

# Introducción a ciencia de datos – 2023

Emilien Joly – Ayu. (por definir)

CIMAT - Semestre Agosto 2023 - Diciembre 2023

# Horarios y enlaces

**Profesor:** Emilien Joly (emilien.joly@cimat.mx)

**Ayudante:** (Por definir) (—@ciamat.mx)

Fecha de inicio : **Martes 15 de agosto**

**Horarios de clase:** martes y jueves, 11:00-12:20


**Horarios de prácticas:** ?

Fecha de fin de clases : **jueves 23 de noviembre**


# Información útil

La información del curso de EM se encontrará en la pagina web :

► <https://joly415.perso.math.cnrs.fr/EnseignementEG.htm>

Emilien Joly

---

Home Research/Publications Teaching CV 

---

**Address :**  
Bureau H105, Bâtiment H  
Centro de Investigación en  
Matemáticas  
De Jalisco SA, Valenciana,  
36240 Guanajuato, Gto.

**Email :**  
[emilien.joly@cimat.mx](mailto:emilien.joly@cimat.mx)

**Phone :**  
(+55) 473-738-09-27

**Teaching at CIMAT**

- A week of pre-course about the fundamentals of probability theory  
Program of the mini-course : [here](#)  
[Day 1](#), [Day 2](#), [Day 3](#), [Day 4](#), [Day 5](#) and [\(Exercises\)](#) Day 5.
- 30 sessions around Empirical Processes and Concentration inequalities, Doctoral level, 2018-2020  
See details here : [2018](#) [2020](#)
- 30 sessions on the fundamentals of rigorous mathematical statistics, Master level, 2018-2020

**Teaching at Université Paris-Sud**

- Exercise sessions, freshman engineering school class, at IUT d'Orsay, 2012-2015
- Exercise sessions, senior year class, at faculté des sciences d'Orsay, 2012-2015
- Exercise sessions, Licence 1, at université Paris Ouest Nanterre, 2016-2017

**Miscellaneous**

- Member of the organizing committee of [Mathematic Park](#). Most of the videos are on [Youtube](#).
- Jury for the [International Tournament of Young Mathematicians](#), at école Polytechnique, 2012
- Oral examinations in classe préparatoire, at lycée Saint Louis, Paris, 2010-2012

► El temario del curso : [aquí](#)

## Introducción a ciencia de datos

Agosto-Diciembre 2023

**Profesor:** Emilien Joly (ext. 531, of. H104, [emilien.joly@cimat.mx](mailto:emilien.joly@cimat.mx))

**Ayudante:** A definir ([xxx@cimat.mx](mailto:xxx@cimat.mx))

**Horarios de clase:** A definir, 11:00-12:20

**Horarios de prácticas:** a definir, 11:00-12:20

# Tareas y exámenes

## Fechas de Tareas:

- ▶ Tarea 1: Inicio 22 agosto → Entrega 31 agosto
- ▶ Tarea 2: Inicio 5 septiembre → Entrega 14 septiembre
- ▶ Examen 1 : Viernes 15 septiembre.
- ▶ Tarea 3: Inicio 19 septiembre → Entrega 28 septiembre
- ▶ Tarea 4: Inicio 3 octubre → Entrega 12 octubre
- ▶ Tarea 5: Inicio 15 octubre → Entrega 26 octubre
- ▶ Examen 2 : Viernes 27 octubre.
- ▶ Tarea 6: Inicio 31 octubre → Entrega 9 noviembre
- ▶ Tarea 7: Inicio 14 noviembre → Entrega 23 noviembre
- ▶ Examen final : Semana 27 noviembre - 1 diciembre.

## Calificación:

Las tareas cuentan por **1/3** de la calificación final.

Los exámenes parciales cuentan por **1/6** cada uno.

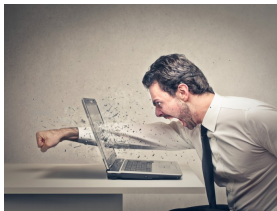
El examen final cuenta por **1/3**.

# Objetivos

1. Entender las principales problemáticas de los datos modernos y su análisis.
2. Dominar las técnicas estadísticas de agrupamiento y clasificación de datos masivos.
3. Entender y usar las técnicas asociadas a reducción de dimensionalidad y regularización.
4. Conocer los elementos de aprendizaje máquina y minería de datos para el análisis de datos masivos.
5. Obtener las bases para desarrollar técnicas raras/dispersas.

¿Qué NO es la ciencia de datos?

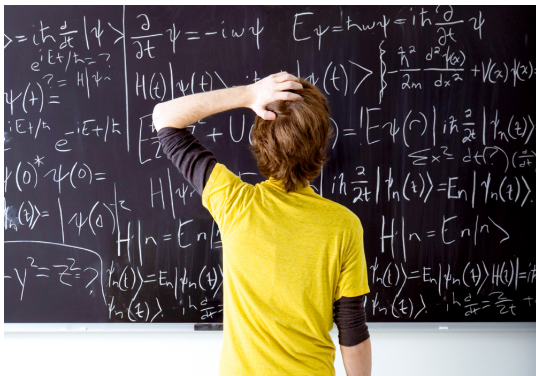
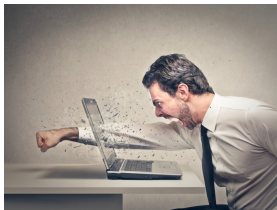
¿Qué NO es la ciencia de datos?





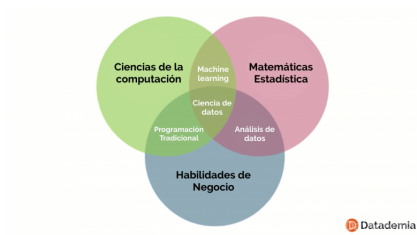


# ¿Qué NO es la ciencia de datos?

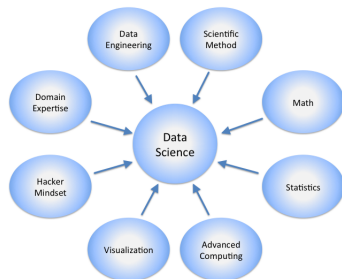
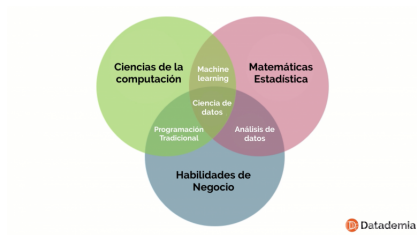


¿Qué es la ciencia de datos?

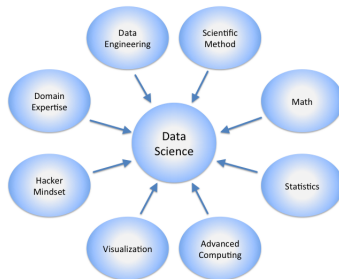
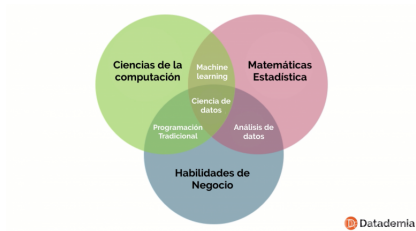
# ¿Qué es la ciencia de datos?



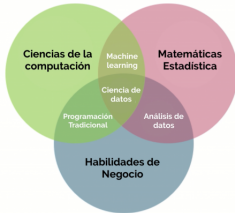
# ¿Qué es la ciencia de datos?



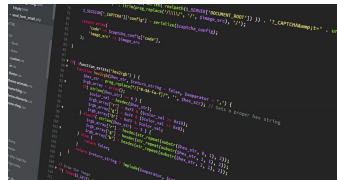
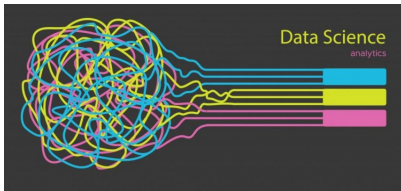
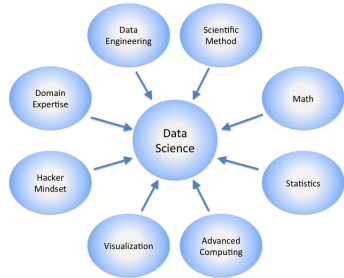
# ¿Qué es la ciencia de datos?



# ¿Qué es la ciencia de datos?



Datademia



# Contenido del curso

1. Métodos de clasificación.
2. Métodos de búsqueda de estructura
3. Métodos sparse
4. Agrupamiento de datos
5. Técnicas de aprendizaje maquina

# Contenido del curso

1. Métodos de clasificación.
2. Métodos de búsqueda de estructura
3. Métodos sparse
4. Agrupamiento de datos
5. Técnicas de aprendizaje maquina

Se verán algunos de los algoritmos más populares como:

- ▶ K-medias
- ▶ SVD (descomposición en valores singulares)
- ▶ Complementación de matrices de rango bajo
- ▶ Compressed Sensing (Sparsity)
- ▶ Difusion maps
- ▶ Descenso por gradiente
- ▶ Aprendizaje "online"
- ▶ Agrupamiento espectral
- ▶ Detección de comunidades (Stochastic Block Model)
- ▶ Estadísticas en altas dimensiones ( $n \sim p$ )



# 1.Métodos de clasificación

## 1.1.Curso del 15 de agosto

## 1.1.1. ¿Que es la clasificación?



## 1.1.1. ¿Que es la clasificación?

### Preguntas naturales:

- ▶ ¿Que son los objetos a clasificar?
- ▶ ¿En que espacio(s) viven?
- ▶ ¿Es clasificación supervisada?
- ▶ ¿Cómo medir una buena/mala tarea de clasificación?
- ▶ ¿Cómo optimizar la tarea de clasificación?
- ▶ ¿Cómo automatizar la tarea de clasificación?

## 1.1.2. Clasificación óptima

### Los datos:

Para una computadora, un dato es un vector de características. En consecuencia, un conjunto de datos es una colección de vectores en un espacio  $\mathbb{R}^p$ .

Denotaremos  $X_1, \dots, X_n$  los datos y el índice  $i$  vale por el  $i$ -ésimo dato. Las entradas de los datos se llaman atributos.

Cada dato tiene una etiqueta  $Y_i$  que toma valores en un conjunto  $\{1, \dots, K\}$  (con  $K \in \mathbb{N}$ ) que representa las  $K$  categorías a clasificar.



## 1.1.2. Clasificación óptima

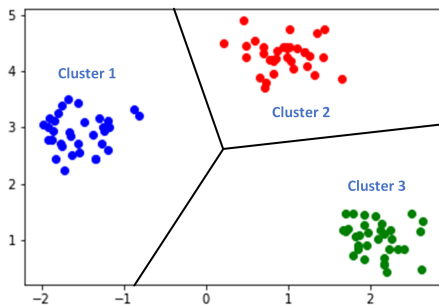
### Regla de clasificación:

Una regla de clasificación es una función  $g : \mathbb{R}^P \rightarrow \{1, \dots, K\}$  que a todo  $x \in \mathbb{R}^P$  asocia una de las  $K$  etiquetas.

Toda regla de clasificación  $g$  induce una partición (propia a  $g$ ) del espacio  $\mathbb{R}^P$  en

$$\mathbb{R}^P = R_1 \cup \dots \cup R_K$$

que representan  $K$  regiones disjuntas y  $R_i = \{x \in \mathbb{R}^P : g(x) = i\}$ .



## 1.1.2. Clasificación óptima

### Un poco más de modelación:

Cada sub-conjunto de datos  $C_i = \{X_j : Y_j = i\}$  se llama población  $i$  o cluster  $i$ . Denotaremos  $\{X \rightarrow R_i\}$  el evento que el punto  $X$  sea categorizado en  $R_i$ .

Supongamos que los datos de la población  $C_i$  tienen una cierta distribución subyacente de densidad dada por una función  $f_i$  y que tiene una probabilidad de inclusión  $\pi_i = \mathbb{P}(X \rightarrow C_i)$ .

Sea  $E$  el evento de hacer un error de clasificación.

$$\begin{aligned}\mathbb{P}(E) &= \mathbb{P}(E \cap \{X \rightarrow C_1\}) + \cdots + \mathbb{P}(E \cap \{X \rightarrow C_K\}) \\ &= \sum_{i=1}^K \mathbb{P}(X \rightarrow C_i) \mathbb{P}(E | \{X \rightarrow C_i\}) = \sum_{i=1}^K \pi_i \mathbb{P}(E | \{X \rightarrow C_i\}) \\ &= \sum_{i=1}^K \pi_i (1 - \mathbb{P}(X \in R_i | \{X \rightarrow C_i\})) = 1 - \sum_{i=1}^K \pi_i \int_{R_i} f_i(x) dx\end{aligned}$$

## 1.1.2. Clasificación óptima

Por otro lado vamos a calcular

$$q_i(x) = \mathbb{P}(X \rightarrow C_i | X = x).$$

Por la regla de Bayes,

$$\begin{aligned} q_i(x) &= \frac{\mathbb{P}(\{X \rightarrow C_i\} \cap \{X = x\})}{\mathbb{P}(X = x)} = \frac{\pi_i \mathbb{P}(X = x | X \rightarrow C_i)}{\mathbb{P}(X = x)} \\ &= \frac{\pi_i \mathbb{P}(X = x | X \rightarrow C_i)}{\sum_{j=1}^K \pi_j \mathbb{P}(X = x | X \rightarrow C_j)} = \frac{\pi_i f_i(x)}{\sum_{j=1}^K \pi_j f_j(x)} \end{aligned}$$

Y tenemos también que  $\mathbb{P}(X = x) = \sum_{j=1}^K \pi_j f_j(x)$  de tal manera que

$$\begin{aligned} \mathbb{P}(\text{"No errores en } g\text{"}) &=: \mathbb{P}(\text{NoErr}(g)) = \sum_{i=1}^K \int_{R_i} q_i(x) \mathbb{P}(X = x) dx \\ &= \int_{\mathbb{R}^p} \mathbb{P}(X = x) \left( \sum_{i=1}^K \mathbb{1}_{x \in R_i} q_i(x) \right) dx \end{aligned}$$

# 1.Métodos de clasificación

## 1.2.Curso del 17 de agosto



## 1.2.1. Clasificación óptima

Vimos que la probabilidad de no hacer errores en una regla de clasificación  $g$  está dada por

$$\mathbb{P}(\text{NoErr}(g)) = \int \mathbb{P}(X = x) h_g(x) dx$$

donde  $h_g(x) = \sum_{i=1}^K \mathbb{1}_{x \in R_i} q_i(x)$ .

Consideramos la regla de clasificación  $g^*$  dada por las regiones

$$R_j^* = \{x \in \mathbb{R}^p : \forall j, q_i(x) \geq q_j(x)\},$$

y consideramos  $g$  una otra regla de clasificación arbitraria. Ahora, para un  $x$  dado tenemos  $x \in R_j$  para un cierto  $j$  y  $x \in R_m^*$  para un cierto  $m$ . Por definición de  $g^*$ , tenemos  $q_m(x) \geq q_j(x)$

$$h_g(x) = \sum_{i=1}^K \mathbb{1}_{x \in R_i} q_i(x) = q_j(x) \leq q_m(x) = \sum_{i=1}^K \mathbb{1}_{x \in R_i^*} q_i(x) = h_{g^*}(x).$$

## 1.2.1. Clasificación óptima

Entonces,

$$\begin{aligned}\mathbb{P}(\text{NoErr}(g)) &= \int \mathbb{P}(X = x) h_g(x) dx \\ &\leq \int \mathbb{P}(X = x) h_{g^*}(x) dx = \mathbb{P}(\text{NoErr}(g^*)).\end{aligned}$$

La regla  $g^*$  es óptima para el criterio “probabilidad de errores”.

