

# Clasificación

**Taller de Procesamiento de Señales**

# Agenda

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica
- 4 Linear Discriminant Analysis
- 5 K-Vecinos más cercanos
- 6 Support Vector Machines
- 7 Árboles de decisión

# Teoría de Clasificación

queremos estimar a  $Y$  en función de  $X$ , pero el conjunto donde vive  $Y$  es finito

## Bases

**Objetivo:** Clasificar  $Y$  (con  $|\mathcal{Y}|$  finito) a partir del valor de  $X$ :  $\hat{Y} = \varphi(X)$

**Función costo:** Hard  $\rightarrow \ell(x, y) = \mathbb{1}\{y \neq \varphi(x)\}$

**Riesgo Esperado:** Probabilidad de error  $\rightarrow \mathbb{P}(Y \neq \varphi(X))$

la función costo ya no es el error cuadrático, sino la función costo 0,1. Costo 0 si acierto, costo 1 si me equivoco

# Teoría de Clasificación

## Bases

**Objetivo:** Clasificar  $Y$  (con  $|\mathcal{Y}|$  finito) a partir del valor de  $X$ :  $\hat{Y} = \varphi(X)$

**Función costo:** Hard  $\rightarrow \ell(x, y) = \mathbb{1}\{y \neq \varphi(x)\}$

**Riesgo Esperado:** Probabilidad de error  $\rightarrow \mathbb{P}(Y \neq \varphi(X))$

## Optimalidad

$$\mathbb{P}(Y \neq \varphi(X)) \geq 1 - \mathbb{E} \left[ \max_y P_{Y|X}(y|X) \right]$$

con igualdad si y solo si  $\varphi(x) = \arg \max_y P_{Y|X}(y|x)$ . clasificador optimo  
argumento maximo de la probabilidad  
condicional

**Clasificador Bayesiano:**  $\varphi(x) = \arg \max_y P_{Y|X}(y|x)$

**Error Bayesiano:**  $1 - \mathbb{E} \left[ \max_y P_{Y|X}(y|X) \right]$  minimo error al q puedo aspirar

# Clasificadores extremos critérios para chequear si mi algoritmo esta aprendiendo

## Clasificador bayesiano

El mejor clasificador (en términos de la probabilidad de error) es:

$$\mathbb{P}(Y \neq \varphi(X)) \geq 1 - \mathbb{E} \left[ \max_y P_{Y|X}(y|X) \right]$$

prob. de error

## Clasificador al azar para $k$ clases

Cualquier clasificador razonable debe ganarle a la decisión al azar:

$$\mathbb{P}(Y \neq \varphi(X)) \leq 1 - \frac{1}{k}$$

prob de error

le tengo q gnar al clasificador al azar, el que tira una moneda y decide

## Clasificador dummy es un poco mejor, elije siempre la misma clase, que en este caso es la mas probable

Otro clasificador muy precario (pero mejor que el azaroso) es elegir siempre la clase más probable. La probabilidad de error del dummy es:

$$\mathbb{P}(Y \neq \varphi(X)) \leq 1 - \max_y P_Y(y)$$

prob. de error

# Clasificador bayesiano

## Objetivo

Quiero buscar  $\varphi(\cdot)$  que minimice  $\mathbb{P}(Y \neq \varphi(X))$ . Es decir aprender el “clasificador bayesiano”:  $\varphi(x) = \arg \max_y P_{Y|X}(y|x)$ .

no sabemos la verdadera distribucion de los datos, usamos muestras y estimaciones empiricas

# Clasificador bayesiano

## Objetivo

Quiero buscar  $\varphi(\cdot)$  que minimice  $\mathbb{P}(Y \neq \varphi(X))$ . Es decir aprender el “clasificador bayesiano”:  $\varphi(x) = \arg \max_y P_{Y|X}(y|x)$ .

## Problemas numéricos

minimizamos el promedio de la funcion costo

La propuesta de buscar  $\varphi(\cdot)$  que minimice el riesgo empírico:  
 $\frac{1}{n} \sum_{i=1}^n \mathbb{1}\{Y_i \neq \varphi(X_i)\}$  suele tener problemas numéricos (no derivable).

el problema: es discontinua, muy abrupta, no es derivable porque vale 0 o 1. la funcion es muy poco sensible.

# Clasificador bayesiano

## Objetivo

Quiero buscar  $\varphi(\cdot)$  que minimice  $\mathbb{P}(Y \neq \varphi(X))$ . Es decir aprender el “clasificador bayesiano”:  $\varphi(x) = \arg \max_y P_{Y|X}(y|x)$ .

## Problemas numéricos

La propuesta de buscar  $\varphi(\cdot)$  que minimice el riesgo empírico:  $\frac{1}{n} \sum_{i=1}^n \mathbb{1}\{Y_i \neq \varphi(X_i)\}$  suele tener problemas numéricos (no derivable).

## Posible solución

El clasificador bayesiano se aprenderá en dos etapas:

- Aprender toda  $P_{Y|X}(y|x)$ . esto en la proxima diap
- Quedarse con el máximo. una vez q tengo toda la prob, me quedo con el argumento que hace que la prob sea la mas grande.



# Elementos de Teoría de Información

los eventos de baja prob, cuando me los confirman, tienen mucha mas informacion.

- $H(X) = \mathbb{E} [-\log P_X(X)]$  Entropía a mayor prob, se entrega menos informacion y viceversa.
- $h(X) = \mathbb{E} [-\log p_X(X)]$  Entropía diferencial para discretas, puede ser negativa
- $H(Y|X) = \mathbb{E} [-\log P_{Y|X}(Y|X)]$  Entropía condicional  
esta esperanza es respecto a las 2 variables X e Y.
- $h(Y|X) = \mathbb{E} [-\log p_{Y|X}(Y|X)]$  Entropía diferencial condicional
- $KL(p_X \| q_X) = \mathbb{E}_{p_X} \left[ \log \left( \frac{p_X(X)}{q_X(X)} \right) \right]$  Divergencia de Kullback Leibler  
"medida de distancia entre 2 distribuciones"  
me dice cuanto se parecen da 0 cuando  $p_X$  y  $q_X$  son iguales
- $I(X; Y) = KL(p_{XY} \| p_X p_Y)$  Información Mutua  
mide cuanto comparten de informacion la variable X y la variable Y

# Elementos de Teoría de Información

- $H(X) = \mathbb{E}[-\log P_X(X)]$  Entropía
- $h(X) = \mathbb{E}[-\log p_X(X)]$  Entropía diferencial
- $H(Y|X) = \mathbb{E}[-\log P_{Y|X}(Y|X)]$  Entropía condicional
- $h(Y|X) = \mathbb{E}[-\log p_{Y|X}(Y|X)]$  Entropía diferencial condicional
- $\text{KL}(p_X \| q_X) = \mathbb{E}_{p_X} \left[ \log \left( \frac{p_X(X)}{q_X(X)} \right) \right]$  Divergencia de Kullback Leibler
- $I(X; Y) = \text{KL}(p_{XY} \| p_X p_Y)$  Información Mutua

## Teorema

$$\text{KL}(P \| Q) \geq 0$$

siempre es positiva  
si es muy grande están lejos  
si es muy chica están muy cerca

con igualdad si y solo si  $P(y) = Q(y)$  para todo  $y \in \mathcal{Y}$ .

