cómo construimos un árbol de decisión?

Problemas de clasificación

- ¿Cuál de los dos cortes es mejor?





¿Cómo construimos un árbol de decisión?

Problemas de clasificación. Medidas de Impureza

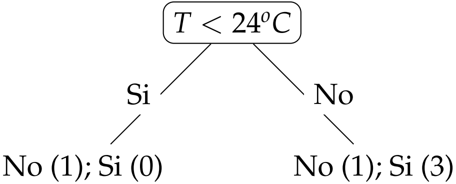
- Medidas de impureza dentro de cada hoja:
  - Índice de Gini : $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
  - Entropía : $-\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$
- Se define la impureza de un árbol por el promedio ponderado de las impurezas de cada hoja. El ponderador es la fracción de observaciones en cada hoja.

¿Cómo construimos un árbol de decisión?

Problemas de clasificación. Impureza

- ¿Cuál de los dos cortes es mejor?

Temperatura °C	Llovió
31	SI
24	NO
29	SI
33	SI
23	NO

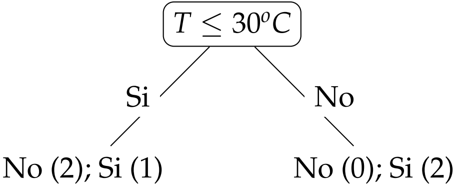


¿Cómo construimos un árbol de decisión?

Problemas de clasificación. Impureza

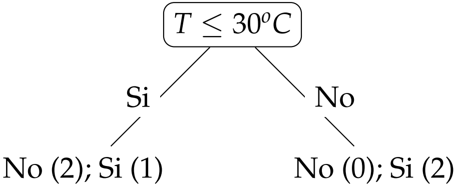
- ¿Cuál de los dos cortes es mejor?

Temperatura °C	Llovió
31	SI
24	NO
29	SI
33	SI
23	NO

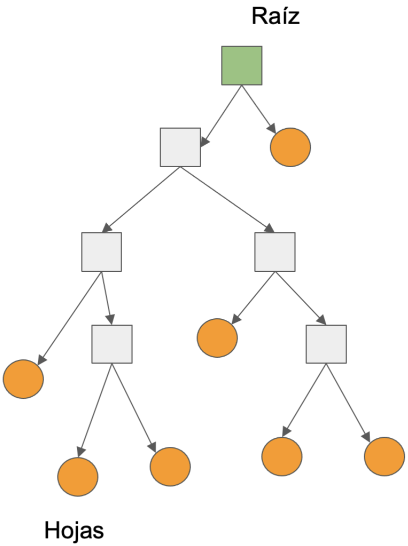


¿Cómo construimos un árbol de decisión?

Problemas de clasificación. Predicción



Sobreajuste



Sobreajuste. Algunas soluciones

- Fijar la profundidad del árbol.
- Fijar la mínima cantidad de datos que están contenidos dentro de cada hoja.
- Pruning (poda).

Agenda

Bagging

- Problema con CART: pocos robustos.
- Podemos mejorar mucho el rendimiento mediante la agregación: Bagging y Random Forests

Bagging

- Bagging:
  - Obtenga repetidamente muestras aleatorias $(X_i^b, Y_i^b)_{i=1}^N$ de la muestra observada (bootstrap).
  - Para cada muestra, ajuste un árbol de regresión $\hat{f}^b(x)$
  - Promedie las muestras de bootstrap

$$\hat{f}_{bag} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{29}$$

- Bosques (forests):
  - Si hay p predictores, en cada partición utiliza un subconjunto de predictores elegidos al azar.
  - Reduce la correlación entre los árboles en el boostrap.

Agenda

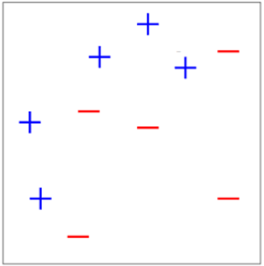
Boosting: Motivation

- Problema con CART: varianza alta.
- Podemos mejorar mucho el rendimiento mediante la agregación
- El boosting toma esta idea pero lo ”encara” de una manera diferente $\rightarrow$ viene de la computación

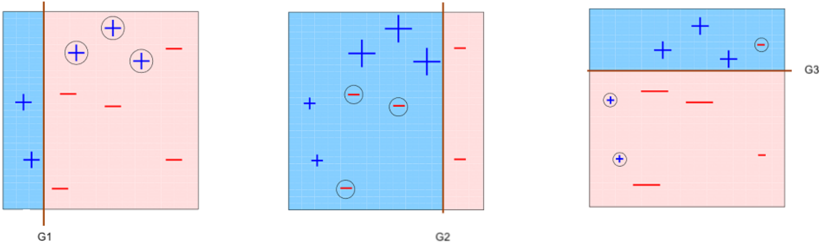
AdaBoost: Boosting Adaptativo

- Vocabulario:
  - $y \in -1, 1$, X vector de predictores.
  - $\hat{y} = G(X)$ (clasificador)
  - $err = \frac{1}{N} \sum_i^N I(y_i \neq G(x_i))$