Encontrar las  $R_i^*$  es comparar los valores  $\pi_1 f_1(x), \dots, \pi_K f_K(x)$ . Si conocemos las  $\pi_i$  y  $f_i$  podemos diseñar el algoritmo:

- 1: **Input:**  $x$
- 2: Calcular  $\pi_1 f_1(x), \dots, \pi_K f_K(x)$ .
- 3: Calcular  $k = \operatorname{argmax}_i \pi_i f_i(x)$ .
- 4:  $k \rightarrow Y$
- 5: **Output:**  $Y$

## 1.2.2. Clasificación óptima de poblaciones normales

Si tenemos  $K$  poblaciones normales de proporciones  $\pi_i$  iguales,

$$f_i(x) = \frac{1}{(2\pi)^{p/2} |\Sigma_i|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right).$$

Las fronteras de las regiones  $R_i^*$  son de la forma  $f_i(x) = f_j(x)$  sujeto a  $\forall k \neq i, j \ f_k(x) \leq f_i(x)$ .

**Ejercicio:** Que es la frontera entre las dos regiones del caso  $\mu_1 = (0, 0)$ ,  $\mu_2 = (0, 1)$  y

$$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}?$$

### 1.2.3. Error de Bayes

En general, digamos que una regla de clasificación  $g^*$  es óptima si

$$g^* = \operatorname{argmin}_{g: \mathbb{R}^p \rightarrow \{1, \dots, K\}} \mathbb{P}(g(X) \neq Y)$$

La cantidad  $L(g) = \mathbb{P}(g(X) \neq Y)$  es la misma que en los cálculos anteriores.

El número real  $L(g^*)$  se llama **error de Bayes**.

El error de Bayes es una noción teórica y no se puede calcular en la práctica salvo en casos bien particulares!

→ La distribución de  $(X, Y)$  es en general desconocida.

Un clasificador real se construye en base a los datos  $(X_1, Y_1), \dots, (X_n, Y_n)$ . Denotaremos  $g_n$  un clasificador así. Su error teórica es  $L(g_n)$  su error estimada es  $L_n(g_n) = \mathbb{P}(g(X) \neq Y | \text{"datos"})$ .

## 1.2.4. Clasificador de Bayes para dos clases

Estamos en el contexto de dos clases  $Y \in \{0, 1\}$ . Denotaremos  $\eta(x) = \mathbb{P}(Y = 1|X = x)$ .

**Ejercicio:** Mostrar que

$$g^*(x) = \begin{cases} 1 & \text{si } \eta(x) > \frac{1}{2} \\ 0 & \text{si no.} \end{cases}$$

**Ejercicio:** Supongamos que  $\eta(x) = x/(x + c)$  con  $c > 0$ . Mostrar que

$$L(g^*) = \mathbb{E}[\min(\eta(X), 1 - \eta(X))] = \mathbb{E}\left[\frac{\min(c, X)}{X + c}\right].$$

Calcular  $L(g^*)$  si  $X$  es constante igual a  $c$  y luego si  $X \sim \text{Unif}[0, 4c]$ .

# 1.Métodos de clasificación

## 1.3.Curso del 22 de agosto

## 1.3.1. Algoritmo Bayes “Naive”

Hemos visto que el clasificador (regla de clasificación) óptimo es

$$g^*(x) = \underset{k}{\operatorname{argmax}} \mathbb{P}(Y = k | X = x) = \underset{k}{\operatorname{argmax}} \pi_k f_k(x)$$

y  $x = (x_1, \dots, x_p)$  es el vector de atributos. Podemos usar el chain rule para escribir

$$f_k(x) = \mathbb{P}(X_1 = x_1 | Y = k) \mathbb{P}(X_2 = x_2 | X_1 = x_1, Y = k) \dots \\ \mathbb{P}(X_p = x_p | X_1 = x_1, \dots, X_{p-1} = x_{p-1}, Y = k).$$

El clasificador “naive” Bayes supone que las variables de atributos son independientes de tal manera que

$$f_k(x) = \prod_{j=1}^p \mathbb{P}(X_j = x_j | Y = k) =: f_{k,1}(x_1) \dots f_{k,p}(x_p)$$

## 1.3.1. Algoritmo Bayes “Naive”

### **Ventajas:**

- ▶ Las probabilidades  $f_{k,1}(x_1)$  se pueden estimar de forma individual (estimación de marginales).
- ▶ Se puede considerar fácilmente mezclas de variables discretas y continuas.
- ▶ En un buen “primer” clasificador que funciona en alta dimensión (e.g. clasificación de textos.)

### **Contra:**

- ▶ Es “naive”! El supuesto puede ser invalido!



## 1.3.1. Algoritmo Bayes “Naive”

Resumen clasificador Naive Bayes (NB):

- ▶ En principio, la tarea de clasificación óptima requiere el conocimiento de las  $\pi_i$  (proporciones en cada población) y de las  $f_i$  (densidades asociadas a cada población.)
- ▶ La información de estas cantidades alrededor de la frontera tiene más importancia.
- ▶ Con información previa sobre las  $f_i$  o  $\pi_i$  podemos simplificar las tareas de clasificación (gaussianas, multinomiales, etc...)
- ▶ Si suponemos atributos independientes, el clasificador se llama “Naive Bayes” .

## 1.3.2. Discriminante lineal (dos poblaciones)

Hemos visto en las clases anteriores que la separación entre dos poblaciones gaussianas con la misma matriz de covarianza  $\Sigma$  resulta en una separación por un hiperplano. Para dos grupos y una separación vertical:

$$g(x) = \begin{cases} y' & \text{si } x \leq x' \\ 1 - y' & \text{si no} \end{cases}$$

Los parámetros del clasificador son los  $x'$  y  $y'$ . Empezamos a estudiar este caso sencillo antes de generalizar.

Vamos a denotar

$$F_1(x) = \mathbb{P}(X \leq x | Y = 1)$$

$$F_0(x) = \mathbb{P}(X \leq x | Y = 0)$$

$$p = \mathbb{P}(Y = 1)$$

las funciones de distribución condicionales a las clases 1 o 0.

## 1.3.2. Discriminante lineal (dos poblaciones)

El riesgo del clasificador óptimo (a dentro de esa clase de discriminantes lineales) es

$$L = \inf_{x', y'} [\mathbb{1}_{y'=0}(pF_1(x') + (1-p)(1-F_0(x')))) \\ + \mathbb{1}_{y'=1}(p(1-F_1(x')) + (1-p)F_0(x'))]$$

Un corte óptimo  $(x^*, y^*)$  se llama corte teórico de Stoller.

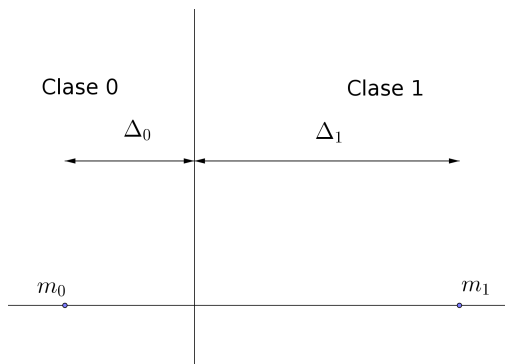
**Ejercicio:** Mostrar que  $L^* \leq L \leq \min(p, 1-p)$ . Usarlo para mostrar que  $L \leq 1/2$  con igualdad si y solo si  $L^* = 1/2$ .

### 1.3.3. Cortes de Stoller

Si denotamos  $m_0 = \mathbb{E}[X|Y = 0]$ ,  $m_1 = \mathbb{E}[X|Y = 1]$ ,  
 $\sigma_0^2 = \text{Var}(X|Y = 0)$  y  $\sigma_1^2 = \text{Var}(X|Y = 1)$ :

Teorema

$$L^* \leq L \leq \frac{1}{1 + \frac{(m_0 - m_1)^2}{(\sigma_0 + \sigma_1)^2}}$$



### 1.3.3. Cortes de Stoller

La probabilidad de error está dada por

$$\begin{aligned} & pF_1(m_0 + \Delta_0) + (1 - p)(1 - F_0(m_0 + \Delta_0)) \\ &= p\mathbb{P}(X \leq m_1 - \Delta_1 | Y = 1) + (1 - p)\mathbb{P}(X > m_0 + \Delta_0 | Y = 0) \\ &\leq p \frac{\sigma_1^2}{\sigma_1^2 + \Delta_1^2} + (1 - p) \frac{\sigma_0^2}{\sigma_0^2 + \Delta_0^2} \end{aligned}$$

Tomando  $\Delta_1 = \frac{\sigma_1}{\sigma_0} \Delta_0$  y  $\Delta_0 = |m_1 - m_0| \sigma_0 / (\sigma_0 + \sigma_1)$  obtenemos

$$L \leq \frac{1}{1 + \frac{(m_0 - m_1)^2}{(\sigma_0 + \sigma_1)^2}}.$$

Usamos que  $\mathbb{P}(X - \mathbb{E}[X] \geq t) \leq \frac{\text{Var}(X)}{\text{Var}(X) + t^2}$ .

# 1.Métodos de clasificación

## 1.4.Curso del 24 de agosto

## 1.4.1. Cortes de Stoller empíricos

Una manera de construir cortes en la práctica es usar

$$(x', y') = \operatorname{argmin}_{x, y} \frac{1}{n} \sum_{i=1}^n (\mathbb{1}_{X_i \leq x, Y_i \neq y} + \mathbb{1}_{X_i > x, Y_i \neq 1-y}).$$

Estos cortes se llaman cortes de Stoller empíricos. Denotamos  $L_n$  su riesgo. Se puede escribir de la forma

$$(x', y') = \operatorname{argmin}_{x, y} \nu_n(C(x, y))$$

donde  $\nu_n$  es la medida empírica (i.e.

$$\nu_n(A) = (1/n) \sum_{i=1}^n \mathbb{1}_{(X_i, Y_i) \in A} \text{ y}$$

$$C(x, y) = [(-\infty, x] \times \{y\}] \cup [[x, +\infty) \times \{1-y\}].$$

Se ve directamente que  $\mathbb{E}[\nu_n(C(x, y))] = \nu(C(x, y)) = L(g)$  donde  $g$  es el clasificador lineal correspondiendo a la pareja  $(x, y)$ .

## 1.4.1. Cortes de Stoller empíricos

Entonces,

$$\begin{aligned}L_n &= \nu(C(x', y')) \\ &= \nu(C(x', y')) - \nu_n(C(x', y')) + \nu_n(C(x', y')) \\ &\leq \sup_{(x,y)} (\nu(C(x, y)) - \nu_n(C(x, y))) + \nu_n(C(x^*, y^*)) \\ &\leq 2 \sup_{(x,y)} (\nu(C(x, y)) - \nu_n(C(x, y))) + \nu(C(x^*, y^*)) \\ &= 2 \sup_{(x,y)} (\nu(C(x, y)) - \nu_n(C(x, y))) + L\end{aligned}$$

Eso permite tener el

### Teorema

Sea  $\epsilon > 0$ ,

$$\mathbb{P}(L_n - L \geq \epsilon) \leq 4e^{-n\epsilon^2/2}$$

y

$$\mathbb{E}[L_n - L] \leq \sqrt{\frac{2 \log(4e)}{n}}.$$



## 1.4.2. En dimensión más alta

En el caso  $d > 1$ , la discriminación lineal se escribe con

$$g(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^d a_i x_i + a_0 > 0 \\ 0 & \text{si no} \end{cases}$$

Con las ideas de resultados tipo Cramér y Wold, podemos tener el  
Teorema

$$L^* \leq L \leq \inf_{a \in \mathbb{R}^p} \frac{1}{1 + \frac{a^T (m_0 - m_1)^2}{((a^T \Sigma_0 a)^{1/2} + (a^T \Sigma_1 a)^{1/2})^2}}$$

La tarea principal es encontrar el buen parámetro  $a = (a_1, \dots, a_d)$  que se acercara de inf!

### 1.4.3. Criterio de Fisher

Es un modo de elección el vector de discriminación  $a$ . Desde la muestra, se puede calcular las medias empíricas de las clases  $\hat{m}_0$  y  $\hat{m}_1$ .

Imagina proyectar los datos  $X_i$  sobre una línea en la dirección  $a$ . Acedemos entonces a  $a^T X_1, \dots, a^T X_n$ . Podemos calcular las varianzas empíricas en esta dirección:

$$\hat{\sigma}_0 = \sum_{i: Y_i=0} (a^T X_i - a^T \hat{m}_0)^2 = a^T S_0 a$$

donde

$$S_0 = \sum_{i: Y_i=0} (X_i - \hat{m}_0)(X_i - \hat{m}_0)^T.$$

De manera similar obtenemos  $\hat{\sigma}_1$  y  $S_1$ .

### 1.4.3. Criterio de Fisher

Se define el discriminante lineal de Fisher como

$$J(a) = \frac{(a^T(\hat{m}_0 - \hat{m}_1))^2}{\hat{\sigma}_0^2 + \hat{\sigma}_1^2} = \frac{(a^T(\hat{m}_0 - \hat{m}_1))^2}{a^T(S_0 + S_1)a}$$

y se selecciona la dirección  $a$  que maximiza este criterio de Fisher. De hecho los cálculos se pueden hacer de manera explícita y se obtiene

$$a = (S_0 + S_1)^{-1}(\hat{m}_0 - \hat{m}_1).$$

Luego para la elección de  $a_0$ , se puede hacer una discriminación lineal unidimensional con los datos  $a^T X_1, \dots, a^T X_n$ .

**Ejercicio:** Mostrar que  $a$  es efectivamente el maximizador de  $J(a)$ . Mostrar de la misma manera que el maximizador de

$$J_2(a) = \frac{(a^T(\hat{m}_0 - \hat{m}_1))^2}{p\hat{\sigma}_0^2 + (1-p)\hat{\sigma}_1^2}$$

está dado por  $a = (pS_0 + (1-p)S_1)^{-1}(\hat{m}_0 - \hat{m}_1)$ .

# 1.Métodos de clasificación

1.5.Curso del 29 de agosto

## 1.5.1. ERM (Minimización del riesgo empírico)

Retomamos la regla de clasificación lineal

$$g(x) = \begin{cases} 1 & \text{si } a^T x + a_0 > 0 \\ 0 & \text{si no} \end{cases}$$

tal que la probabilidad de error se escribe  $\mathbb{P}(g(X) \neq Y)$ . El riesgo empírico está dado por la cantidad

$$L_n(g) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{g(X_i) \neq Y_i}.$$

Para este clasificador suponemos que  $X$  tenga una densidad. Tomamos arbitrariamente  $d$  datos  $X_{i_1}, \dots, X_{i_d}$  en el conjunto de datos total. Por el supuesto de densidad, los datos definen un único hiperplano con probabilidad 1.

## 1.5.1. ERM (Minimización del riesgo empírico)

Corresponden dos clasificadores

$$g_1(x) = \begin{cases} 1 & \text{si } a^T x + a_0 > 0 \\ 0 & \text{si no} \end{cases}$$

y

$$g_2(x) = \begin{cases} 1 & \text{si } a^T x + a_0 < 0 \\ 0 & \text{si no} \end{cases}$$

Entonces, para cada uno de estos clasificadores se puede calcular  $L_n(g_1)$  y  $L_n(g_2)$ .

A cada elección de  $d$  puntos, se puede asociar una par de clasificadores potenciales. Al total son  $2\binom{n}{d}$ . Al final se selecciona el clasificador

$$\hat{g} = \underset{g}{\operatorname{argmin}} L_n(g)$$

donde la minimización se hace sobre esta clase de clasificadores lineales particulares.