(Hint:  $\log(x) \leq x - 1$ ).

# Divergencia de Kullback Leibler

## Propuesta inicial

Busco  $\hat{P}(y|x)$  que minimice:

$$\underbrace{\mathbb{E} \left[ KL \left( P_{Y|X}(\cdot|X) \parallel \hat{P}(\cdot|X) \right) \right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E} \left[ -\log \hat{P}(Y|X) \right]}_{\text{Cross-entropy}} - \underbrace{H(Y|X)}_{\substack{\text{Entropía condicional} \\ \text{es un factor constante}}}$$

esta es la que se busca minimizar

# Divergencia de Kullback Leibler

## Propuesta inicial

Busco  $\hat{P}(y|x)$  que minimice:

$$\underbrace{\mathbb{E} \left[ KL \left( P_{Y|X}(\cdot|X) \parallel \hat{P}(\cdot|X) \right) \right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E} \left[ -\log \hat{P}(Y|X) \right]}_{\text{Cross-entropy}} - \underbrace{H(Y|X)}_{\text{Entropía condicional}}$$

Optimalidad para  $\ell(x, y) = -\log \hat{P}(y|x)$  "logloss"

$$\mathbb{E} \left[ -\log \hat{P}(Y|X) \right] \geq H(Y|X)$$

son igualdad si y solo si  $\hat{P}(y|x) = P_{Y|X}(y|x)$  para todo  $(x, y)$ .

# Divergencia de Kullback Leibler

## Propuesta inicial

Busco  $\hat{P}(y|x)$  que minimice:

$$\underbrace{\mathbb{E} \left[ KL \left( P_{Y|X}(\cdot|X) \parallel \hat{P}(\cdot|X) \right) \right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E} \left[ -\log \hat{P}(Y|X) \right]}_{\text{Cross-entropy}} - \underbrace{H(Y|X)}_{\text{Entropía condicional}}$$

Optimalidad para  $\ell(x, y) = -\log \hat{P}(y|x)$

$$\mathbb{E} \left[ -\log \hat{P}(Y|X) \right] \geq H(Y|X)$$

son igualdad si y solo si  $\hat{P}(y|x) = P_{Y|X}(y|x)$  para todo  $(x, y)$ .

## Mismatch de métricas

El mínimo de la cross entropy no tiene por que coincidir exactamente con el mínimo de la probabilidad de error. En general se mira la cross entropy para reducir el bias y la probabilidad de error para prevenir el overfitting.

# Hard/Soft Decision

## Decisión Suave

"predict probab"

Sea  $x \in \mathbb{R}^d$ , llamamos predicción *soft* de un algoritmo a la predicción de las probabilidades estimadas  $\hat{P}(\cdot|x)$ . Esta estimación es un vector de probabilidades de todas las clases posibles (no negativas y suman 1). Su desempeño se suele medir con la entropía cruzada  $\mathbb{E}[-\log \hat{P}(Y|X)]$ .

## Decisión Dura

"predict"

Sea  $x \in \mathbb{R}^d$ , llamamos predicción *hard* de un algoritmo a la predicción final de la clase estimada  $\varphi(x)$ . Es decir, es una estimación del valor de  $Y$ . Generalmente se la suele definir a partir de la predicción *soft* como:

$$\varphi(x) = \arg \max_{y \in \mathcal{Y}} \hat{P}(y|x)$$

Se desempeño se suele medir con la probabilidad de acierto  $\mathbb{P}(Y = \varphi(X))$ .

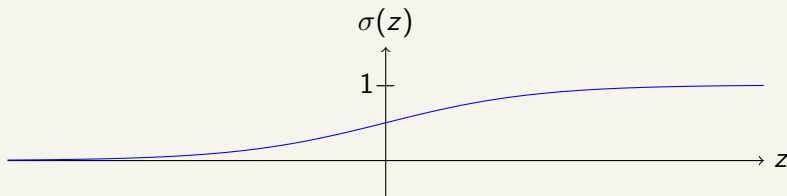
hasta aca los 2 primeros ej. guia 2

# Regresión Logística Binaria

## Función Sigmoide

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- $\sigma(z)$  representa probabilidades. porque siempre da algo entre 0 y 1
- $z$  recibe el nombre de *logit*.



# Regresión Logística Binaria

## Función Sigmoide

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- $\sigma(z)$  representa probabilidades.
- $z$  recibe el nombre de *logit*.



## Propuesta

traemos la idea de la regresión lineal

$$\begin{aligned}\hat{P}(1|x) &= \sigma(w^T x + b) \\ \hat{P}(0|x) &= 1 - \sigma(w^T x + b)\end{aligned}$$



# Regresión Logística Binaria

## Riesgo empírico

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n \ell(X_i, Y_i) = \\ - \frac{1}{n} \sum_{i=1}^n Y_i \log \left( \sigma(w^T X_i + b) \right) + (1 - Y_i) \log \left( 1 - \sigma(w^T X_i + b) \right) \end{aligned}$$

# Regresión Logística Binaria

## Riesgo empírico

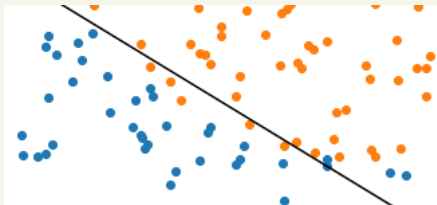
$$\frac{1}{n} \sum_{i=1}^n \ell(X_i, Y_i) =$$
$$- \frac{1}{n} \sum_{i=1}^n Y_i \log \left( \sigma(w^T X_i + b) \right) + (1 - Y_i) \log \left( 1 - \sigma(w^T X_i + b) \right)$$

vamos a seguir buscando el  $w$  y el  $b$  que minimizan

## Elección del máximo

ver si el logit es positivo o negativo

$$\hat{P}(1|x) \leq \hat{P}(0|x) \Leftrightarrow w^T x + b \leq 0$$



ahora tengo q encontrar el máximo

la frontera de decision es una recta, una solución lineal

# Curvas ROC

## Agregar un umbral

Puedo darle más peso a una clase:

$$w^T x + b \leq t$$

muevo la recta de decision

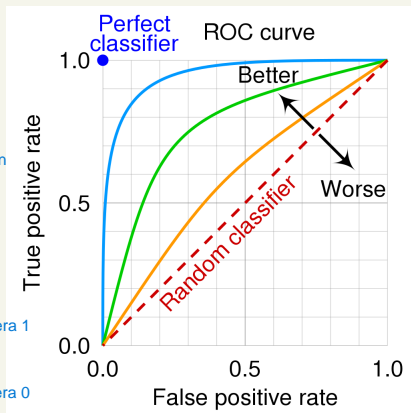
forma de evaluar todos los clasificadores  
juntos, para todos los valores de  $t$

$$\text{TPR} = \mathbb{P}(Y = \phi(X) | Y = 1)$$

si la verdadera clase era 1

$$\text{FPR} = \mathbb{P}(Y \neq \phi(X) | Y = 0)$$

si la verdadera clase era 0



## Area Under the Curve (AUC)

criterios para comparar clasificadores

El AUC es el área bajo la curva ROC.