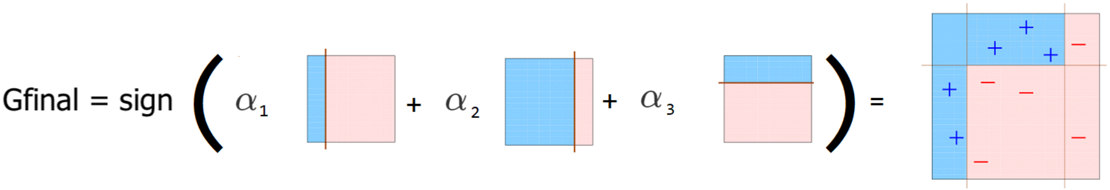
AdaBoost



AdaBoost



AdaBoost



AdaBoost.M1

1. Comenzamos con ponderadores $w_i = 1/N$
2. Para m = 1 hasta M:
   1. Estimar $G_m(x)$ usando ponderadores w_i .
   2. Computar el error de predicción

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{30}$$

   3. Obtener $\alpha_m = ln\left[\frac{(1-err_m)}{err_m}\right]$
   4. Actualizar los ponderadores : $w_i \leftarrow w_i c_i$

$$c_i = exp\left[\alpha_m I(yi \neq G_m(x_i))\right] \tag{31}$$

3. Resultado: $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

AdaBoost.M1

- $c_i = exp\left[\alpha_m I(y_i \neq G_m(x_i))\right]$
- Si fue correctamente predicho, $c_i = 1$.
- En caso contrario, $c_i = exp(\alpha_m) = \frac{(1-err_m)}{err_m} > 1$
- En cada paso el algoritmo da mas importancia relativa a las predicciones incorrectas.
- Paso final: promedio ponderado de estos pasos

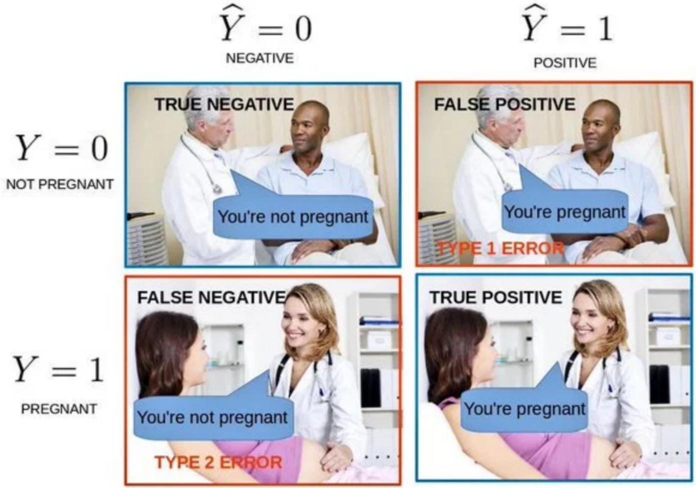
$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{32}$$

Example: Default

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/

Agenda

Misclassification Rates



Misclassification Rates

$$
\begin{array}{ccc}
 & & \hat{y}_i & \\
 & & 0 & 1 \\
\multirow{2}{*}{y_i} & 0 & TN & FP \\
 & 1 & FN & TP
\end{array}
$$

- We have several types of error associated with this that we can use as a measure of performance

Agenda

ROC

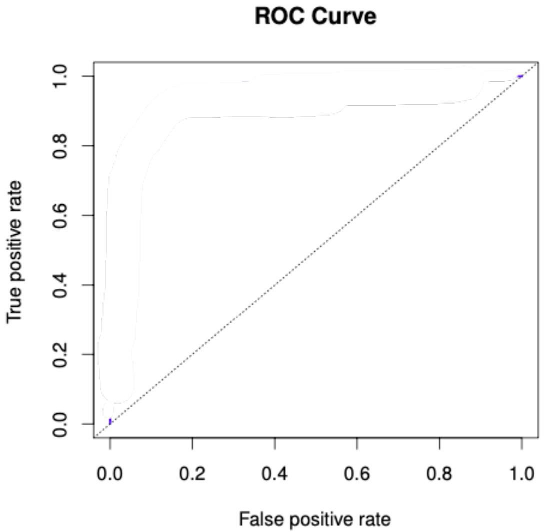
			$\hat{y}_i$	
			0	1
y_i		0	TN	FP
		1	FN	TP