## 1.5.1. ERM (Minimización del riesgo empírico)

Este clasificador tiene una buena garantía teórica porque satisface el

### Teorema

*Supongamos que  $n \geq d$  (donde  $d$  es la dimensión del vector  $X$  de atributos), entonces, para cada distribución de  $(X, Y)$  y cada  $2d/n \leq \epsilon \leq 1$ , tenemos,*

$$\mathbb{P}(L(\hat{g}) > L + \epsilon) \leq e^{2d\epsilon} \left( 2 \binom{n}{d} + 1 \right) e^{-n\epsilon^2/2}$$

*y en consecuencia,*

$$\mathbb{E}[L(\hat{g}) - L] \leq \sqrt{\frac{2}{n}((d+1) \log n + (2d+2))}.$$

## 1.5.2. Criterio mínimos cuadrados

Este criterio está basado sobre la regresión lineal de la variable  $Y$  sobre  $X$ . Una medición natural del error de regresión está dado por los mínimos cuadrados

$$MC_n(g) = \frac{1}{n} \sum_{i=1}^n (Y_i - a^T X_i - a_0)^2.$$

Cuando  $n$  es grande esta cantidad se acerca de

$$MC(g) = \mathbb{E} \left[ (Y - a^T X - a_0)^2 \right]$$

que es, en general, alejado de  $L(g)$ .

**Pro:**

- ▶ Como en regresión lineal,  $a$  y  $a_0$  son explícitos.

**Contra:**

- ▶ Se pueden encontrar distribuciones tal que  $L(\hat{g}) - L \sim 1$ .



## 1.5.2. Criterio mínimos cuadrados

**Justificación:** Consideramos el caso de una distribución triatomica:

$$\mathbb{P}((X, Y) = (-\theta, 1)) = \mathbb{P}((X, Y) = (1, 1)) = \epsilon/2$$

y

$$\mathbb{P}((X, Y) = (0, 0)) = 1 - \epsilon$$

para  $1 > \epsilon > 0$  y  $\theta > 0$ . La mejor estrategia de corte lineal es asignar la clase 0 al interval  $[-\theta/2, \infty)$ . Eso permite calcular que  $L = \epsilon/2$ . Pero resolviendo el cálculo del minimizador de los mínimos cuadrados, obtenemos que

$$\begin{cases} 2a_0 = 2\epsilon - a_1\epsilon(1 + \theta) \\ (1 - \theta) - a_1(1 + \theta^2) = a_0(1 - \theta) \end{cases}$$

Eso da (en el regimen  $\epsilon \sim 0$  y  $\theta \sim \infty$ ),  $a_1 \sim -1/\theta$  y  $a_0 \sim 3\epsilon/2$ . Entonces,  $L(g) \geq \mathbb{P}(Y = 0 \text{ y } -\theta^{-1}X + 3\epsilon/2 > 0) = 1 - \epsilon$ .

### 1.5.3. El criterio Perceptron

Existen otros criterios que podemos optimizar en la búsqueda del buen corte. El siguiente criterio se llama **criterio perceptron**:

$$J(a, a_0) = \sum_{2Y_i - 1 \neq \text{sg}(a^T X_i + a_0)} |a^T X_i + a_0|.$$

Se puede mostrar que este criterio sufre de los mismos defectos que el criterio de los mínimos cuadrados (con ideas similares de distribuciones caóticas).

## 1.5.4. Criterio de regresión Sigmoid

Se puede también usar un criterio menos lineal y minimizar la cantidad

$$SC_n(g) = \frac{1}{n} \sum_{i=1}^n (Y_i - \sigma(a^T X_i - a_0))^2$$

donde  $\sigma(u)$  es una función sigmoide. Formalmente, es una función creciente de 0 hasta 1. Por ejemplo:  $\sigma(u) = 1/(1 + \exp(-u))$ .

De manera análoga a los casos anteriores, se puede mostrar que existen distribuciones tal que  $L(\hat{g}) - L \sim 1$  pero si tomamos  $\sigma_h(u) = \sigma(hu)$  y que dejamos  $h \rightarrow \infty$  cuando  $n \rightarrow \infty$  entonces podemos mostrar que para cualquier distribución,

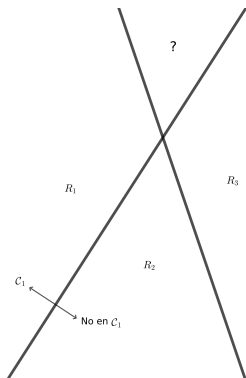
$$\mathbb{E} [L_n(\hat{g})] \longrightarrow L.$$

# 1. Métodos de clasificación

## 1.6. Curso del 31 de agosto

## 1.6.1. Clasificación de $K$ poblaciones: un problema

- ▶ En las clases anteriores vimos algunas maneras naturales para discriminar entre 2 clases.
- ▶ No es tan obvio de generalizar estas nociones de clasificación lineal a más clases.



## 1.6.1. Clasificación de $K$ poblaciones: un problema

Dos ideas (no buenas...) para reducir el problema:

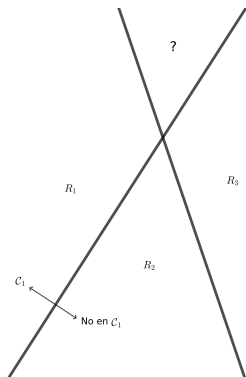
- ▶ Se consideran  $K$  clasificadores tipo “**1 vs los demas**”.
- ▶ Se consideran  $K(K - 1)/2$  clasificadores “**1 vs 1**”

Como agregar los clasificadores para tener una partición del espacio en zonas  $R_1, \dots, R_K$  de tal manera que las separaciones sean lineales?

## 1.6.1. Clasificación de $K$ poblaciones: un problema

En el caso  $K$  clasificadores tipo “**1 vs los demas**”:

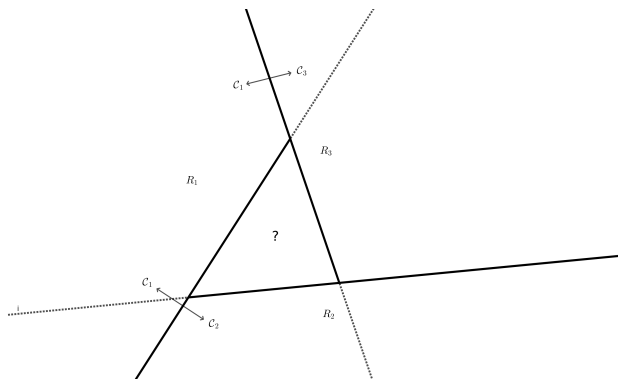
Ya tenemos un problema tratando de agregar dos clasificadores!



## 1.6.1. Clasificación de $K$ poblaciones: un problema

En el caso  $K$  clasificadores tipo “**1 vs 1**”:

Un otro tipo de problema surge en este caso!





## 1.6.2. Clasificación de $K$ poblaciones: Parentesis

Hemos visto que la elección de la dirección óptima es  $a = (S_0 + S_1)^{-1}(m_0 - m_1)$ . Como generalizar esta técnica?

En general, si tenemos dos matrices cuadradas  $A$  y  $B$  de misma dimensión y simétricas definidas positivas,

$$\max_{a \in \mathbb{R}^d} \frac{a^T A a}{a^T B a} = \lambda_1$$

donde  $\lambda_1$  es el valor propio máximo de la matriz  $B^{-1}A$ . Además la maximización se hace en el vector propio que correspondiendo a este valor propio  $\lambda_1$ .

**Idea:** Si queremos seleccionar más que una dirección para la proyección de nuestros datos  $X_1, \dots, X_n$  podemos usar el espacio dado por las  $r$  primeros valores propios  $\lambda_1, \dots, \lambda_r$ .

## 1.6.3. Clasificación de $K$ poblaciones: Reducción de Fisher (LDA)

Que van a ser  $A$  y  $B$ ?

Para simplificar vamos a denotar  $x_{i,j}$  los datos tal que  $i$  designa la población y  $j$  designa el índice en  $\{1, \dots, n_i\}$  del dato adentro de la población  $i$ .

Medida de dispersión a dentro de las clases:

$$S_D = \sum_{k=1}^K S_i$$
$$S_i = \sum_{j=1}^{n_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$
$$\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{i,j}$$

### 1.6.3. Clasificación de K poblaciones: Reducción de Fisher (LDA)

La medida de dispersión total es

$$S_T = \sum_{i=1}^K \sum_{j=1}^{n_i} (x_{i,j} - \bar{x})(x_{i,j} - \bar{x})^T$$

donde  $\bar{x}$  es la media sobre todos los datos. Entonces,

$$\begin{aligned} S_T &= \sum_{i=1}^K \sum_{j=1}^{n_i} (x_{i,j} - \bar{x}_i + \bar{x}_i - \bar{x})(x_{i,j} - \bar{x}_i + \bar{x}_i - \bar{x})^T \\ &= \sum_{i=1}^K S_i + \sum_{i=1}^K n_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T \\ &=: S_D + S_E \end{aligned}$$

y  $S_E$  es una “medida” de la dispersión entre clases.

### 1.6.3. Clasificación de $K$ poblaciones: Reducción de Fisher (LDA)

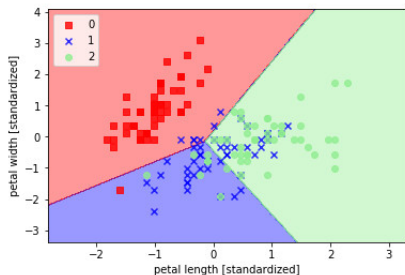
La buena noción es de tomar  $A = S_E$  y  $B = S_D$ .

→ Se reduce a un problema de algebra lineal de búsqueda de valores propios y vectores propios.

En general se toman los  $r$  valores propios más grandes y se elige  $r$  de tal manera que los valores propios ignorados son de magnitud despreciable.

## 1.6.4. Solución para la clasificación lineal

Queremos alcanzar hacer una clasificación del tipo:



Podemos definir un solo clasificador lineal (siguiendo las ideas del Naive Bayes).

Consideramos  $K$  funciones lineales  $y_1, \dots, y_K$ ,

$$y_k(x) = a_k^T x + a_{k,0}$$

y se asigna la clase  $k$  si y solo si

$$\forall j \neq k, \quad y_k(x) > y_j(x).$$

## 1.6.4. Solución para la clasificación lineal

Las fronteras son las partes del espacio que corresponden a un subconjunto de

$$\{x \in \mathbb{R}^d : y_k(x) = y_j(x)\} = \{x \in \mathbb{R}^d : (a_k - a_j)^T x + (a_{k,0} - a_{j,0}) = 0\}.$$

Son pedazos de hiperplanos.

Las regiones  $R_i$  son completamente conectadas y convexas.

**Ejercicio:** Mostrar que las  $R_i$  son convexas. (Usar la linealidad de las  $y_j$ )

# 1.Métodos de clasificación

## 1.7.Curso del 5 de septiembre

## 1.7.1. $k$ -vecinos más cercanos: definición

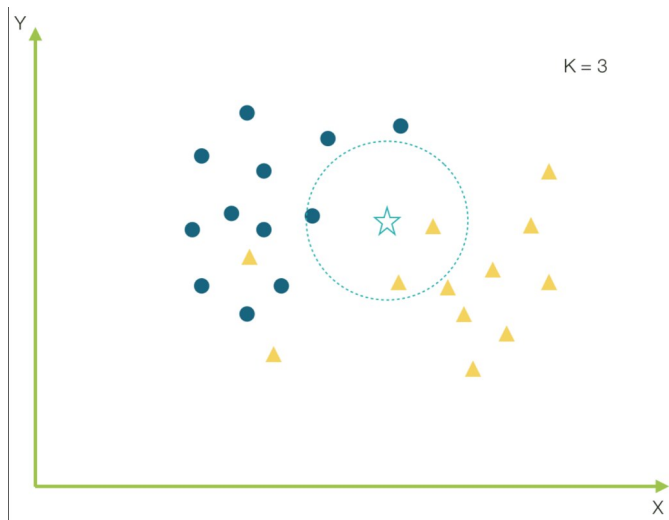
El clasificador de  $k$ -vecinos más cercanos está basado sobre la filosofía: Los puntos cercanos se parecen.

**Definición:** Dado un punto del espacio  $x$ , encontramos los  $k$  puntos de entrenamiento  $X_{(1)}, \dots, X_{(k)}$  los más cercanos de  $x$ . Luego, la clasificación de  $Y(x)$  está dada por el valor de  $Y$  lo más representado en los  $Y_i$  correspondientes.

- ▶ Los empates se deciden al azar entre los mayoritarios.
- ▶ Típicamente, se usa la distancia euclidiana.
- ▶ Si  $k$  es pequeño, el clasificador es muy local. Si  $k$  es grande, el clasificador es más robusto errores en los datos de entrenamiento.



## 1.7.1. $k$ -vecinos más cercanos: definición



## 1.7.2. $k$ -vecinos más cercanos: Problemáticas

Formalmente, podemos escribir el clasificador  $g_n$  en un formalismo kernel.

$$g_n(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{n,i} \mathbb{1}_{Y_i=1} > \sum_{i=1}^n w_{n,i} \mathbb{1}_{Y_i=0} \\ 0 & \text{si no,} \end{cases}$$

donde  $w_{n,i} = 1/k$  si  $X_i$  es uno de los vecinos más cercanos de  $x$ .  
Las problemáticas de  $k$  vecinos más cercanos:

- ▶ Consistencia universal: Que pasa si  $k \rightarrow \infty$ ,  $k/n \rightarrow 0$  y  $n \rightarrow \infty$ ?
- ▶ Si  $k$  es finito, que podemos decir si  $n \rightarrow \infty$ ?
- ▶ Que son los buenos pesos a poner en la definición?
- ▶ Que son los retos de programación de los  $k$ -NN?

## 1.7.2. $k$ -vecinos más cercanos: Problemáticas

Denotamos  $X_{(i)}(x)$  el  $i$ -ésimo vecino más cercano a  $x$ . Hay un resultado fácil a obtener:

### Lema

Si  $X \sim \mu$  y  $k/n \rightarrow 0$  entonces para  $x \in \text{soporte}(\mu)$ ,  
 $\|X_{(k)}(x) - x\| \rightarrow 0$  en probabilidad.

La idea es sencilla:

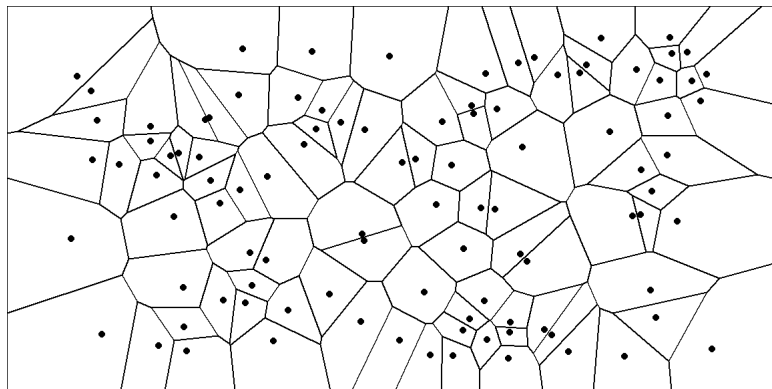
Si  $S_{x,\epsilon}$  es la bola centrada en  $x$  y de radio  $\epsilon$ ,

$$\|X_{(k)}(x) - x\| > \epsilon \quad \Leftrightarrow \quad \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{X_i \in S_{x,\epsilon}} < \frac{k}{n}.$$

Por la ley de grandes números  $\frac{1}{n} \sum_{i=1}^n \mathbb{1}_{X_i \in S_{x,\epsilon}} \rightarrow \mu(S_{x,\epsilon})$  y la cantidad de la derecha tiende a 0.

### 1.7.3. 1-NN: el vecino más cercano

En este caso se decide de  $Y(x)$  viendo únicamente el vecino más cercano de  $x$  en los datos. Las regiones son de la forma:



Esas formas son conocidas como las celdas de Voronoi.