## Equal Error Rate (EER)

El EER es el error para el cuál los errores  $\text{FPR} = 1 - \text{TPR}$ .

# Métricas para clases desbalanceadas

## Precision-Recall

cuando hay un error que es mucho mas grave que el otro.

Cuando el costo de las clases está desbalanceado se utilizan las métricas *Precision* y *Recall*.

- $\text{Precision} = \mathbb{P}(Y = \phi(X) | \phi(X) = 1)$ . *Precision* se utiliza cuando los falsos positivos tiene consecuencias graves. Por ejemplo, diagnosticar erróneamente una enfermedad a una persona sana
- $\text{Recall (TPR)} = \mathbb{P}(Y = \phi(X) | Y = 1)$ . *Recall* se utiliza cuando cuando los falsos negativos tiene consecuencias graves. Por ejemplo, en la detección de fraudes, no detectar una transacción fraudulenta.

# Métricas para clases desbalanceadas

## Precision-Recall

Cuando el costo de las clases está desbalanceado se utilizan las métricas *Precision* y *Recall*.

- $\text{Precision} = \mathbb{P}(Y = \phi(X) | \phi(X) = 1)$ . *Precision* se utiliza cuando los falsos positivos tiene consecuencias graves. Por ejemplo, diagnosticar erróneamente una enfermedad a una persona sana
- $\text{Recall (TPR)} = \mathbb{P}(Y = \phi(X) | Y = 1)$ . *Recall* se utiliza cuando cuando los falsos negativos tiene consecuencias graves. Por ejemplo, en la detección de fraudes, no detectar una transacción fraudulenta.

## F1-score

Cuando la proporción de las clases está desbalanceada se utiliza la métrica F1:

$$F_1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

# Outline

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica**
- 4 Linear Discriminant Analysis
- 5 K-Vecinos más cercanos
- 6 Support Vector Machines
- 7 Árboles de decisión

# Regresión Logística Categórica ( $k$ clases)

## Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \dots, k-1\} \\ \frac{1}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

# Regresión Logística Categórica ( $k$ clases)

## Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \dots, k-1\} \\ \frac{1}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

## Softmax

gasta mas parametros, es mas o menos lo mismo, pero perdemos la identificabilidad

$$\hat{P}(y|x) = \frac{e^{w_y^T x + b_y}}{\sum_{j=1}^k e^{w_j^T x + b_j}}, \quad y \in \{1, \dots, k\}$$



# Regresión Logística Categórica ( $k$ clases)

## Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \dots, k-1\} \\ \frac{1}{1 + \sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

## Softmax

$$\hat{P}(y|x) = \frac{e^{w_y^T x + b_y}}{\sum_{j=1}^k e^{w_j^T x + b_j}}, \quad y \in \{1, \dots, k\}$$

## Riesgo empírico

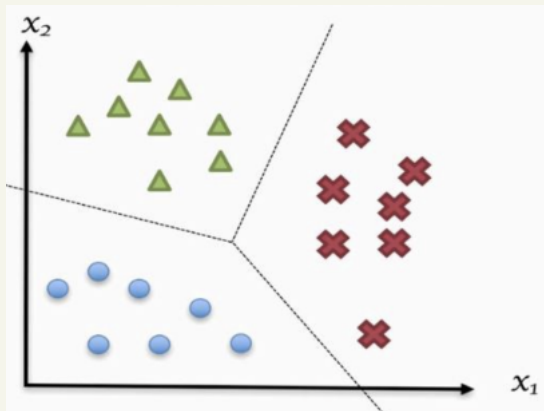
$$\frac{1}{n} \sum_{i=1}^n \ell(X_i, Y_i) = \frac{1}{n} \sum_{i=1}^n \left[ \log \left( \sum_{j=1}^k e^{w_j^T X_i + b_j} \right) - (w_{Y_i}^T X_i + b_{Y_i}) \right]$$

# Regresión Softmax

## Elección del máximo

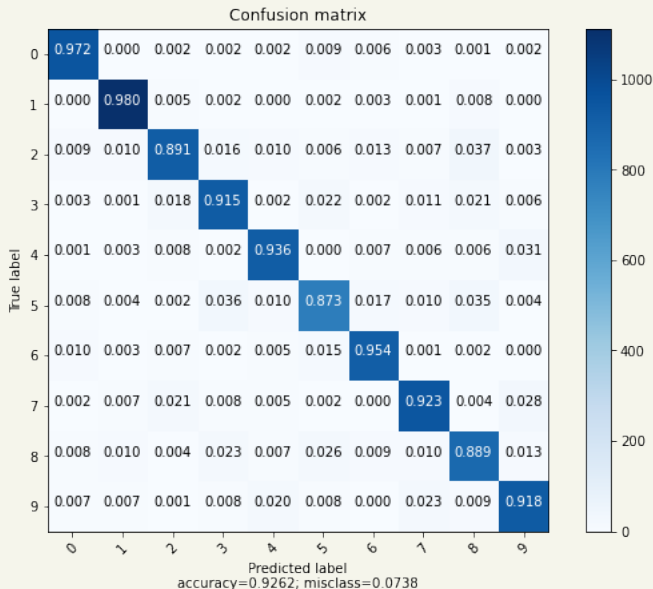
$$\arg \max_y \hat{P}(y|x) = \arg \max_y w_y^T x + b_y$$

Se separa con hiperplanos!









# Confusion Matrix







capacidad de confundirse un 0 con un 1, un 0 con un 2. para ver donde suelo confundirme mas.