- A classification rule, or cutoff, is the probability p at which you predict
  - $\hat{y}_i = 0$ if $p_i < c$
  - $\hat{y}_i = 1$ if $p_i > c$
- Bayes classifier $c = 0.5$
- Changing c changes predictions, changes FP and FN
- There is a trade-off: reducing one error increases the other

ROC

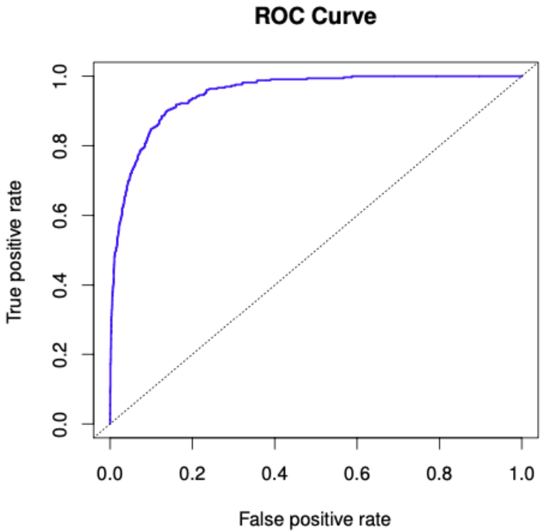
- ROC curve: Receiver operating characteristic curve
- ROC curve illustrates the trade-off of the classification rule
- Gives us the ability
  - Measure the predictive capacity of our model
  - Compare between models

ROC



ROC Curve
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

ROC


ROC Curve
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

Example: Default

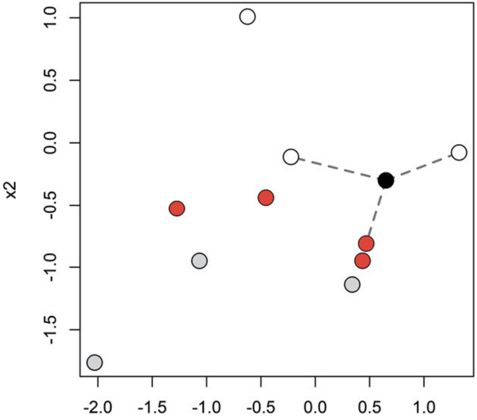
photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/

Agenda

Agenda

K-Nearest Neighbors

- What happens when we have to predict multiple outcomes?
- K nearest neighbor (K-NN) algorithm predicts class $\hat{y}$ for x by asking
  What is the most common class for observations around x?



K-Nearest Neighbors

- K nearest neighbor (K-NN) algorithm predicts class $\hat{y}$ for x by asking
  What is the most common class for observations around x?
- Algorithm: given an input vector x_f where you would like to predict the class label
  - Find the K nearest neighbors in the dataset of labeled observations, $\{x_i, y_i\}_{i=1}^n$, the most common distance is the Euclidean distance:

$$d(x_i, x_f) = \sqrt{\sum_{j=1}^{p}(x_{ij} - x_{fj})^2} \tag{33}$$

  - This yields a set of the K nearest observations with labels:

$$[x_{i1}, y_{i1}], \ldots, [x_{iK}, y_{iK}] \tag{34}$$

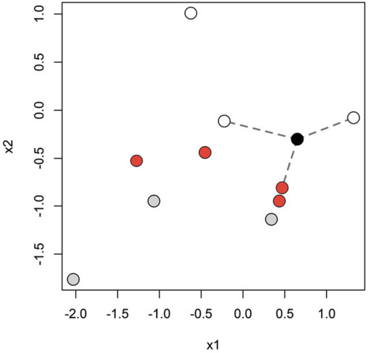
  - The predicted class of x_f is the most common class in this set

$$\hat{y}_f = mode\{y_{i1}, \ldots, y_{iK}\} \tag{35}$$

K-Nearest Neighbors

- There are some major problems with practical implications
  - Knn predictions are unstable as a function of K

$$
\begin{aligned}
K = 1 &\implies \hat{p}(white) = 0 \\
K = 2 &\implies \hat{p}(white) = 1/2 \\
K = 3 &\implies \hat{p}(white) = 2/3 \\
K = 4 &\implies \hat{p}(white) = 1/2
\end{aligned}
$$



Source: Taddy (2019)

K-Nearest Neighbors

- There are some major problems with practical implications
  - Knn predictions are unstable as a function of K
  - This instability of prediction makes it hard to choose the optimal K and cross validation doesn't work well for KNN
  - Since prediction for each new x requires a computationally intensive counting, KNN is too expensive to be useful in most big data settings.
  - KNN is a good idea, but too crude to be useful in practice