### 1.7.3. 1-NN: el vecino más cercano

En sus papers teóricos, Stone, Devroye y Györfi muestran unos teoremas de convergencia y cotas superiores. Unas ideas (para dos clases) son:

- ▶ Para  $x \in \mathbb{R}^d$ ,  $Y_{(1)}(x), \dots, Y_{(k)}(x)$  son las  $Y$  de los  $k$ -vecinos más cercanos.
- ▶ Se define  $Y'_i(x) = \mathbb{1}_{U_i \leq \eta(x)}$  donde  $\eta(x) = \mathbb{P}(Y = 1 | X = x)$  donde  $U_i$  es una variable uniforme independiente de las otras variables.
- ▶ Se denota  $g'_n$  el clasificador  $k$ -NN donde reemplazamos las  $Y_{(i)}(x)$  por las  $Y'_{(i)}(x)$ .

Se muestra que  $\mathbb{E}[L_n - L'_n] \rightarrow 0$  cuando  $k/n \rightarrow 0$ , donde  $L_n = \mathbb{E}[g_n(X) \neq Y | D_n]$  y  $L'_n = \mathbb{E}[g'_n(X) \neq Y | D'_n]$ .

### 1.7.3. 1-NN: el vecino más cercano

Entonces si denotamos  $L_{NN} = \lim_{n \rightarrow \infty} \mathbb{E}[L_n]$  (para  $k = 1$ ) es suficiente calcular  $\mathbb{E}[L'_n]$ .

- ▶ Es más fácil!
- ▶ Las variables  $Y'_i(x)$  son i.i.d. Bernoulli de parámetro  $\eta(x)$ .
- ▶ Para  $k = 1$ ,  
$$\mathbb{E}[L'_n] = \mathbb{P}\left(Y'_{(1)}(X) \neq Y\right) = \mathbb{E}[2\eta(X)(1 - \eta(X))].$$
- ▶ Eso demuestra directamente que

$$L^* \leq L_{NN} \leq 2L^*$$

## 1.7.4. $k$ general y impar

De manera general tenemos el

$$L_{kNN} = \mathbb{E} \left[ \sum_{j=0}^k \binom{k}{j} \eta(X)^j (1 - \eta(X))^{k-j} (\eta(X) \mathbb{1}_{j < k/2} + (1 - \eta(X)) \mathbb{1}_{j > k/2}) \right]$$

Ademas, tenemos que

$$L^* \leq \dots \leq L_{(2k+1)NN} \leq L_{(2k-1)NN} \leq \dots \leq L_{3NN} \leq L_{NN} \leq L^*$$

En particular, cuando la tarea de clasificación es fácil ( $L \approx 0$ ), el algoritmo de  $k$ -NN da buenos resultados.

## 1.7.5. Complejidad del algoritmo

Hay opciones de algoritmos que estructuran (por agrupamientos) los datos para facilitar un uso repetido del clasificador.

- ▶ **Greedy:**

- ▶ Training: Tiempo  $O(1)$
- ▶ Training: Memoria  $O(1)$  (no hay entrenamiento)
- ▶ Testing: Tiempo  $O(d * k * n)$ .
- ▶ Testing: Memoria  $O(1)$ .

- ▶ **Método arboles k-d, Método Ball-Tree**

- ▶ Training: Tiempo  $O(d * n * \log n)$
- ▶ Training: Memoria  $O(d * n)$
- ▶ Testing: Tiempo  $O(k \log n)$ .
- ▶ Testing: Memoria  $O(1)$ .



# 1.Métodos de clasificación

## 1.8.Curso del 7 de septiembre

## 1.8.1. Árboles de decisión

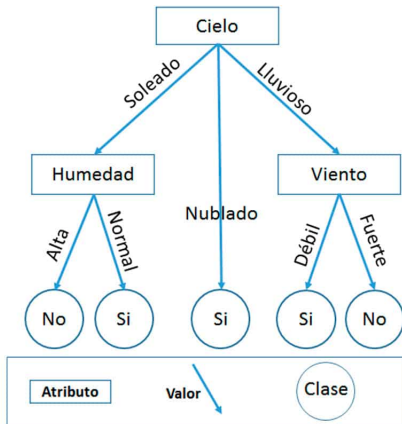
Los métodos de reducción para los  $k$ -vecinos más cercanos (k-d-trees o Ball-trees) son casos particulares de un método más general: **Árboles de decisión**.

- ▶ Un algoritmo tipo árbol de decisión construye un árbol donde las hojas del árbol corresponde a una categoría.
- ▶ Cada nodo del árbol corresponde a una decisión de clasificación.
- ▶ Clasificar un nuevo dato consiste a bajar en el árbol y contestar una serie de preguntas.

Es un algoritmo que sirve de clasificador pero también puede también cumplir con la tarea de regresión.

## 1.8.1. Árboles de decisión

Ejemplo de árbol de decisión construido.



## 1.8.2. Árboles de decisión: Definición formal

Un árbol de decisión se construye recursivamente:

1. Se elige al azar un nodo activo del árbol.
  2. Se elige al azar un componente  $i$  dentro de las  $d$  posibles de los vectores de atributos ( $X_j$ ) que pertenecen al nodo.
  3. En base a los  $X_{j,i}$  y  $Y_j$  del nodo, se separan los datos en dos grupos  $G_1$  y  $G_2$  en base a una regla de clasificación (sencilla).
  4. Se crea dos nodo “hijos” que reciben respectivamente los datos de  $G_1$  y de  $G_2$ .
    - ▶ Si un criterio de paro está verificado el nodo hijo se transforma en una hoja del árbol y
    - ▶ Si no, el nodo se considera activo.
- ▶ **Inicio:** El algoritmo inicia con la raíz que contiene todos los datos.
- ▶ **Fin:** El algoritmo termina cuando no hay más nodo activos.

## 1.8.2. Árboles de decisión: Definición formal

- ▶ En la etapa 3, en general se usa una clasificación muy sencilla: lineal unidimensional.
- ▶ En general se trata de tener dos grupos lo más separados posibles.

### **Gini Impurity:**

Es un índice que mide que tanto se parecen los datos a dentro de un cierto grupo  $G$ . Denotamos  $p_1, \dots, p_K$  las proporciones de cada población a dentro de  $G$ .

- ▶ Para una población  $i$ , la impuridad está dada por
$$p_i(\sum_{j \neq i} p_j) = p_i(1 - p_i).$$
- ▶ La impuridad global está dada por
$$I_G = \sum_{i=1}^K p_i(1 - p_i) = 1 - \sum_{i=1}^K p_i^2$$

## 1.8.2. Árboles de decisión: Definición formal

### Entropía de información:

De la misma manera se consideran  $p_1, \dots, p_K$  y se calcula

$$H_G = - \sum_{i=1}^K p_i \log(p_i).$$

### Varianza:

Se considera sencillamente la varianza dentro del grupo  $G$  con

$$V_G = \sum_{i \in G} (y_i - \bar{y}_G)^2.$$

Para una decisión de clasificación y una medida de disparidad  $I$ , se maximiza el criterio siguiente:

$$C = I_{\text{nodo padre}} - I_{\text{nodo hijo}_1} - I_{\text{nodo hijo}_2}$$

### 1.8.3. Criterios de paro

Los criterios de paro son mayormente de dos categorías:

- ▶ Si un grupo tiene menos datos que un cierto número  $n_0$ .
- ▶ Si el criterio  $C$  máximo (de la etapa de corte) está abajo de un cierto nivel  $C_0$ .

A veces, se consideran criterios de paro híbridos.

#### **Ventajas:**

- ▶ Los arboles de decisión son fácilmente interpretables.
- ▶ Pueden funcionar para datos que mezclan atributos continuos y discretos.

#### **Contra:**

- ▶ Tiene la tendencia de sobre ajustar las regiones a los datos.
- ▶ Las fronteras de separaciones entre las clases son principalmente paralelas a los ejes.

## 1.8.4. Bosques aleatorios

Los bosques aleatorios son conjuntos de arboles  $\mathcal{T}_1, \dots, \mathcal{T}_m$  de decisión, nada más.

- ▶ Para clasificar, se hace una votación entre las respuestas de clasificación de cada árbol de decisión de la clase.
- ▶ Los diferentes arboles de decisión se construyen de la manera la más independiente posible.
- ▶ Por ejemplo, se puede seleccionar al azar un sub-conjunto de los datos y crear un árbol de decisión sobre estos datos.



## 2. Búsqueda de estructura: Cambio de espacio

### 2.1. Clase del 21 de septiembre

## 2.1.1. Contexto

La búsqueda de estructura en un conjunto de datos es generalmente un trabajo **anterior** a cualquier tipo de estimación, predicción, etc... sobre los datos.

**Idea principal:** Supongamos tener a disposición un conjunto de datos  $X_1, \dots, X_n$  en un espacio  $\mathcal{X}$ . En general tendremos que  $\mathcal{X}$  sea un espacio lineal. Aquí podemos considerar que cada  $X_i$  contiene toda la información del dato  $i$  (variables atributos o explicativas y variables de categorías o de respuesta).

Queremos hacer una modificación de los datos  $T : \mathcal{X} \rightarrow \mathcal{X}'$  de tal manera que los variables  $T(X_1), \dots, T(X_n)$  contengan la mayoría de la información. La transformación  $T$  es *buen*a si es más **facil** de hacer el estudio de datos sobre los  $T(X_1), \dots, T(X_n)$ .

## 2.1.1. Contexto

Hay dos formas de considerar que el estudio es más fácil.

1. El espacio  $\mathcal{X}'$  es un espacio más *chico* de  $\mathcal{X}$ .
  - ▶ En general, la dimensión del espacio  $\mathcal{X}'$  es menor (idealmente mucho menor) que la dimensión del espacio  $\mathcal{X}$ .
  - ▶ La complejidad de los algoritmos es más baja en  $\mathcal{X}'$ . Por ejemplo, el espacio nuevo tiene una estructura topológica adicional (convexidad, compacidad, datos sparse, etc...)
2. La tarea de estudio de datos es *más potente* en  $\mathcal{X}'$ .
  - ▶ Los datos son más separados en el nuevo espacio.
  - ▶ Las técnicas a usar luego son más sencillas (algoritmos lineales, etc...)

## 2.1.2. Descomposición en valores singulares (SVD)

Nos ponemos en el contexto de un espacio vectorial  $\mathcal{X} = \mathbb{R}^d$ .

La primera idea es de considerar una proyección lineal

$T : \mathbb{R}^d \rightarrow \mathbb{R}^s$  con  $s$  más chico de que  $d$ . Una idea muy popular es la **descomposición en valores singulares (SVD)**.

Consideramos una matriz  $A$ . Esta matriz será de tamaño  $n \times d$  (denotamos  $A \in \mathcal{M}_{n,d}(\mathbb{R})$ ) pero supongamos que  $A$  no es de rango completo si no que es de rango  $r$  (denotado  $\text{rg}(A) = r$ ) con  $r < \min(n, d)$ .

### Teorema (Descomposición (SVD))

*Existen matrices ortogonales  $U \in \mathcal{O}_n(\mathbb{R})$  y  $V \in \mathcal{O}_d(\mathbb{R})$  y una matriz  $\Lambda \in \mathcal{M}_r(\mathbb{R})$  diagonal de valores diagonales  $\lambda_1, \dots, \lambda_r$  positivos tal que*

$$A = UDV^T, \quad D = \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix}$$

y  $D \in \mathcal{M}_{n,d}(\mathbb{R})$ .

## 2.1.2. Descomposición en valores singulares (SVD)

La construcción se hace considerando la matriz cuadrada  $A^T A$ . Es una matriz de tamaño  $d \times d$  simétrica y positiva. Entonces, existe una matriz ortogonal  $V \in \mathcal{M}_d(\mathbb{R})$  tal que

$$A^T A = V \Gamma V^T \quad \Gamma = \begin{pmatrix} \Lambda^2 & 0 \\ 0 & 0 \end{pmatrix}$$

donde  $V = (V_1 \ V_2)$  con  $V_1 \in \mathcal{M}_{d,r}(\mathbb{R})$ . Completamos la definición poniendo  $U_1 = AV_1\Lambda^{-1}$  y  $U = (U_1 \ U_2)$  con una completación de base por Gram-Schmidt.

Esta descomposición es importante para poder reducir la complejidad de muchos problemas a problemas de dimensión  $r$ .

**Ejercicio:** Mostrar que  $A = UDV^T$  y que  $A = U_1\Lambda V_1^T$ .

## 2.1.2. Descomposición en valores singulares (SVD)

**Ejercicio:** Mostrar que existen matrices  $G$  y  $H$  de rango  $r$  (de las cuales daremos las dimensiones) tal que

$$A = GH^T$$

Mostrar también que  $A$  es una suma de  $r$  matrices de rango 1 dada por

$$A = \sum_{i=1}^r \lambda_i u_i v_i^T.$$

Por fin, mostrar que las columnas de  $U$  son vectores propios de  $AA^T$  y que las columnas de  $V$  son vectores propios de  $A^T A$ .

## 2. Búsqueda de estructura: Cambio de espacio

2.2. Curso 26 septiembre

## 2.2.1. Descomposición en valores singulares (SVD)

**Solución Ejercicio:** Se tiene que verificar los cálculos matriciales por bloques. Por definición, tenemos

$$A^T A = V \Gamma V^T \quad \text{entonces,} \quad V^T A^T A V = \Gamma.$$

Por bloques eso da,

$$\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} A^T A \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \begin{pmatrix} \Lambda^2 & 0 \\ 0 & 0 \end{pmatrix},$$

en particular, tenemos  $V_1^T A^T A V_1 = \Lambda^2$  y también  $V_2^T A^T A V_2 = 0$ . Eso implica directamente  $A V_2 = 0$ . Ahora para terminar, podemos calcular la matriz  $U^T A V$ . De nuevo, por bloques tenemos

$$U^T A V = \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} A \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \begin{pmatrix} U_1^T A V_1 & U_1^T A V_2 \\ U_2^T A V_1 & U_2^T A V_2 \end{pmatrix}$$



## 2.2.1. Descomposición en valores singulares (SVD)

Entonces tenemos (como  $AV_1 = U_1\Lambda$ )

$$U^T AV = \begin{pmatrix} U_1^T AV_1 & U_1^T AV_2 \\ U_2^T AV_1 & U_2^T AV_2 \end{pmatrix} = \begin{pmatrix} \Lambda & 0 \\ U_2^T U_1 \Lambda & 0 \end{pmatrix} = \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix}.$$

Como tenemos  $U^T AV = D$ , obtenemos  $A = UDV^T$ . En la definición, tenemos directamente  $A = U_1\Lambda V_1^T$ .

Para el **segundo ejercicio**, podemos tomar por ejemplo  $G = U_1 \in \mathcal{M}_{n,r}(\mathbb{R})$  y  $H = V_1\Lambda \in \mathcal{M}_{d,r}(\mathbb{R})$ . Para terminar, podemos denotar  $u_1, \dots, u_r$  los vectores columna de  $U_1$  y  $v_1, \dots, v_r$  los vectores columna de  $V_1$ . Abusamos la notación

$$u_i = (0 \mid u_i \mid 0)$$

de tal manera que  $U_1 = \sum_i u_i$  y  $V_1 = \sum_i v_i$ . Finalmente tenemos  $U_1\Lambda V_1^T = (\sum_i u_i)\Lambda(\sum_i v_i)^T = \sum_i \lambda_i u_i v_i^T$ .

## 2.2.2. SVD: aplicación

Una propiedad simpática es la siguiente:

### Teorema (Eckart-Young)

Sea  $A$  una matriz de  $\mathcal{M}_{n,d}(\mathbb{R})$  y sea  $k < \min(n, d)$ . Denotamos  $A = \sum_{i=1}^d \lambda_i u_i v_i^T$  su descomposición en valores singulares donde los valores singulares están ordenados de manera decreciente  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ . Entonces, la mejor aproximación de rango menor

$$A_k = \operatorname{argmin}_{\operatorname{rango}(B)=k} \|A - B\|$$

está dada por

$$A_k = \sum_{i=1}^k \lambda_i u_i v_i^T$$

donde la norma  $\|\cdot\|$  es la norma de Frobenius o la norma espectral.