# Generalización del F1 score

		Predicted		
		 Airplane	 Boat	 Car
Actual	 Airplane	2	1	0
	 Boat	0	1	0
	 Car	1	2	3




# Generalización del F1 score

		Predicted		
		 Airplane	 Boat	 Car
Actual	 Airplane	2	1	0
	 Boat	0	1	0
	 Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
 Airplane	2	1	1	0.67	0.67	$2 * (0.67 * 0.67) / (0.67 + 0.67)$ <b>= 0.67</b>
 Boat	1	3	0	0.25	1.00	$2 * (0.25 * 1.00) / (0.25 + 1.00)$ <b>= 0.40</b>
 Car	3	0	3	1.00	0.50	$2 * (1.00 * 0.50) / (1.00 + 0.50)$ <b>= 0.67</b>

# Generalización del F1 score

		Predicted		
		 Airplane	 Boat	 Car
Actual	 Airplane	2	1	0
	 Boat	0	1	0
	 Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
 Airplane	2	1	1	0.67	0.67	$2 * (0.67 * 0.67) / (0.67 + 0.67)$ <b>= 0.67</b>
 Boat	1	3	0	0.25	1.00	$2 * (0.25 * 1.00) / (0.25 + 1.00)$ <b>= 0.40</b>
 Car	3	0	3	1.00	0.50	$2 * (1.00 * 0.50) / (1.00 + 0.50)$ <b>= 0.67</b>

$$\text{Macro-}F_1 = \frac{0.67 + 0.40 + 0.67}{3} = 0.58$$

objetivo: evaluar desempeño con un solo número

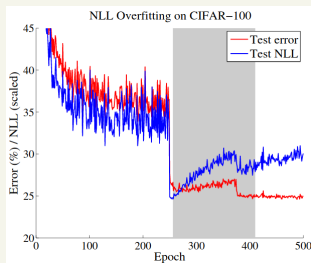
## Calibración

**Si valido hiperparámetros con respecto a la probabilidad de error, ¿la salida siguen siendo probabilidades?**

Para cada par  $(x, y)$ , existe  $0 \leq \zeta \leq \log |\mathcal{Y}|$  tal que

$$-\log \hat{P}(y|x) = -(\mathbf{z}(x))_y + \log \left( \mathbf{1}^T \cdot e^{\mathbf{z}(x)} \right) = \max_{y' \in \mathcal{Y}} (\mathbf{z}(x))_{y'} - (\mathbf{z}(x))_y + \zeta$$

Esta concentración de probabilidades es buena para acercarme al clasificador bayesiano, pero puede descalibrar la interpretación probabilística de  $\hat{P}$ .



*Guo et al. 2017: "On Calibration of Modern Neural Networks".*

# Calibración

## Temperature Scaling

Se soluciona con la inclusión de un nuevo parámetro (o hiper)  $T > 0$ :

lo desparrama, lo suaviza

$$\hat{P}(y|x) = \frac{e^{(z(x))_y / T}}{\mathbf{1}^T \cdot e^{z(x) / T}}$$

Es importante mirar la cross-entropy en la etapa de validación!

Dataset	Model	Uncalibrated	Temp. Scaling
Birds	ResNet 50	0.9786	<b>0.8792</b>
Cars	ResNet 50	0.5488	0.5311
CIFAR-10	ResNet 110	0.3285	0.2102
CIFAR-10	ResNet 110 (SD)	0.2959	0.1718
CIFAR-10	Wide ResNet 32	0.3293	0.2283
CIFAR-10	DenseNet 40	0.2228	<b>0.1750</b>
CIFAR-10	LeNet 5	0.4688	0.459
CIFAR-100	ResNet 110	1.4978	<b>1.0442</b>
CIFAR-100	ResNet 110 (SD)	1.1157	<b>0.8613</b>
CIFAR-100	Wide ResNet 32	1.3434	<b>1.0565</b>
CIFAR-100	DenseNet 40	1.0134	0.9026
CIFAR-100	LeNet 5	1.6639	<b>1.6560</b>
ImageNet	DenseNet 161	0.9338	0.8885
ImageNet	ResNet 152	0.8961	<b>0.8657</b>
SVHN	ResNet 152 (SD)	0.0842	<b>0.0821</b>
20 News	DAN 3	0.7949	0.7387
Reuters	DAN 3	0.102	0.0994
SST Binary	TreeLSTM	0.3367	<b>0.2739</b>
SST Fine Grained	TreeLSTM	1.1475	1.1168

Guo et al. 2017: "On Calibration of Modern Neural Networks".



# Outline

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica
- 4 Linear Discriminant Analysis**
- 5 K-Vecinos más cercanos
- 6 Support Vector Machines
- 7 Árboles de decisión

# Modelos Discriminativos y Generativos

## Clasificación de Algoritmos

los que vimos la clase pasada

- **Modelos Discriminativos:** Modelan la dist. condicional  $\hat{P}(y|x)$ .

- **Modelos Generativos:** Modelan la dist. conjunta  $\hat{P}(x, y)$ .  
modelan toda la conjunta

Los modelos generativos permiten generar datos sintéticos!

permite generar tus propios datos, generas muestras nuevas

# Modelos Discriminativos y Generativos

## Clasificación de Algoritmos

- **Modelos Discriminativos:** Modelan la dist. condicional  $\hat{P}(y|x)$ .
- **Modelos Generativos:** Modelan la dist. conjunta  $\hat{P}(x, y)$ .

Los modelos generativos permiten generar datos sintéticos!

## Linear Discriminant Analysis (LDA)

todas con la misma  
covarianza

$$Y \sim \text{Cat}(\{c_1, \dots, c_K\}), \quad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$$



# Linear Discriminant Analysis

## Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^K c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2} |\Sigma|^{1/2}}$$
$$\hat{P}(y|x) = \frac{e^{\mu_y^T \Sigma^{-1} x - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log(c_y)}}{\sum_{k=1}^K e^{\mu_k^T \Sigma^{-1} x - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(c_k)}}$$

## Relación con Regresión Logística

el modelo de LDA es mas rigido, pide un poco mas de condiciones que se tiene que cumplir

Si  $w_y = \Sigma^{-1} \mu_y$  y  $b_y = -\frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log(c_y)$ ,  $\hat{P}(y|x)$  es el softmax. LDA utiliza hipótesis más fuertes ya que no solo asume  $\hat{P}(y|x)$  softmax, sino también  $\hat{p}(x)$  mezcla de gaussianas.

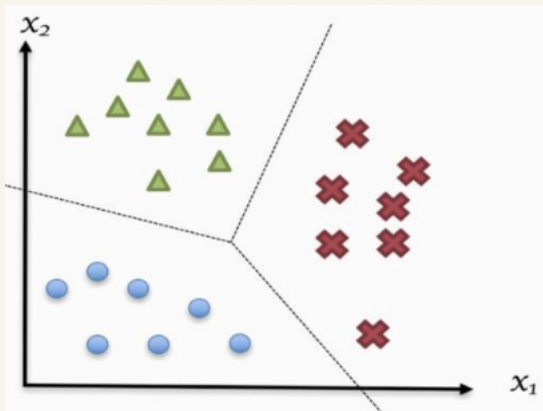
# Regresión Softmax

## Elección del máximo

la frontera vuelve a quedar lineal

$$\arg \max_y \hat{P}(y|x) = \arg \max_y w_y^T x + b_y$$

Se separa con hiperplanos!



# Estimación Insegada de Parámetros

## Estimadores

entrenamiento

$$\mathcal{D}_k = \{x_i : 1 \leq i \leq n \wedge y_i = k\}$$

armo distintos conjuntos de datos, cada uno correspondiente a una etiqueta

$$c_k = \frac{\#(\mathcal{D}_k)}{n}$$

la proporción de muestras en ese conjunto: positivos/totales

$$\mu_k = \frac{1}{\#(\mathcal{D}_k)} \sum_{x \in \mathcal{D}_k} x$$

la media la estimamos con el promedio

$$\Sigma_k = \frac{1}{\#(\mathcal{D}_k) - 1} \sum_{x \in \mathcal{D}_k} (x - \mu_k)(x - \mu_k)^T$$

usamos los estimadores insegados

$$\Sigma = \frac{1}{n - K} \sum_{k=1}^K (\#(\mathcal{D}_k) - 1) \Sigma_k$$

como son todas iguales tengo que hacer este paso adicional

# Quadratic Discriminant Analysis

que pasa si asumo que las covarianzas son diferentes?

## Quadratic Discriminant Analysis (QDA)

$$Y \sim \text{Cat}(\{c_1, \dots, c_K\}), \quad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

tendencia al overfitting, pero si tengo muchas muestras no es tanto problema

# Quadratic Discriminant Analysis

## Quadratic Discriminant Analysis (QDA)

$$Y \sim \text{Cat}(\{c_1, \dots, c_K\}), \quad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

### Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^K c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2} |\Sigma_k|^{1/2}}$$

$$\hat{P}(y|x) = \frac{e^{-\frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y) + \log(c_y) - \frac{\log |\Sigma_y|}{2}}}{\sum_{k=1}^K e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + \log(c_k) - \frac{\log |\Sigma_k|}{2}}}$$



# Quadratic Discriminant Analysis

## Quadratic Discriminant Analysis (QDA)

$$Y \sim \text{Cat}(\{c_1, \dots, c_K\}), \quad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

### Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^K c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2} |\Sigma_k|^{1/2}}$$

$$\hat{P}(y|x) = \frac{e^{-\frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y) + \log(c_y) - \frac{\log |\Sigma_y|}{2}}}{\sum_{k=1}^K e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + \log(c_k) - \frac{\log |\Sigma_k|}{2}}}$$

Elección del máximo: NO ES LINEAL, ES CUADRÁTICO

$$\arg \max_y -\frac{1}{2}(x - \mu_y)^T \Sigma_y^{-1}(x - \mu_y) + \log(c_y) - \frac{\log |\Sigma_y|}{2}$$

hasta aca ejercicio 7

# Outline

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica
- 4 Linear Discriminant Analysis
- 5 K-Vecinos más cercanos**
- 6 Support Vector Machines
- 7 Árboles de decisión

# Modelos Paramétricos y No Paramétricos

## Clasificación de Algoritmos

- **Modelos Paramétricos:** Asumen conocimiento parcial sobre la distribución, indexándola por parámetros.
- **Modelos No Paramétricos:** No se asume una estructura a priori para la distribución.

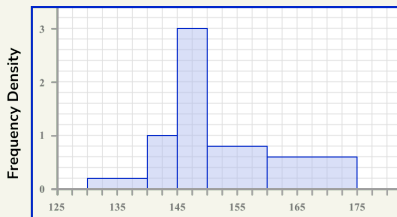
# Modelos Paramétricos y No Paramétricos

## Clasificación de Algoritmos

- **Modelos Paramétricos:** Asumen conocimiento parcial sobre la distribución, indexándola por parámetros.
- **Modelos No Paramétricos:** No se asume una estructura a priori para la distribución.

## Histograma

El histograma asume una densidad constante por regiones. En cada región asigna  $\hat{p}(x) = \frac{K}{n \cdot V}$  donde  $n$  es la cantidad de muestras totales,  $K$  la cantidad de muestras en dicha región y  $V$  el volumen de la región.



# K-Vecinos más cercanos (KNN)

## Adaptando el concepto a aprendizaje supervisado

Asumiendo que  $\hat{P}(y) = \frac{N_y}{n}$  con  $N_y$  el número de muestras de la clase  $y$ , y que (en cada región)  $\hat{p}(x|y) = \frac{K_y}{N_y \cdot V}$  con  $K_y$  la cantidad de muestras que caen en la región de la clase  $y$ , se obtiene:

$$\hat{P}(y|x) = \frac{\hat{p}(x|y)\hat{P}(y)}{\sum_{i=1}^K \hat{p}(x|i)\hat{P}(i)} = \frac{K_y}{K}$$

las muestras del bin de  $x$

Es decir, la proporción de muestras de la clase  $y$  en la región.

## K-Vecinos más cercanos (KNN)

### Adaptando el concepto a aprendizaje supervisado

Asumiendo que  $\hat{P}(y) = \frac{N_y}{n}$  con  $N_y$  el número de muestras de la clase  $y$ , y que (en cada región)  $\hat{p}(x|y) = \frac{K_y}{N_y \cdot V}$  con  $K_y$  la cantidad de muestras que caen en la región de la clase  $y$ , se obtiene:

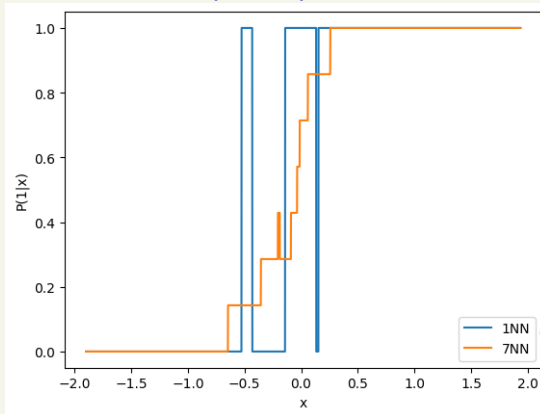
$$\hat{P}(y|x) = \frac{\hat{p}(x|y)\hat{P}(y)}{\sum_{i=1}^K \hat{p}(x|i)\hat{P}(i)} = \frac{K_y}{K}$$

Es decir, la proporción de muestras de la clase  $y$  en la región.