## 2.2.2. SVD: aplicación

Recordatorio:

- ▶ Norma de Frobenius:  $\|A\|_F = (\text{tr}(A^T A))^{1/2}$ .
- ▶ Norma espectral:  $\|A\|_2^2 = \sup_{u:\|u\|=1} (A^T A u, u)$ .

Se pueden calcular las distancias de error por ejemplo:

$$\|A - A_k\|_2 = \lambda_{k+1}$$

- ▶ Se puede aproximar matrices de  $\mathcal{M}_{n,d}(\mathbb{R})$  que requiere  $n \times d$  parámetros por  $k$  productos de vectores. Eso requiere  $k \times (1 + n + d)$  valores.
- ▶ Si  $k$  es pequeño,  $k \times (1 + n + d)$  es mucho menor que  $n \times d$   
 $\Rightarrow$  Compresión de datos!

## 2.2.3. SVD: Compleción de matrices de rango bajo

Imaginamos el contexto siguiente:

							...
	★★★★★	?	★★★★☆	?	?	?	...
	?	★★★☆☆	?	?	★★★★☆	?	...
	?	?	?	★★★★☆	★★★★☆	?	...
	?	★★★★☆	★★★★☆	?	?	★★★★★	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

## 2.2.3. SVD: Compleción de matrices de rango bajo

Tenemos una matriz  $A$  de rango pequeño de la cual no observamos todas las entradas.

Existe un conjunto  $\Omega$  que corresponde a los pares de índices  $(i, j)$  donde la matriz  $A$  está observada y una proyección  $P_\Omega$  asociada.

Entonces la matriz  $P_\Omega(A)$  tiene las entradas  $p_{i,j} = a_{i,j}$  cuando  $(i, j) \in \Omega$  y  $p_{i,j} = 0$  si no. La propuesta de la completación de  $A$  es la siguiente:

$$\underset{M}{\text{minimize}} H(M) := \frac{1}{2} \|P_\Omega(A - M)\|_F^2 + \lambda \|M\|_*$$

donde  $\|\cdot\|_*$  es la norma nuclear (la suma de los valores singulares).

## 2.2.3. SVD: Compleción de matrices de rango bajo

Se propone el algoritmo iterativo siguiente:

1. Reemplazar las entradas de  $A$  faltantes por las entradas de la estimación corriente  $\hat{M}$ .

$$\hat{A} = P_{\Omega}(A) + P_{\Omega}^{\perp}(\hat{M})$$

2. Update de la matriz  $\hat{M}$  con el soft-threshold:

$$\hat{A} = UDV^T$$

$$\hat{M} = US_{\lambda}(D)V^T$$

donde  $S_{\lambda}$  opera sobre cada elemento de la diagonal con la función  $x \mapsto (x - \lambda)_+$ .

Por ejemplo, si  $\lambda$  es muy grande, muchos de los valores diagonales de  $D$  se fijan en 0.

## 2. Búsqueda de estructura: Cambio de espacio

2.3. Curso del 3 de Octubre

## 2.3.1. Factorización no-negativa:intuición

Vimos la factorización SVD importante  $A = UDV^T$  para cualquier tamaño  $n \times d$  de matriz.

- ▶ Permite una descomposición en matrices de rango 1.
- ▶ Permite una aproximación por matrices de rango menor.
- ▶ Se puede escribir  $A = GH^T$  donde las matrices  $G$  y  $H$  tienen el rango de  $A$ .

De vez en cuando se necesita una otra aproximación parecida pero diferente: **Aproximación por matrices no-negativas.**



## 2.3.1. Factorización no-negativa: intuición

**Contexto:** Imaginamos que  $A$  es una matriz de entradas no-negativas:  $a_{i,j} \geq 0$ . Supongamos que  $A$  se escriba  $A = UV^T$  donde las entradas de  $U \in \mathcal{M}_{n,k}(\mathbb{R}_+)$  y  $V \in \mathcal{M}_{d,k}(\mathbb{R}_+)$  sean no-negativas.

Los vectores renglones  $(a_i^T)_i$  de  $A$  se escriben

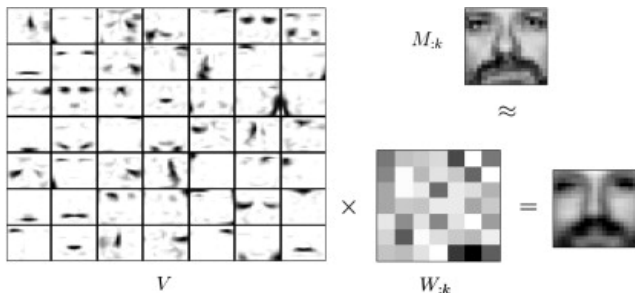
$$A = \begin{pmatrix} a_1^T \\ \dots \\ a_n^T \end{pmatrix} = \begin{pmatrix} u_1^T \\ \dots \\ u_n^T \end{pmatrix} \times V^T$$

Entonces, cada  $a_i$  se escribe como una combinación lineal de vectores columna de  $V$ :

$$a_i = u_{i,1}v_1 + \dots + u_{i,k}v_k$$

## 2.3.1. Factorización no-negativa: intuición

Los  $u_{i,j}$  se interpretan como pesos de una base de vectores de atributos dado por  $v_1, \dots, v_k$ .



Cada vector  $a_i$  corresponde a una imagen y se combinan vectores de referencia  $v_j$  para obtener la imagen final.

$\Rightarrow$  Es una aproximación!

## 2.3.2. Factorización no negativa (NMF):definición

En general no existen matrices con entradas no negativas tal que  $A = UV^T$ . Se hace la aproximación

$$(U^*, V^*) = \operatorname{argmin}_{U, V \geq 0} \frac{1}{2} \|A - UV^T\|_F^2 =: \operatorname{argmin}_{U, V \geq 0} J(U, V).$$

La cantidad  $J$  se puede escribir

$$J(U, V) = \frac{1}{2} \operatorname{tr}(AA^T + UV^T VU^T - AVU^T - UV^T A^T).$$

El truco habitual para minimizar una funcional bajo restricciones es considerar el Lagrangiano:

$$L(U, V) = J(U, V) - \sum_{i=1}^n \sum_{j=1}^k \alpha_{i,j} u_{i,j} - \sum_{i=1}^d \sum_{j=1}^k \beta_{i,j} v_{i,j}$$

## 2.3.2. Factorización no negativa (NMF):definición

Si consideramos la notación  $P_\alpha = (\alpha_{i,j})_{i,j}$  y  $P_\beta = (\beta_{i,j})_{i,j}$  entonces el Lagrangiano se escribe

$$L(U, V) = J(U, V) - \text{tr}(P_\alpha U^T) - \text{tr}(P_\beta V^T).$$

Ahora la tarea se reduce a minimizar la funcional  $L$ . Para eso, podemos considerar una noción extendida de derivación: la derivación matricial.

### 2.3.3. Paréntesis : Derivación matricial

Por definición formal se considera la derivación de una función a valores reales de una matriz  $A$  como la matriz de derivadas al respecto de cada entrada de  $A$ . Formalmente,

$$\frac{df(A)}{dA} = \left( \frac{df(A)}{da_{i,j}} \right)_{i,j}$$

For example,

$$\frac{d\text{tr}(AB)}{dA} = B^T, \quad \frac{d\text{tr}(A^T A)}{dA} = 2A, \quad \frac{d\text{tr}(A^T BA)}{dA} = (B+B^T)A$$

**Ejercicio:** Mostrar esas formulas.

## 2.3.4. (NMF) fin de calculo

La derivación de  $L$  al respecto de  $U$  y  $V$  da las dos ecuaciones matriciales:

$$\begin{aligned} -AV + UV^T V - P_\alpha &= 0 \\ -A^T U + VU^T U - P_\beta &= 0 \end{aligned}$$

y las condiciones de positividad dan

$$\begin{aligned} \alpha_{i,j} u_{i,j} &= 0 \quad \forall i,j \\ \beta_{i,j} v_{i,j} &= 0 \quad \forall i,j. \end{aligned}$$

La combinación de esas ecuaciones permite obtener las relaciones

$$\begin{aligned} (AV)_{ij} u_{ij} &= (UV^T V)_{ij} u_{ij} \\ (A^T U)_{ij} v_{ij} &= (VU^T U)_{ij} v_{ij} \end{aligned}$$

## 2.3.4. (NMF) fin de calculo

Para resolver este sistema, se considera una aproximación iterativa:

1. Inicialización: Tomar  $U_0$  y  $V_0$  positivas (al azar por ejemplo)
2. A cada iteración se calculan todos los

$$u_{ij}^{(t+1)} = \frac{(U^{(t)}(V^{(t)})^T V^{(t)})_{ij} u_{ij}^{(t)}}{(AV^{(t)})_{ij}}$$
$$v_{ij}^{(t+1)} = \frac{(V^{(t)}(U^{(t)})^T U^{(t)})_{ij} v_{ij}^{(t)}}{(A^T U^{(t)})_{ij}}$$

3. Usar un criterio de paro para terminar el algoritmo.

Los valores  $U^{(t)}$  y  $V^{(t)}$  de salida forman la factorización aproximada de  $A$ .

Para una referencia de convergencia ver:

*Algorithms for Non-negative Matrix Factorization - Lee, Seung*

## 2. Búsqueda de estructura: Cambio de espacio

2.4. Curso del 10 de octubre



## 2.4.1. PCA: Análisis en componentes principales

Supongamos una vez más tener una matriz de datos  $A \in \mathcal{M}_{n,p}(\mathbb{R})$  donde  $n$  es el número de individuos de la muestra y  $p$  el número de atributos.

En general podemos tener todos los casos:  $n \gg p$ ,  $n > p$ ,  $n \approx p$ ,  $n < p$  o  $n \ll p$ .

Las herramientas vistas en las clases anteriores permiten hacer una reducción de dimensionalidad (Eckart-Young, Factorización no negativa, Criterio de Fisher,...).

Vamos a ver una otra técnica (como caso particular de Eckart-Young).

## 2.4.1. PCA: Análisis en componentes principales

Aquí se supone que las variables correspondiendo a las filas son i.i.d. (idénticamente distribuidas según una distribución desconocida y independientes). La matriz de varianzas de  $A$  está dado por  $\Sigma = A^T A \in \mathcal{M}_p(\mathbb{R})$ .

Queremos “proyectar” los datos  $X_i$  (las filas) sobre un espacio de dimensión más bajo que  $p$  (su dimensión natural) de tal manera que se guarda la mayor variación posible dentro de los datos.

Entonces, si  $G$  corresponde a la matriz de proyección, los nuevos datos son  $y_i = GX_i^T$ . La matriz  $G$  es de tamaño  $k \times p$  con  $k \leq p$ . La nueva matriz de varianzas está dada por

$$\text{Var}(y) = G\Sigma G^T$$

Para tener una cantidad en  $\mathbb{R}_+$ , se considera la varianza total dada por  $\text{tr}(G\Sigma G^T)$ .

## 2.4.1. PCA: Análisis en componentes principales

Para una  $k$  dada, definemos la transformación  $G$  óptima como la matriz de proyección que maximice la varianza total. Formalmente, queremos resolver el problema de optimización siguiente:

$$\max_{G: GG^T = I_k} \text{tr}(G\Sigma G^T) = \lambda_1 + \dots + \lambda_k$$

donde las  $\lambda_1, \dots, \lambda_k$  corresponden a los  $k$  valores propios más grandes de la matriz  $\Sigma$ .

La solución está dada por la matriz  $G^T = [t_1, \dots, t_k]$  de los vectores propios (ortogonales) correspondiendo a los valores propios  $\lambda_1, \dots, \lambda_k$ .

## 2.4.2. PCA: Un lemma importante

Al final la cosa a mostrar es el

### Teorema

Sea  $A$  una matriz simétrica  $n \times n$  con descomposición espectral

$$A = T\Delta T^T \quad \Delta = \text{diag}(\delta_1, \dots, \delta_n)$$

donde supongamos que los valores  $\delta_1, \dots, \delta_n$  están ordenados de manera decreciente. Tenemos  $TT^T = T^T T = I_n$ . Sea  $k \leq n$ , entonces

$$\max_{GG^T=I_k} \text{tr}(GAG^T) = \delta_1 + \dots + \delta_k.$$

Y la cota superior se alcanza para  $G^T = [t_1, \dots, t_k]$  la matriz de los vectores propios (ortogonales) correspondiendo a los  $k$  valores propios más grandes.

## 2.4.2. PCA: Un lemma importante

- ▶ Sea  $P = T^T G^T G T$ , entonces

$$\text{tr}(GAG^T) = \text{tr}(GT\Delta T^T G^T) = \text{tr}(T^T G^T G T \Delta) = \text{tr}(P\Delta)$$

- ▶ Tenemos

$$P^2 = T^T G^T G T T^T G^T G T = T^T G^T G G^T G T = T^T G^T G T = P$$

entonces  $P$  es un proyector.

- ▶ Fácilmente, tenemos  $P^T = P$  entonces el proyector es ortogonal.
- ▶ En particular sabemos que  $\text{tr}(P) = \text{rango}(P)$  lo que implica  $p_{11} + \dots + p_{nn} = k$  y que  $0 \leq p_{ij} \leq 1$  (Usar  $P = P^T P$ ).
- ▶ Finalmente,

$$\text{tr}(GAG^T) = p_{11}\delta_1 + \dots + p_{nn}\delta_n \leq \delta_1 + \dots + \delta_k$$

## 2.4.2. PCA: Un lemma importante

Al final si se elige  $G^T = [t_1, \dots, t_k]$ , tenemos la proyección  $P = T^T G^T G T$  que se reduce a

$$P = \begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}$$

y entonces tenemos que  $p_{11} = \dots = p_{kk} = 1$  y  $p_{jj} = 0$  para  $j > k$ . Entonces, encontramos una matriz que alcanza la cota superior.

## 2.4.3. PCA: Solución general

Además de maximizar  $\text{tr}(G\Sigma G^T)$  vimos que la descomposición en componentes principales tiene las propiedades:

Sea  $y = Gx^T$  la transformación de la variable  $x$  en un espacio de dimensión  $k$ . Denotamos  $y_1, \dots, y_k$  los componentes de  $y$ .

1. Tenemos  $\text{Var}(y_1) \geq \dots \geq \text{Var}(y_k)$ .
2. Las variables  $y_i$  y  $y_j$  son descorrelacionadas.

**Ejercicio:** Mostrar esas dos propiedades.

## 2. Búsqueda de estructura: Cambio de espacio

2.5. Curso del 12 de octubre



## 2.5.1. Proyecciones aleatorias

Como en el caso de PCA, queremos modificar los datos  $x_1, \dots, x_n$  por una matriz  $P$  de tal manera que los datos  $y_i = Px_i$  conservan la mayoría de la información.

El enfoque de esta parte es de conservar algo de las distancias  $\|x_i - x_j\|$ : aspecto geométrico.

Solución: Lema de Johnson-Lindenstrauss.

La mayoría de las proyecciones (al azar) van a funcionar. Hay una restricción sobre la dimensión más baja sobre la cual se hace la proyección.

## 2.5.1. Proyecciones aleatorias

### Lema (Johnson-Lindenstrauss)

Sean  $u_1, \dots, u_n$  puntos arbitrarios de  $\mathbb{R}^p$ . Definemos la matriz  $P \in \mathcal{M}_{p,k}(\mathbb{R})$  como

$$p_{ij} = \frac{1}{\sqrt{k}} z_{ij} \quad \text{donde} \quad z_{ij} \sim \mathcal{N}(0, 1)$$

Sean  $v_1, \dots, v_n$  los  $n$  puntos obtenidos por  $v_i = P^T u_i$ . Entonces con probabilidad al menos  $1/2$ ,

$$(1 - \epsilon) \|u_i - u_j\|^2 \leq \|v_i - v_j\|^2 \leq (1 + \epsilon) \|u_i - u_j\|^2$$

con  $\epsilon \in (0, 1/2)$  y  $k \geq 9 \frac{\log n}{\epsilon^2 - \epsilon^3}$ .