### K-Vecinos más cercanos

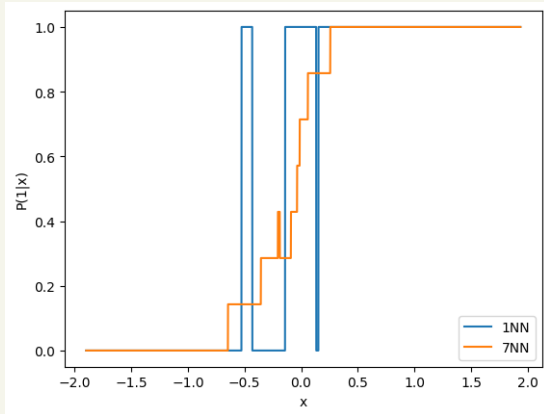
KNN fija el valor de vecinos  $K$  y en base a esto define las regiones. Por ejemplo, la región utilizada para computar un *feature*  $x$  es la región centrada en  $x$  que posee  $K$  muestras (las  $K$  más cercanas a  $x$ ).

# K-Vecinos más cercanos (KNN)



el vecino mas ercano  
los 7 vecinos mas cercanos

## K-Vecinos más cercanos (KNN)



### Elección del máximo

puedo elegir clasificar por mayoría: "si la mayoría de los 7 vecinos mas cercanos son de la clase X, entonces el que quiero clasificar es de esa misma clase"

Notar que para quedarse con el máximo de  $\hat{P}(y|x)$  no hace falta computarla. Simplemente se clasifica según sus  $K$  vecinos más cercanos, por mayoría.



# Outline

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica
- 4 Linear Discriminant Analysis
- 5 K-Vecinos más cercanos
- 6 Support Vector Machines**
- 7 Árboles de decisión

# Support Vector Machines

## Clases linealmente separables

estudia clasificación binaria, y propone que existe al menos una recta que puede separar las clases

$>0$  : clase 1

$<0$  : clase 0

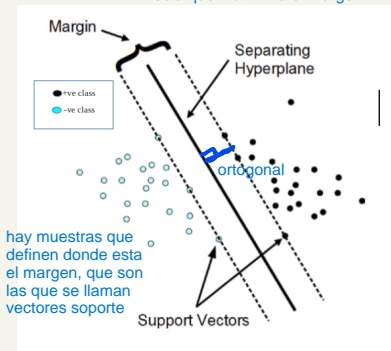
Sea la clasificación binaria  $y \in \{-1, 1\}$  y  $z(x) = w^T \cdot x + b = 0$  su frontera de decisión. Decimos que las clases son linealmente separables, si existen  $w$  y  $b$  tales que  $y \cdot z(x) > 0$  para todo  $(x, y) \in \mathcal{D}_n$  (set de entrenamiento). Llamamos  $f_i(w, b) = y_i z(x_i) > 0$  con  $1 \leq i \leq n$ .

# Support Vector Machines

## Clases linealmente separables

Sea la clasificación binaria  $y \in \{-1, 1\}$  y  $z(x) = w^T \cdot x + b = 0$  su frontera de decisión. Decimos que las clases son linealmente separables, si existen  $w$  y  $b$  tales que  $y \cdot z(x) > 0$  para todo  $(x, y) \in \mathcal{D}_n$  (set de entrenamiento). Llamamos  $f_i(w, b) = y_i z(x_i) > 0$  con  $1 \leq i \leq n$ .

recta que maximiza el margen.



- $w$  es ortogonal a la frontera y por lo tanto  $w \parallel (x - x_*)$  con  $x_*$  la proyección ortogonal de  $x$  sobre la frontera.

$$|w^T(x - x_*)| = \|w\| \|x - x_*\|$$

- Dado que  $x_*$  está sobre la frontera,  $w^T(x - x_*) = z(x)$  y por lo tanto:

$$d(x_i) = \|x_i - x_*\| = \frac{|z(x_i)|}{\|w\|} = \frac{y_i \cdot z(x_i)}{\|w\|}$$

# Support Vector Machines

## Margen

si  $x^*$  esta en la frontera:  $z(x^*) + 0$   
entonces tiene q cumplir la eq de la frontera  $w.T * x^* + b = 0$   
 $w.T(x - x^*) = (w.T * x + b) - (w.T * x^* + b) = z(x)$

Se define el margen unilateral como criterio de peor caso:

$$m(w, b) = \min_{1 \leq i \leq n} \frac{y_i(w^T \cdot x_i + b)}{\|w\|} = \frac{1}{\|w\|} \min_{1 \leq i \leq n} f_i(w, b) = \frac{f_k(w, b)}{\|w\|}$$

con  $k$  un índice óptimo (función de  $w$  y  $b$ ). Por lo tanto, el problema a resolver es maximizar el margen:  $\max_{w, b} m(w, b)$  st.  $f_i(w, b) > 0$  para  $i = 1, \dots, n$ .

# Support Vector Machines

## Margen

Se define el margen unilateral como criterio de peor caso:

$$m(w, b) = \min_{1 \leq i \leq n} \frac{y_i(w^T \cdot x_i + b)}{\|w\|} = \frac{1}{\|w\|} \min_{1 \leq i \leq n} f_i(w, b) = \frac{f_k(w, b)}{\|w\|}$$

con  $k$  un índice óptimo (función de  $w$  y  $b$ ). Por lo tanto, el problema a resolver es maximizar el margen:  $\max_{w, b} m(w, b)$  st.  $f_i(w, b) > 0$  para  $i = 1, \dots, n$ .

## Escala

Sea  $\alpha > 0$ , está claro la decisión  $z(x) \geq 0$  no se ve afectada si reescalamos los parámetros  $w \leftarrow \alpha w$  y  $b \leftarrow \alpha b$ . Esto mismo ocurre con el margen  $m(\alpha w, \alpha b) = m(w, b)$ . Con lo cual no se pierde generalidad al asumir  $f_k(w, b) = 1$ . Luego  $m(w, b) = \frac{1}{\|w\|}$  y  $f_i(w, b) \geq 1$  para todo  $1 \leq i \leq n$ .

en vez de maximizar el margen, minimizo la inversa  
buscamos el alfa q haga que se cumpla esto

Las muestras en las que  $f_i(w, b) = 1$  se denominan vectores soporte.

# Support Vector Machines

## Problema de optimización

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad (\forall 1 \leq i \leq n)$$

una mala muestra me puede influenciar mucho la frontera de decision

# Support Vector Machines

## Problema de optimización

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad (\forall 1 \leq i \leq n)$$

## Relajando los márgenes

Mitigar problemas con outliers. Sea  $C \geq 0$ ,

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.t.} \quad \begin{cases} y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad (\forall 1 \leq i \leq n)$$

tiene q ser chico  
esto indica cuanto pase la linea q no tendria q haber pasado

permite que las muestras puedan penetrar los bordes de los vectores soporte

# Support Vector Machines

## Problema de optimización

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad (\forall 1 \leq i \leq n)$$

## Relajando los márgenes

Mitigar problemas con outliers. Sea  $C \geq 0$ ,

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.t.} \quad \begin{cases} y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad (\forall 1 \leq i \leq n)$$

## Generalización a fronteras no lineales

Este método es generalizable a diferentes fronteras  $z(x) = w^T \phi(x) + b$ . La función  $k(x_1, x_2) = \phi^T(x_1) \phi(x_2)$  recibe el nombre de Kernel, siendo el más conocido el gaussiano o rbf:  $k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$ .



# Support Vector Machines

## Generalización a $K$ -clases

- *one-vs-one*: Se toman todas las combinaciones de pares de clases (son  $\frac{K(K-1)}{2}$ ) y se entrenan clasificadores binarios. Se clasifica seleccionando a la clase con más *votos*.
- *one-vs-the-rest*: Se entrenan  $K$  clasificadores binarios, donde cada uno toma una clase como positiva y el resto como negativa. Se clasifica según  $\arg \max_k w_k^T \phi(x) + b_k$ .

# Support Vector Machines

## Generalización a $K$ -clases

- *one-vs-one*: Se toman todas las combinaciones de pares de clases (son  $\frac{K(K-1)}{2}$ ) y se entrenan clasificadores binarios. Se clasifica seleccionando a la clase con más *votos*.
- *one-vs-the-rest*: Se entrenan  $K$  clasificadores binarios, donde cada uno toma una clase como positiva y el resto como negativa. Se clasifica según  $\arg \max_k w_k^T \phi(x) + b_k$ .

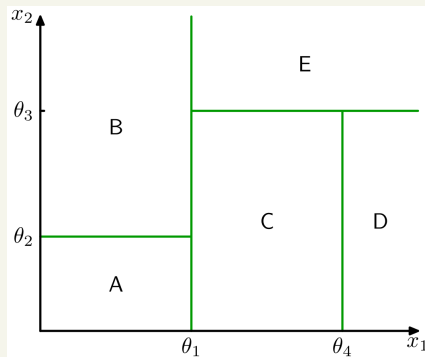
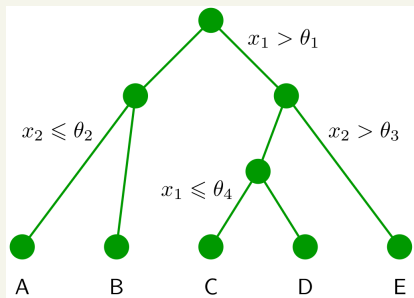
## Generalización a Regresión

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad |w^T x_i + b - y_i| \leq \epsilon \quad (\forall 1 \leq i \leq n)$$

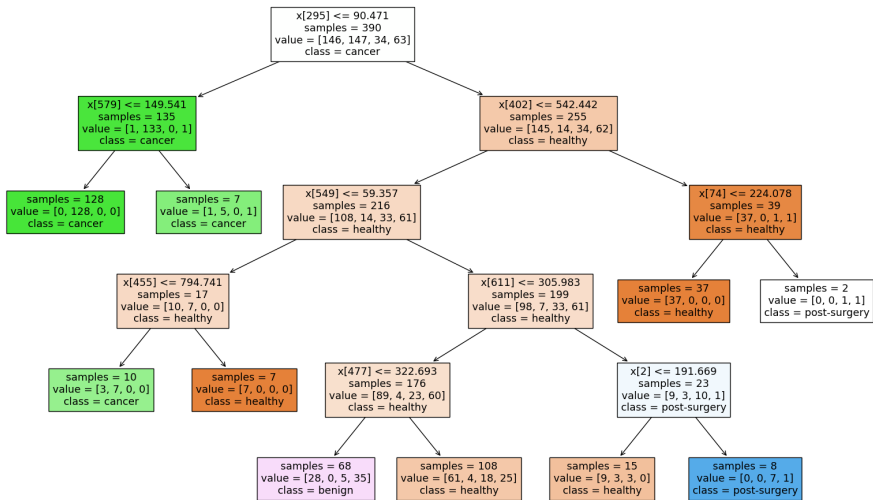
# Outline

- 1 Introducción al problema de clasificación
- 2 Regresión Logística Binaria
- 3 Regresión Logística Categórica
- 4 Linear Discriminant Analysis
- 5 K-Vecinos más cercanos
- 6 Support Vector Machines
- 7 Árboles de decisión**

# CART: Classification and Regression Trees



# Árboles de decisión



# Árboles de decisión

## Modelado matemático por nodo

Llamamos:

- $Q_m$  al conjunto de datos en el nodo  $m$ .
- $Q_m^L(j_m, t_m) = \{(x, y) \in Q_m : x_{j_m} \leq t_m\}$ .
- $Q_m^R(j_m, t_m) = \{(x, y) \in Q_m : x_{j_m} > t_m\}$ .
- $H(Q_m)$  a la función impureza del conjunto  $Q_m$ .
- $G_m(j_m, t_m) = \frac{|Q_m^L(j_m, t_m)|}{|Q_m|} H(Q_m^L(j_m, t_m)) + \frac{|Q_m^R(j_m, t_m)|}{|Q_m|} H(Q_m^R(j_m, t_m))$ .
- Busco para cada nodo  $(j_m^*, t_m^*) = \arg \min_{j_m, t_m} G_m(j_m, t_m)$

# Árboles de decisión

## Modelado matemático por nodo

Llamamos:

- $Q_m$  al conjunto de datos en el nodo  $m$ .
- $Q_m^L(j_m, t_m) = \{(x, y) \in Q_m : x_{j_m} \leq t_m\}$ .
- $Q_m^R(j_m, t_m) = \{(x, y) \in Q_m : x_{j_m} > t_m\}$ .
- $H(Q_m)$  a la función impureza del conjunto  $Q_m$ .
- $G_m(j_m, t_m) = \frac{|Q_m^L(j_m, t_m)|}{|Q_m|} H(Q_m^L(j_m, t_m)) + \frac{|Q_m^R(j_m, t_m)|}{|Q_m|} H(Q_m^R(j_m, t_m))$ .
- Busco para cada nodo  $(j_m^*, t_m^*) = \arg \min_{j_m, t_m} G_m(j_m, t_m)$

## Funciones impurezas habituales

Sea  $p_{m,k}$  la proporción de muestras de la clase  $k$  en el nodo  $m$ :

- Gini:  $H(Q_m) = \sum_k p_{m,k} (1 - p_{m,k})$ .
- Entropía:  $H(Q_m) = \sum_k -p_{m,k} \log_2(p_{m,k})$ .

# Árboles de decisión

## Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Otras (ver documentación).



# Árboles de decisión

## Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Otras (ver documentación).

## Variables Categóricas

Dada la característica binaria de los árboles, estas variables se codifican en estructuras binarias (ej. *one hot encoding*).