## 2.5.2. Prueba del lema de J-L

Si denotamos  $u \in \mathbb{R}^p$  y  $v = P^T u$ . Entonces  $\mathbb{E} [\|v\|^2] = \|u\|^2$ .  
Denotamos  $P = [p_1, \dots, p_k]$  en columnas. Entonces,

$$\begin{aligned}\mathbb{E} [\|v\|^2] &= \mathbb{E} \left[ \sum_{j=1}^k v_j^2 \right] = \mathbb{E} \left[ \sum_{j=1}^k (p_j^T u)^2 \right] = \mathbb{E} \left[ \sum_{j=1}^k \left( \sum_{i=1}^p p_{ij} u_i \right)^2 \right] \\ &= \frac{1}{k} \mathbb{E} \left[ \sum_{j=1}^k \left( \sum_{i=1}^p z_{ij} u_i \right)^2 \right] = \frac{1}{k} \sum_{j=1}^k \mathbb{E} \left[ \left( \sum_{i=1}^p z_{ij} u_i \right)^2 \right] \\ &= \frac{1}{k} \sum_{j=1}^k \text{Var} \left( \sum_{i=1}^p z_{ij} u_i \right) = \frac{1}{k} \sum_{j=1}^k \sum_{i=1}^p u_i^2 = \|u\|^2\end{aligned}$$

## 2. Búsqueda de estructura: Cambio de espacio

2.6. Curso del 17 de octubre

## 2.6.1. Prueba del lema de J-L

Para  $j = 1, \dots, k$  definamos

$$x_j = \frac{\sqrt{k}}{\|u\|} p_j^T u = \frac{\sqrt{k}}{\|u\|} v \quad \implies \quad \|x\|^2 = \frac{k}{\|u\|^2} \|v\|^2$$

Denotamos  $y = \|x\|^2$ .

$$\begin{aligned} \mathbb{P}(\|v\|^2 \geq (1 + \epsilon)\|u\|^2) &= \mathbb{P}(y \geq (1 + \epsilon)k) \\ &= \mathbb{P}(e^{\lambda y} \geq e^{\lambda(1+\epsilon)k}) \\ &\leq \frac{\mathbb{E}[e^{\lambda y}]}{e^{\lambda(1+\epsilon)k}} \\ &\leq \frac{\mathbb{E}[\exp(\lambda \sum_j x_j^2)]}{e^{\lambda(1+\epsilon)k}} = \frac{1}{e^{\lambda(1+\epsilon)k}} \prod_j \mathbb{E}[e^{\lambda x_j^2}] \\ &\leq \frac{1}{e^{\lambda(1+\epsilon)k}} \frac{1}{(1 - 2\lambda)^{k/2}} = \left( \frac{e^{-2\lambda(1+\epsilon)}}{1 - 2\lambda} \right)^{k/2} \end{aligned}$$

## 2.6.1. Prueba del lema de J-L

Sabemos que  $\mathbb{P}(\|v\|^2 \geq (1 + \epsilon)\|u\|^2) \leq \left(\frac{e^{-2\lambda(1+\epsilon)}}{1-2\lambda}\right)^{k/2}$  entonces si  $\lambda = \epsilon/2(1 + \epsilon)$  tenemos que

$$\mathbb{P}(\|v\|^2 \geq (1 + \epsilon)\|u\|^2) \leq ((1 + \epsilon)e^{-\epsilon})^{k/2}$$

y debido a que  $(1 + \epsilon) \leq e^{\epsilon - (\epsilon^2 - \epsilon^3)/2}$  (cuidado no tenemos  $(1 + \epsilon) \leq e^{\epsilon - \epsilon^2/2}$ ),

$$\mathbb{P}(\|v\|^2 \geq (1 + \epsilon)\|u\|^2) \leq e^{-k(\epsilon^2 - \epsilon^3)/4}.$$

De la misma manera tenemos que

$$\mathbb{P}(\|v\|^2 \leq (1 - \epsilon)\|u\|^2) \leq e^{-k(\epsilon^2 - \epsilon^3)/4}$$

En particular, si resumimos las dos desigualdades, tenemos que

$$\mathbb{P}((1 - \epsilon)\|u\|^2 \leq \|v\|^2 \leq (1 + \epsilon)\|u\|^2) \geq 1 - 2e^{-k(\epsilon^2 - \epsilon^3)/4}$$

## 2.6.1. Prueba del lema de J-L

Resumimos: Mostramos que para cualquier  $u$ , y para  $v = P^T u$ .

Si lo aplicamos para una diferencia  $u_i - u_j$ , podemos denotar  $\Delta_{i,j}$  el evento “malo”

$$(1 - \epsilon) \|u_i - u_j\|^2 \geq \|v_i - v_j\|^2 \quad \text{o} \quad \|v_i - v_j\|^2 \leq (1 + \epsilon) \|u_i - u_j\|^2$$

tenemos  $\mathbb{P}(\Delta_{i,j}) \leq 2e^{-k(\epsilon^2 - \epsilon^3)/4}$ . Podemos usar la desigualdad para las  $n(n-1)/2$  pares de puntos. Entonces,

$$\mathbb{P}\left(\bigcup_{i,j} \Delta_{i,j}\right) \leq n(n-1)e^{-k(\epsilon^2 - \epsilon^3)/4} \leq n^2 e^{-k(\epsilon^2 - \epsilon^3)/4}$$

Entonces, si tomamos  $k \geq 4 \log(2n^2)/(\epsilon^2 - \epsilon^3)$  tenemos  $\mathbb{P}(\cup_{i,j} \Delta_{i,j}) \leq 1/2$ .

## 2.6.2. Ventajas de las proyecciones de J-L

Resumimos lo visto:

- ▶ Es un resultado de reducción de dimensionalidad. Nos dice que si repetimos el experimento de lanzar al azar  $P$ , muy rápidamente (con esperanza  $\mathbb{E}[\mathcal{E}(1/2)] = 2$ ) obtendremos una  $P$  que satisface la casi-isometría.
- ▶ La dimensión del espacio reducido es del orden  $\log(n)$  y **no** depende de la dimensión de origen  $p$  del espacio ambiente.



## 2.6.3. Resumen global de reducción de dimensionalidad

Hemos estudiado varias técnicas que permiten pre-procesar los datos:

1. SVD (descomposición en valores singulares)
2. Complementación de matrices de rango bajo.
3. Factorización no-negativa (NMF)
4. PCA : analisis en componentes principales.
5. Proyecciones aleatorias (Lema de J-L)

Cada técnica tiene un rol preciso y tiene su lista de ventajas-desventajas.

## 3.Agrupamiento de datos (Clustering)

### 3.1.Clase del 26 de Octubre

### 3.1.1. k-medias:definición

El contexto estadístico es idéntico a la problemática de clasificación. Se supone que tengamos  $x_1, \dots, x_n$  datos de un espacio ambiente  $\mathbb{R}^p$ .

- ▶ Queremos  $k$  clusters  $C_1, \dots, C_k$  que forman una partición de los datos  $x_1, \dots, x_n$ .
- ▶ En  $k$ -medias nos referimos a los centroides de las clases:

$$\mu(C_i) = \operatorname{argmin}_{\mu \in \mathbb{R}^p} \sum_{x \in C_i} d^2(x, \mu)$$

- ▶ La solución es explícita para cada una de las  $k$  clases y dada por

$$\mu(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} x \quad (k\text{-medias})$$

**Ejercicio:** Demostrar este hecho.

### 3.1.1. k-medias:definición

- ▶ La función de riesgo que se considera es

$$G(C_1, \dots, C_k) = \sum_{i=1}^k \sum_{x \in C_i} d^2(x, \mu(C_i)).$$

- ▶ El objetivo de  $k$ -medias es encontrar la mejor partición  $C_1, \dots, C_k$  en el sentido de la minimización de  $G(C_1, \dots, C_k)$ .
- ▶ Dado el centroide, el  $C_i$  que minimiza

$$\sum_{x \in C_i} d^2(x, \mu(C_i))$$

está dado por

$$C_i = \{x : i = \operatorname{argmin}_j d^2(x, \mu(C_j))\}$$

## 3.1.2. El algoritmo $k$ -medias

1. Inicialización: Escoger aleatoriamente  $k$  centroides

$$\mu_1^0, \dots, \mu_k^0 \in \mathbb{R}^d.$$

2. Hasta que los centroides no cambien:

- 2.1 En la iteración  $t$ , para  $i = 1, \dots, k$  definir los clusters

$$C_i^t = \left\{ x : i = \underset{j}{\operatorname{argmin}} \|x - \mu_j^{t-1}\|^2 \right\}$$

- 2.2 Para cada cluster  $C_i$  calcular

$$\mu_i^t = \frac{1}{|C_i^t|} \sum_{x \in C_i^t} x$$

La propiedad importante es que la función de riesgo es decreciente.

### 3.1.3. Convergencia de $k$ -medias

- ▶ Supongamos construidos los clusters  $C_1^{t-1}, \dots, C_k^{t-1}$  y los centroides  $\mu_1^{t-1}, \dots, \mu_k^{t-1}$  de la etapa  $t-1$ .
- ▶ Entonces,

$$G(C_1^t, \dots, C_k^t) = \sum_{i=1}^k \sum_{x \in C_i^t} \|x - \mu_i^t\|^2 \leq \sum_{i=1}^k \sum_{x \in C_i^t} \|x - \mu_i^{t-1}\|^2$$

- ▶ Por otro lado, los clusters  $C_i^{t-1}$  minimizan la cantidad

$$\sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i^{t-1}\|^2$$

sobre la clase de todos los clusters posibles. Entonces,

$$\sum_{i=1}^k \sum_{x \in C_i^t} \|x - \mu_i^{t-1}\|^2 \leq \sum_{i=1}^k \sum_{x \in C_i^{t-1}} \|x - \mu_i^{t-1}\|^2$$

### 3.1.3. Convergencia de $k$ -medias

Mostramos que  $\forall t$ ,

$$G(C_1^t, \dots, C_k^t) \leq G(C_1^{t-1}, \dots, C_k^{t-1}).$$

Entonces  $t \mapsto G(C_1^t, \dots, C_k^t)$  es decreciente. Como las particiones de los datos forma un conjunto finito, a partir de una cierta etapa  $G(C_1^t, \dots, C_k^t)$  es constante.

El algoritmo es convergente.

### 3.1.4. Una alternativa: $k$ -medoides

- ▶ La idea principal es de reemplazar la distancia euclidianna por una medida de disimilaridad  $D$  entre puntos de la muestra.
- ▶ Se pierde la noción de centroides iguales a las medias empíricas de cada cluster.  
Los medoides serán siempre elementos de la muestra misma.
- ▶ Algoritmo PAM (Partition Around Medoids).

1. Se inicia con una partición dada  $C_1, \dots, C_k$ .
2. Hasta la convergencia se hace:

2.1 Se encuentran los medoides según la formula

$$m_i = \operatorname{argmin}_{x \in C_i} \sum_{y \in C_i} D(x, y)$$

2.2 Se actualizan los clusters asignando los datos más cercanos (en el sentido de  $D$ ) a los medoides.



### 3.1.5. Una otra alternativa: $k$ -centros

- ▶ La idea es optimizar una otra función de riesgo

$$G(C_1, \dots, C_k) = \max_j \max_{x \in C_j} d(\mu_j, x)$$

- ▶ En general es un problema difícil si permitimos que los centros  $\mu_j$  sean elementos genéricos de  $\mathbb{R}^d$ .
- ▶ El algoritmo de aproximación es más sencillo que  $k$ -means. Los centroides se pueden elegir sucesivamente.

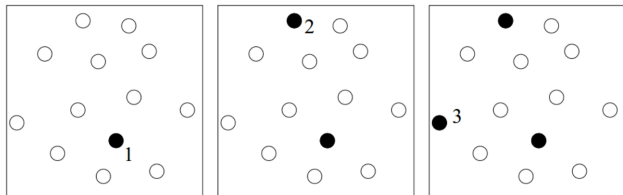
#### **Farthest-First Traversal**

1. Inicio con un punto  $\mu_1$  arbitrario.
2. Para cada punto de los datos  $x \in \mathcal{D}$ , calcular  $d(\mu_1, x)$  y elegir el punto  $\mu_2$  el más alejado.
3. A la  $i$  ésima interacción se identifica el punto  $\mu_i$  realizando el

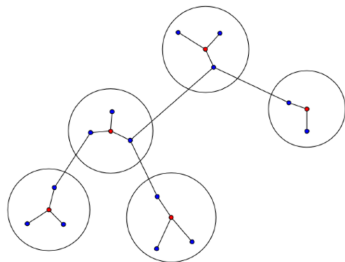
$$\operatorname{argmax}_{x \in \mathcal{D}} \min_{j < i} d(x, \mu_j).$$

### 3.1.5. Una otra alternativa: $k$ -centros

Una visualización del algoritmo:



Al final se obtiene una partición:



# 3.Agrupamiento de datos (Clustering)

## 3.2.Clase del 31 de octubre

### 3.2.1. Agrupamiento jerárquico: definición

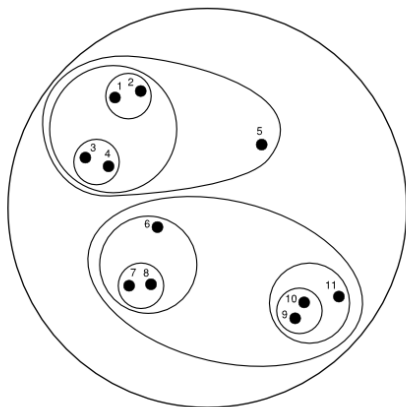
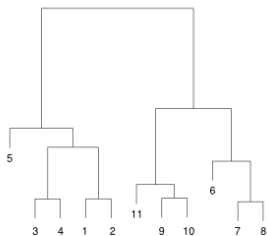
Entramos en una nueva manera de agrupar: Agrupamiento jerárquico.

- ▶ Son agrupamientos sucesivos. Al tiempo  $t = 1$  los clusters corresponden a la partición las más fina  $\{1\}, \dots, \{n\}$ . Para simplificar las notaciones confundimos los clusters sobre los índices con los clusters sobre los datos  $x_1, \dots, x_n$ .
- ▶ Si  $C_1^t, \dots, C_k^t$  es la partición de los datos al tiempo  $t$  entonces al tiempo  $t + 1$  la nueva partición es tal que

$$(C_i^t)_i \text{ es más fina que } (C_i^{t+1})_i$$

- ▶ Al terminar el proceso, obtenemos un único cluster  $C_1^n = \{1, \dots, n\}$  que contiene todos los datos.

## 3.2.1. Agrupamiento jerárquico: definición



### 3.2.1. Agrupamiento jerárquico: definición

Supongamos que tenemos acceso a las distancias  $d(x_i, x_j)$  entre parejas de datos.

Una vez que se agruparon datos, tenemos distancias entre clusters. Eso da luz a varias maneras de proceder.