# Árboles de decisión

## Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Otras (ver documentación).

## Variables Categóricas

Dada la característica binaria de los árboles, estas variables se codifican en estructuras binarias (ej. *one hot encoding*).

## Importancia de cada Feature

La *Gini Importance* se define como la disminución total de la impureza del nodo, ponderada por la probabilidad de llegar a ese nodo (normalizado).

# Árboles de decisión

## Problemas de regresión

Modelando la función regresión como constante por regiones, este método puede ser adaptado. Como función impureza suele usarse el error cuadrático medio:

$$H(Q_m) = \sum_{(x,y) \in Q_m} (y - \bar{y}_m)^2$$

donde  $\bar{y}_m$  es el promedio de las  $y$  en  $Q_m$ .

# Árboles de decisión

## Problemas de regresión

Modelando la función regresión como constante por regiones, este método puede ser adaptado. Como función impureza suele usarse el error cuadrático medio:

$$H(Q_m) = \sum_{(x,y) \in Q_m} (y - \bar{y}_m)^2$$

donde  $\bar{y}_m$  es el promedio de las  $y$  en  $Q_m$ .

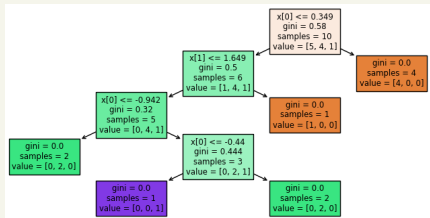
## Podado: Regularización

Sea  $T$  un árbol determinado (sin condiciones de parado fuertes),  $L(T)$  su respectivo conjunto de hojas y  $\alpha$  el parámetro de complejidad. Se denomina medida de costo-complejidad a

$$H_\alpha(T) = \sum_{m \in L(T)} \frac{|Q_m|}{n} \cdot H(Q_m) + \alpha \cdot |L(T)|$$

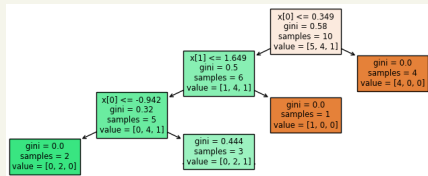
La poda se basa en quedarse con el subárbol de menor costo-complejidad.

$T_1$



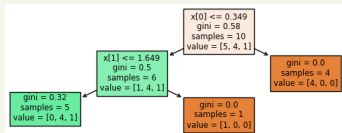
$$H_{\alpha}(T) = 0 + 5\alpha$$

$T_2$



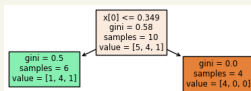
$$H_{\alpha}(T) = 0.13 + 4\alpha$$

$T_3$



$$H_{\alpha}(T) = 0.16 + 3\alpha$$

$T_4$



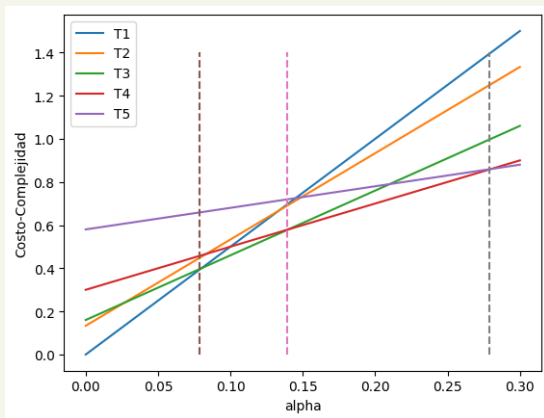
$$H_{\alpha}(T) = 0.3 + 2\alpha$$

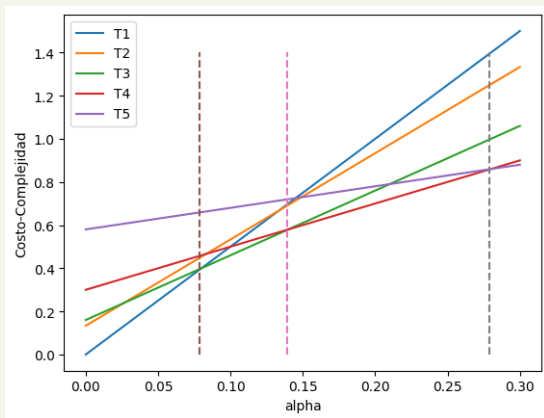
$T_5$



$$H_{\alpha}(T) = 0.58 + \alpha$$

# Poda





- La cantidad de candidatos a óptimos es menor a la cantidad de subárboles (el  $T_2$  nunca es el de menor costo-complejidad).
- El subárbol se elige por validación (típicamente sobre el error de clasificación) comparando todos los casos posibles (en este caso 4 candidatos).

# Bosques aleatorios

## Bagging

El problema de los árboles de decisión es el *overfitting*. Las condiciones de stop y la poda ayudan a combatirlo, pero muchas veces no son suficiente. Es por eso que surge *Bagging*: Entrenar múltiples algoritmos y decidir por mayoría o promedio (en clasificación o regresión respectivamente). Un algoritmo de múltiples árboles se llama bosque.

## ¿Por que promediar?

- El promedio mantiene la esperanza y reduce la varianza en muestras i.i.d:

$$\mathbb{E} \left[ \frac{1}{B} \sum_{b=1}^B Z_b \right] = \mu, \quad \text{var} \left( \frac{1}{B} \sum_{b=1}^B Z_b \right) = \frac{\sigma^2}{B}$$

- En clasificación, si se piensan etiquetas en codificación *one-hot*, promediar para luego elegir el máximo equivale a elegir la respuesta mayoritaria.



## Bosques aleatorios

Se desea entrenar varios algoritmos (de manera que sean variados). Para asegurar ésto, se toman dos decisiones:

### No usar todos los features

En lugar de usar todos los  $d_x$  *features*, para asegurar variedad en los árboles, para cada nodo se eligen al azar  $\sqrt{d_x}$  *features*.

## Bosques aleatorios

Se desea entrenar varios algoritmos (de manera que sean variados). Para asegurar ésto, se toman dos decisiones:

### No usar todos los features

En lugar de usar todos los  $d_x$  features, para asegurar variedad en los árboles, para cada nodo se eligen al azar  $\sqrt{d_x}$  features.

### Bootstrap

Generar  $B$  conjuntos de datos diferentes del mismo tamaño que el dataset original  $n$ . Para esto, se utiliza una técnica llamada Bootstrap: Se eligen al azar  $n$  datos del conjunto *con reposición* y se arma cada conjunto Bootstrap, de manera que la probabilidad que un dato no esté en el conjunto es del  $\approx 37\%$  :

$$\left(1 - \frac{1}{n}\right)^n \rightarrow e^{-1}$$