El agrupamiento jerárquico tiene *cuatro* versiones populares:

- ▶ Simple linkage: la distancia mínima entre puntos de cada cluster.

$$\rho(C, D) = \inf_{x \in C, y \in D} d(x, y)$$

- ▶ Complete linkage: la distancia máxima entre puntos de cada cluster.

$$\rho(C, D) = \sup_{x \in C, y \in D} d(x, y)$$

(diametro del cluster  $C \cup D$ )

### 3.2.1. Agrupamiento jerárquico: definición

- ▶ Average linkage: la distancia promedio entre los clusters.

$$\rho(C, D) = \frac{1}{|C||D|} \sum_{x \in C} \sum_{y \in D} d(x, y)$$

- ▶ Ward linkage: la varianza promedio dentro de la unión de dos clusters.

$$\rho(C, D) = \frac{1}{|C| + |D|} \sum_{x, y \in C \cup D} d(x, y)^2$$

En cada tiempo  $t$ , se agrupan clusters que corresponden a la distancia mínima entre clusters,

$$C, D = \operatorname{argmin}_{C, D \in (C_i^t)_i} \rho(C, D)$$

Los clusters al tiempo  $t + 1$  son  $((C_i^t)_i \setminus \{C, D\}) \cup \{C \cup D\}$

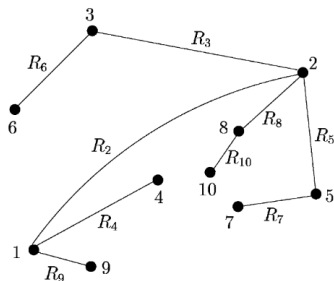
## 3.2.2. Un algoritmo basado en FFT

Retomamos el algoritmo FFT:

**Input:**  $n$  puntos con sus distancias respectivas  $d(x_i, x_j)$ .

1. Al azar elegir un punto  $x_i$  y etiquetarlo 1.
2. Para  $i = 2, \dots, n$ 
  - ▶ Buscar el punto el más alejado de  $\{1, \dots, i-1\}$  y etiquetarlo  $i$ .
  - ▶ Sea  $\pi(i) = \operatorname{argmin}_{j < i} d(i, j)$ .
  - ▶ Sea  $R_i = d(i, \pi(i))$ .

**Output:**





## 3.2.2. Un algoritmo basado en FFT

Para definir el agrupamiento,

- ▶ Definimos los centros  $\mu_1, \dots, \mu_k$  como los puntos  $x_j$  que corresponden a las etiquetas  $1, \dots, k$ .
- ▶ Los puntos se asignan a su centroide el más cercano. (Es  $k$ -centros)
- ▶ Los  $R_i$  son las distancias de  $i$  a su padre  $\pi(i)$ .

Si denotamos  $R(C_i) = \max_{x \in C_i} d(x, \mu_i)$  el riesgo, tenemos

**Lema**

*Tenemos*

- ▶  $R_2 \geq \dots \geq R_n$
- ▶ Para cada  $k$ ,  $R(C_k) = R_{k+1}$ .

## 3.2.2. Un algoritmo basado en FFT

### Prueba:

Tomando  $i < j$ ,

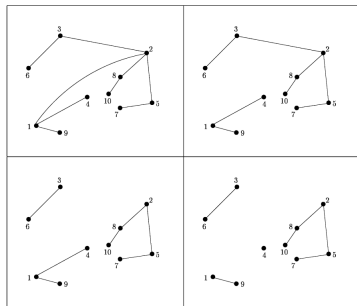
$$\begin{aligned}R_j &= d(j, \pi(j)) \\ &= d(j, \{1, \dots, j-1\}) \\ &\leq d(j, \{1, \dots, i-1\}) \\ &\leq d(i, \{1, \dots, i-1\}) = R_i\end{aligned}$$

Para el segundo punto, la distancia de un punto  $i > k$  a su punto más cercano es

$$d(i, \{1, \dots, k\}) \leq d(k+1, \{1, \dots, k\}) = R_{k+1}$$

### 3.2.3. Agrupamiento jerárquico con FFT

- ▶ Sea  $\rho : \{2, \dots, n\} \rightarrow \{1, \dots, n\}$  tal que  $\rho(i) < i$ .
- ▶ La gráfica con vertices  $\{1, \dots, n\}$  y aristas  $\{(i, \rho(i)) : 2 \leq i \leq n\}$  es un árbol denotado  $T_\rho$ .
- ▶ Para cada  $k$ , el agrupamiento está definido:
  - ▶ Quitamos las  $k - 1$  aristas  $(2, \rho(2)), \dots, (k, \rho(k))$  de  $T_\rho$ . Eso deja  $k$  componentes.
  - ▶ Cada grupo es uno de las componentes restantes.



# 3.Agrupamiento de datos (Clustering)

3.3.Curso 7 de noviembre

### 3.3.1. Agrupamiento espectral: definición

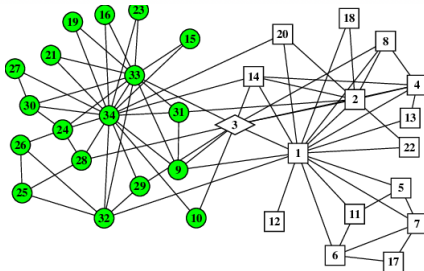
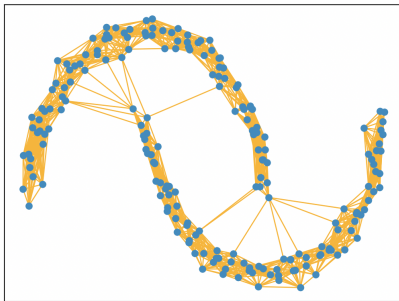
En este tipo de agrupamiento, usaremos una vez más las ideas de descomposición espectral de una matriz. Supongamos que tenemos acceso a una matriz de similitud simétrica  $S \in \mathcal{M}_n(\mathbb{R})$ .

- ▶ Imaginamos que podemos organizar los datos en una forma de grafo  $G = (V, E)$  donde los vertices  $V = \{v_1, \dots, v_n\}$  son los datos.
- ▶ El grafo es ponderado en el sentido de que existe una matriz  $W = (w_{i,j})_{1 \leq i,j \leq n}$  de tal manera que si  $w_{i,j} = 0$  los vertices  $v_i$  y  $v_j$  no están conectados.
- ▶ Consideramos la noción siguiente de grado para el vertice  $i$ :

$$d_i = \sum_{j=1}^n w_{i,j}$$

y la matriz de grados  $D$  está definida como la matriz diagonal con los  $(d_i)_i$  en la diagonal.

### 3.3.1. Agrupamiento espectral: definición



### 3.3.2. Grafos populares

**$\epsilon$ -vecindad:** Dos vertices  $v_i$  y  $v_j$  se conectan si su distancia es menor a  $\epsilon$ . El peso  $w_{i,j}$  asociado es 1 si están conectados y zero si no.

**$k$ -vecinos más cercanos:** ( *$k$ -vecinos más cercanos*) Un vertice  $v_i$  se conecta a  $v_j$  si es uno de sus  $k$  vecinos más cercanos. El peso  $w_{i,j}$  asociado es  $d(v_i, v_j)$  (o su disimilaridad  $D(v_i, v_j)$  en el caso generalizado) la distancia entre los dos puntos.

( *$k$ -vecinos más cercanos mutual*) Dos vertices  $v_i$  y  $v_j$  se conectan si  $v_j$  es un  $k$  vecinos más cercanos de  $v_i$  y  $v_i$  es un  $k$  vecinos más cercanos de  $v_j$ .

**Grafos completos:** Cada par de vertices están conectados. El peso  $w_{i,j}$  asociado es una función de similaridad. Por ejemplo,

$$w_{i,j} = \exp(-\|x_i - x_j\|^2 / 2\sigma^2).$$

### 3.3.3. Noción de tamaño en un grafo

Vamos a tener que considerar sub-grafos de  $G$ .

- ▶ Para  $A$ , un subconjunto de  $V$ ,  $i \in A$  es el conjunto de índices  $\{i \mid v_i \in A\}$ .
- ▶ El tamaño de un conjunto  $A$  se puede definir como:

$$|A| = \sum_{i \in A} 1 = \text{el número de elementos de } A.$$

$$\text{vol}(A) = \sum_{i \in A} d_i$$

- ▶ Para dos sub-conjuntos  $A, B$  de  $V$ , definimos

$$W(A, B) = \sum_{i \in A, j \in B} w_{i,j}$$



### 3.3.4. El Laplaciano

Definimos la matriz  $L = D - W$ . Se llama **Laplaciano no normalizado**.

#### Proposición

*Tenemos los siguiente*

1.  $L$  es simétrico y definido positivo.
2. Para un vector  $u \in \mathbb{R}^n$ ,

$$u^T L u = \frac{1}{2} \sum_{i,j=1}^n w_{i,j} (u_i - u_j)^2$$

3.  $0$  es valor propio de  $L$  y el vector  $\mathbf{1}$  (con  $1$  en cada entrada) es vector propio asociado.

**Ejercicio:** Demostrar la proposición.

### 3.3.4. El Laplaciano

El Laplaciano es importante por el lema siguiente:

#### Lema (fundamental)

*Sea  $G$  un grafo con pesos  $(w_{i,j})$  no negativos. Denotamos  $k$  la multiplicidad del valor propio 0 de  $L$ . Entonces, el número de componentes conectadas  $A_1, \dots, A_k$  de  $G$  es igual a  $k$ . Además, el espacio propio asociado al valor propio 0 está generado por los vectores  $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ ,*

$$\mathbb{1}_{A_i} = (u_1, \dots, u_n) \quad \text{donde } u_j = 1 \text{ si y solo si } v_j \in A_i.$$

### 3.3.4. El Laplaciano

**Prueba:** Los pesos  $w_{i,j}$  tal que  $i \in A_s$  y  $j \in A_t$  son todos iguales a 0 por definición. En efecto,  $A_s$  y  $A_t$  no se conectan por ninguna arista. Entonces, para cualquier  $u \in \mathbb{R}^n$  podemos escribir,

$$u^T L u = \sum_{i,j=1}^n w_{i,j} (u_i - u_j)^2 = \sum_{t=1}^k \sum_{i,j \in A_t} w_{i,j} (u_i - u_j)^2.$$

Si  $u$  es un vector propio para el valor 0, entonces  $u^T L u = 0$ . Pero como  $u^T L u$  es una suma de valores no negativos, todos los términos valen 0. Entonces, tenemos

$$\forall t = 1, \dots, k, \quad \forall i, j \in A_t, \quad u_i = u_j.$$

Resulta que  $u$  es constante sobre cada componente  $A_t$ . Entonces  $u$  es de la forma  $u = a_1 \mathbb{1}_{A_1} + \dots + a_k \mathbb{1}_{A_k}$ .

### 3.3.5. El Laplaciano normalizado

Se definen dos otras nociones de Laplaciano que son

$$L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$$

$$L_{rw} = D^{-1}L = I - D^{-1}W.$$

#### Proposición

Tenemos los siguiente

1.  $L_{sym}$  y  $L_{rw}$  son simétricos y definido positivo.
2. Para un vector  $u \in \mathbb{R}^n$ ,

$$u^T L_{sym} u = \frac{1}{2} \sum_{i,j=1}^n w_{i,j} \left( \frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}} \right)^2$$

3. 0 es valor propio de  $L_{sym}$  y  $L_{rw}$  y el vector  $D^{-1/2}\mathbf{1}$  es vector propio asociado (para  $L_{sym}$ ), el  $\mathbf{1}$  es vector propio asociado (para  $L_{rw}$ )

### 3.3.5. El Laplaciano normalizado

Además de lo anterior tenemos algo ligeramente más general para los valores propios esas matrices:

$$\begin{aligned}\lambda \text{ es v.p. de } L_{rw} \text{ con v.p. } u &\Leftrightarrow \lambda \text{ es v.p. de } L_{sym} \text{ con v.p. } D^{1/2}u \\ &\Leftrightarrow Lu = \lambda Du \text{ (pb de v. p. generalizado)}\end{aligned}$$

La grande ventaja de esas definiciones alternativas proviene de la ponderación de los  $u_i$  por el grado  $d_i$ .

Es un problema más robusto! (Pensar en el caso de outliers)

### 3.3.6. El algoritmo

**Input:** La matriz  $S$ . El número de clusters  $k$ .

1. Construir el grafo de similitudes.
2. Calcular el Laplaciano  $L$ .
3. Calcular los primeros  $k$  vectores propios  $u_1, \dots, u_k$  de  $L$ .
4. Definir  $U = (u_1 | \dots | u_k) \in \mathcal{M}_{n,k}(\mathbb{R})$ .
5. Definir los vectores  $(y_i)_{i \leq n}$  de  $\mathbb{R}^k$  como la  $i$ -ésima fila de  $U$ .
6. Hacer  $k$  clusters  $C_1, \dots, C_k$  con los datos  $(y_i)_{i \leq n}$  por el medio de  $k$ -medias.

**Output:** Los clusters  $A_1, \dots, A_k$  tal que  $A_i = \{j | y_j \in C_i\}$ .

## 3.Agrupamiento de datos (Clustering)

3.4.Curso del 9 de noviembre

### 3.4.1. Interpretación del algoritmo

Para una descomposición  $A_1, \dots, A_k$ , tenemos varias maneras de medir la calidad del corte.

$$\text{cut}(A_1 \dots, A_k) = \frac{1}{2} \sum_{i=1}^k W(A_i, \bar{A}_i)$$

$$\text{RatioCut}(A_1 \dots, A_k) = \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{|A_i|}$$

$$\text{NCut}(A_1 \dots, A_k) = \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{\text{Vol}(A_i)}$$

El factor  $1/2$  es de pura normalización (para contar una sola vez cada arista).

En los dos últimos criterios, no se permite tener  $A_i$  demasiado “pequeños”.



### 3.4.2. Interpretación del algoritmo ( $k = 2$ )

En el caso  $k = 2$ . Queremos optimizar el corte en del tipo RatioCut. Es Formalmente,

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}).$$

Tomando el vector  $u = (u_1, \dots, u_n)^T$  tal que  $u_i = \sqrt{|\bar{A}|/|A|}$  si  $v_i \in A$  y  $u_i = -\sqrt{|A|/|\bar{A}|}$  si  $v_i \notin A$ , vemos que

$$\begin{aligned} u^T L u &= \frac{1}{2} \sum_{i,j=1}^n w_{i,j} (u_i - u_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \bar{A}} w_{i,j} \left( \sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \bar{A}, j \in A} w_{i,j} \left( -\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \end{aligned}$$

### 3.4.2. Interpretación del algoritmo ( $k = 2$ )

$$\begin{aligned}u^T L u &= \text{cut}(A, \bar{A}) \left( \frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\&= \text{cut}(A, \bar{A}) \left( \frac{|\bar{A}| + |A|}{|A|} + \frac{|\bar{A}| + |A|}{|\bar{A}|} \right) \\&= |V| \text{RatioCut}(A, \bar{A})\end{aligned}$$

Adicionalmente, tenemos

$$\sum_{i=1}^n u_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = \sqrt{|\bar{A}||A|} - \sqrt{|\bar{A}||A|} = 0$$

El vector  $u$  es ortogonal al vector  $\mathbf{1}$ .

### 3.4.2. Interpretación del algoritmo ( $k = 2$ )

También tenemos

$$\|u\|^2 = \sum_{i \in A} \frac{|\bar{A}|}{|A|} + \sum_{i \in \bar{A}} \frac{|A|}{|\bar{A}|} = |\bar{A}| + |A| = n$$

Podemos reescribir el problema de minimización de la manera siguiente

$$\begin{aligned} & \text{Minimizar} && \text{RatioCut}(A, \bar{A}) \\ & \textbf{Subject to} && A \subset V \\ & && \text{Minimizar} \quad u^T L u \\ \Leftrightarrow & \textbf{Subject to} && u_i \in \left\{ \sqrt{\frac{|\bar{A}|}{|A|}}, -\sqrt{\frac{|A|}{|\bar{A}|}} \right\}, \\ & && \|u\| = \sqrt{n}, u \perp \mathbf{1} \end{aligned}$$

Un problema más fácil es resolver

$$\begin{aligned} & \text{Minimizar} && u^T L u \\ & \textbf{Subject to} && \|u\| = \sqrt{n}, u \perp \mathbf{1} \end{aligned}$$

## 3.4.2. Interpretación del algoritmo ( $k = 2$ )

El problema de minimización

$$\begin{array}{ll} \text{Minimizar} & u^T L u \\ \text{Subject to} & \|u\| = \sqrt{n}, u \perp \mathbf{1} \end{array}$$

se resuelve teóricamente con la teoría espectral.

- ▶ La solución está dada por un vector propio asociado al segundo valor propio más chico de  $L$ . O equivalentemente por el primer valor propio no nulo.
- ▶ Falta transformar el vector  $u$  obtenido en un vector que toma los dos valores permitidas:

$$u_i \in \left\{ \sqrt{|\bar{A}|/|A|}, -\sqrt{|A|/|\bar{A}|} \right\}$$

**Idea :** Si  $\text{sgn}(u_i) = +$  entonces  $u_i \leftarrow \sqrt{|\bar{A}|/|A|}$  si no

$$u_i \leftarrow -\sqrt{|A|/|\bar{A}|}$$

### 3.4.3. Interpretación del algoritmo ( $k > 2$ )

En el caso  $k > 2$ , la partición del espacio no se hace tan fácilmente.

- ▶ Consideramos los vectores  $h_j = (h_{j,i})_i$  tal que

$$h_{j,i} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } v_i \in A_j \\ 0 & \text{si no} \end{cases} \quad \text{con } i = 1, \dots, n \quad j = 1, \dots, k$$

- ▶ Definimos la matriz  $H \in \mathcal{M}_{n,k}(\mathbb{R})$  poniendo en columna los vectores  $h_j$ . Tenemos  $H^T H = I_k$ .
- ▶ Entonces,

$$(H^T L H)_{jj} = h_j^T L h_j = \frac{\text{cut}(A_j, \bar{A}_j)}{|A_j|} = \frac{1}{2} \frac{W(A_j, \bar{A}_j)}{|A_j|}.$$

Lo que da

$$\text{Tr}(H^T L H) = \text{RatioCut}(A_1, \dots, A_k)$$

### 3.4.3. Interpretación del algoritmo ( $k > 2$ )

De la misma manera que en el caso  $k = 2$ , podemos relajar el problema en

$$\begin{array}{ll} \text{Minimizar} & \text{Tr}(H^T L H) \\ \text{Subject to} & H^T H = I_k \end{array}$$

- ▶ De nuevo la solución es explícita y dada por la matriz de los  $k$  primeros vectores propios.
- ▶ Luego tenemos que modificar los vectores  $h_j$  de tal manera que los  $h_{j,i}$  tomen los  $k$  valores posibles  $1/\sqrt{|A_j|}$ .

**Es una tarea de clustering!**

- ▶ En la etapa de clustering de los  $(h_{j,i})_j$  (vectores filas) se usa un método de clustering ingenuo (por ejemplo  $k$ -means).

### 3.4.4. Formas alternativas del algoritmo

Podemos hacer una versión que usa el Laplaciano renormalizado.

**Input:** La matriz  $S$ . El número de clusters  $k$ .

1. Construir el grafo de similitudes.
2. Calcular el Laplaciano  $L$ .
3. **Calcular los primeros  $k$  vectores propios (generalizados)  $u_1, \dots, u_k$  correspondiendo al problema  $Lu = \lambda Du$ .**
4. Definir  $U = (u_1 | \dots | u_k) \in \mathcal{M}_{n,k}(\mathbb{R})$ .
5. Definir los vectores  $(y_i)_{i \leq n}$  de  $\mathbb{R}^k$  como la  $i$ -ésima fila de  $U$ .
6. Hacer  $k$  clusters  $C_1, \dots, C_k$  con los datos  $(y_i)_{i \leq n}$  por el medio de  $k$ -medias.

**Output:** Los clusters  $A_1, \dots, A_k$  tal que  $A_i = \{j | y_j \in C_i\}$ .

### 3.4.5. Interpretación probabilista: Random Walk sobre $G$

Una interpretación natural de la disimilaridad  $NCut$  viene de las caminatas aleatorias sobre el grafo  $G$ .

- ▶ Estando en un punto  $v_i$ , definimos la probabilidad de transición a  $v_j$  como

$$p_{i,j} = \frac{w_{i,j}}{d_i}$$

La matriz de transición formada  $P = (p_{i,j})_{i,j=1,\dots,n}$  está dada por

$$P = D^{-1}W$$

- ▶ La teoría de cadenas de Markov dice que existe una única medida estable  $\pi$  cuando el grafo es conectado y no-bipartito,

$$\pi_i = \frac{d_i}{\text{Vol}(V)}.$$



### 3.4.5. Interpretación probabilista: Random Walk sobre $G$

- ▶ Supongamos que el grafo es conectado y no-bipartito. Y denotamos  $X_0$  una variable a valor en  $V$  con distribución  $\pi$ .
- ▶ Denotamos  $(X_t)_{t \geq 1}$  la caminata aleatoria con estado inicial  $X_0$ .
- ▶ Para  $A, B \subset V$  disjuntos, denotamos  $\mathbb{P}(B|A) = \mathbb{P}(X \in B | X_0 \in A)$

#### Proposición

*En esas condiciones,*

$$NCut(\bar{A}, A) = \mathbb{P}(\bar{A}|A) + \mathbb{P}(A|\bar{A})$$

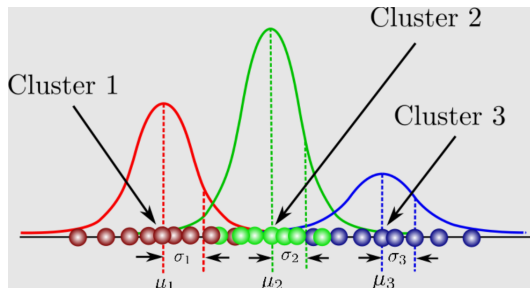
**Ejercicio:** Probar esa igualdad. (Pista: Calcular primero  $\mathbb{P}(X_0 \in A, X_1 \in B)$ )

## 3.Agrupamiento de datos (Clustering)

3.5.Curso del 14 de noviembre

## 3.5.1. Clustering con estimación de densidad

Nos interesamos a la clase de algoritmos que estiman una densidad en una etapa.



El caso popular es pensar que los datos son mezclas de variables gaussianas.

## 3.5.2. Algoritmo EM: Principio

- ▶ El algoritmo EM (por Expectation-Maximisation) está diseñado originalmente para poder encontrar iterativamente un máximo de verosimilitud.
- ▶ Consideramos una muestra  $X_1, \dots, X_n \in \mathbb{R}^d$  de densidad común  $f(\cdot|\theta_0)$  donde  $\theta_0$  es un (vector de) parámetro(s).
- ▶ Para encontrar el parámetro  $\theta_0$  se define la log-verosimilitud

$$L_x(\theta) = \sum_{i=1}^n \log f(X_i|\theta)$$

- ▶ En general la maximización de  $L$  puede ser muy compleja.

## 3.5.2. Algoritmo EM:Principio

**Ejemplo:** Si consideramos variables que son mezclas de 2 gaussianas en dimensión 3.

$$X_i = \lambda_i Y_i + (1 - \lambda_i) Z_i$$

donde

$$Y_i \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$Z_i \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

$$\lambda_i \sim \mathcal{B}(\pi) \quad \text{con } \pi \in (0, 1)$$

y si  $\phi_{\mu, \sigma}(x)$  representa la densidad de una variable normal con parámetros  $(\mu, \sigma^2)$ , tenemos

$$f(x, \theta) = \pi \phi_{\mu_1, \sigma_1}(x) + (1 - \pi) \phi_{\mu_2, \sigma_2}(x)$$

y  $\theta = (\pi, \mu_1, \sigma_1, \mu_2, \sigma_2)$  (9 parámetros reales!).

## 3.5.2. Algoritmo EM: Principio

**Idea:** Se inventa una variable (escondida) que auxilia para la maximización.

- ▶ Imaginamos  $(Z_1, \dots, Z_n)$  variables desconocidas de las cuales imaginamos que las  $X_i$  dependen... (Ideas de cadenas de Markov escondida)
- ▶ Calculando la verosimilitud

$$\begin{aligned}L_{x,z}(\theta) &= \sum_{i=1}^n \log f(X_i, Z_i | \theta) = \sum_{i=1}^n \log f(Z_i | X_i, \theta) + \log f(X_i | \theta) \\ &= \sum_{i=1}^n \log f(Z_i | X_i, \theta) + L_x(\theta)\end{aligned}$$

- ▶ La diferencia entre las dos log-verosimilitud está contenida en la función  $f(z|x, \theta)$  (la distribución de  $Z$  dado  $X$  y  $\theta$ )

## 3.5.2. Algoritmo EM: Principio

- ▶ Fijamos el parámetro  $\theta^{(c)}$  (como si fuera conocido).
- ▶ Entonces tomando la esperanza a través del operador  $\mathbb{E}[\cdot | X, \theta^{(c)}]$  (significa esperanza en  $Z$  con respecto a  $f(z|x, \theta^{(c)})$ ) en la igualdad anterior,

$$\begin{aligned}\mathbb{E} \left[ L_x(\theta) | X, \theta^{(c)} \right] &= \mathbb{E} \left[ L_{x,z}(\theta) | X, \theta^{(c)} \right] \\ &= \mathbb{E} \left[ \sum_{i=1}^n \log f(Z_i | X_i, \theta) \middle| X, \theta^{(c)} \right]\end{aligned}$$

Que volvemos a escribir

$$L_x(\theta) = Q(\theta, \theta^{(c)}) - H(\theta, \theta^{(c)})$$

donde  $Q(\theta, \theta^{(c)}) = \mathbb{E} [L_{x,z}(\theta) | X, \theta^{(c)}]$  y  
 $H(\theta, \theta^{(c)}) = \mathbb{E} [\sum_{i=1}^n \log f(Z_i | X_i, \theta) | X, \theta^{(c)}]$ .

### 3.5.3. Algoritmo EM: La magia

► Magia: Solo vamos a considerar  $Q(\theta, \theta^{(c)})$  en el algoritmo.

1. Inicialización:  $\theta^{(0)}$  se elige al azar,  $t = 0$

2. Hasta que el algoritmo converja:

2.1 Etapa E: Evaluación de la esperanza:

$$Q(\theta, \theta^{(t)}) = \mathbb{E} \left[ L_{x,z}(\theta) | X, \theta^{(t)} \right]$$

2.2 Etapa M: Maximización:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta, \theta^{(t)})$$

#### Proposición

*El algoritmo EM converge a un máximo local de  $L_x(\theta)$ .*

En la practica se corre muchas veces el algoritmo para acercarse del máximo global.



### 3.5.3. Algoritmo EM: La magia

**Prueba:** Haciendo la misma descomposición que antes para  $\theta = \theta^{(t)}$ , tenemos

$$L_x(\theta^{(t)}) = Q(\theta^{(t)}, \theta^{(t)}) - H(\theta^{(t)}, \theta^{(t)})$$

Entonces,

$$\begin{aligned} L_x(\theta) - L_x(\theta^{(t)}) &= Q(\theta, \theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)}) \\ &\quad + \left( H(\theta^{(t)}, \theta^{(t)}) - H(\theta, \theta^{(t)}) \right) \end{aligned}$$

Pero  $H(\theta^{(t)}, \theta^{(t)}) \geq H(\theta, \theta^{(t)})$  (desigualdad de Gibbs) en efecto, si denotamos  $p(z) = f(z|x, \theta^{(t)})$  y  $q(z) = f(z|x, \theta)$ , tenemos

$$H(\theta^{(t)}, \theta^{(t)}) - H(\theta, \theta^{(t)}) = \sum_{i=1}^n \int \log(p(z_i))p(z_i) - \log(q(z_i))p(z_i) dz_i$$

### 3.5.3. Algoritmo EM: La magia

Para cada  $i$ ,

$$\begin{aligned}\int \log(p(z_i))p(z_i) - \log(q(z_i))p(z_i)dz_i &= - \int \log \frac{q(z_i)}{p(z_i)}p(z_i)dz_i \\ &\geq - \int \left( \frac{q(z_i)}{p(z_i)} - 1 \right) p(z_i)dz_i \\ &= \int p(z_i)dz_i - \int q(z_i)dz_i \\ &= 1 - 1 = 0\end{aligned}$$

Finalmente tenemos que

$$L_x(\theta) - L_x(\theta^{(t)}) \geq Q(\theta, \theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)})$$

Entonces  $\theta^{(t)}$  converge a un máximo local de  $L_x$ .

### 3.5.4. EM para mezclas

Retomamos el modelo de mezclas.

- ▶ Supongamos que los datos  $X_1, \dots, X_n$  son mezclas de  $k$  densidades  $g(\cdot|\theta_1), \dots, g(\cdot|\theta_k)$ . Con probabilidad  $\pi_j$  una variable  $X_i$  proviene de un grupo  $G_j$  (que corresponde a tener la densidad  $g(\cdot|\theta_j)$ .)
- ▶ Entonces

$$f(x|\Phi) = \sum_{i=1}^k \pi_i g(x|\theta_i)$$

donde el parámetro  $\Phi = (\pi_1, \dots, \pi_k, \theta_1, \dots, \theta_k)$ .

- ▶ Se “inventa” la variable  $Z_i = (z_{i,1}, \dots, z_{i,k})$  tal que  $z_{i,j} = \mathbb{1}_{X_i \in G_j}$
- ▶ Entonces,

$$L_{x,z}(\Phi) = \sum_{i=1}^n \sum_{j=1}^k z_{i,j} \log(\pi_j g(X_i|\theta_j))$$

### 3.5.4. EM para mezclas

- ▶ Como consecuencia, tenemos

$$Q(\Phi, \Phi^{(c)}) = \sum_{i=1}^n \sum_{j=1}^k \mathbb{E} \left[ z_{i,j} | X_i, \Phi^{(c)} \right] \log(\pi_j g(X_i | \theta_j))$$

y solo se tiene que estimar

$$t_{i,j} = \mathbb{E} \left[ z_{i,j} | X_i, \Phi^{(c)} \right].$$

Esa cantidad se estima con un estimador de Bayes

$$\hat{t}_{i,j} = \frac{\pi_j^{(c)} g(X_i | \theta_j^{(c)})}{\sum_{\ell=1}^k \pi_{\ell}^{(c)} g(X_i | \theta_{\ell}^{(c)})}$$

## 3.5.4. EM para mezclas

El algoritmo se puede escribir explícitamente

1. Inicialización:  $\Phi^{(0)}$  se elige al azar,  $t = 0$
2. Hasta que el algoritmo converja:
  - 2.1 Etapa E: Evaluación de la esperanza:

$$\hat{Q}(\Phi, \Phi^{(t)}) \quad \text{con} \quad \hat{t}_{i,j}$$

- 2.2 Etapa M: Maximización:

$$\Phi^{(t+1)} = \underset{\Phi}{\operatorname{argmax}} \hat{Q}(\Phi, \Phi^{(t)})$$

3. Output: El dato  $i$  se clasifica en el grupo  $j$  tal que

$$j = \underset{j}{\operatorname{argmax}} \hat{t}_{i,j}$$

## 4.Nociones de redes neuronales

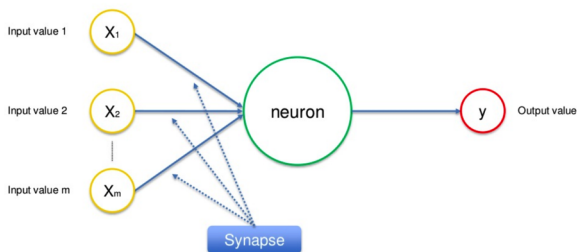
### 4.1.Curso del 16 de noviembre

## 4.1.1. El neurona

Es un objeto de programación que recibe una lista de valores  $x_1, \dots, x_n$  y contesta un valor de salida.

### The Neuron

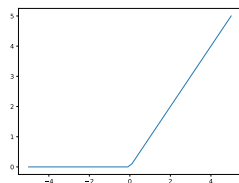
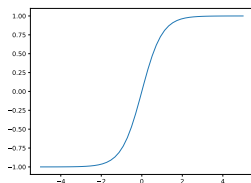
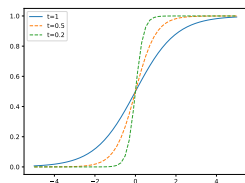
Clip icon



En aprendizaje maquina la salida tiene la forma  $y = \sigma(w^T x + v)$ , donde  $\sigma$  es una función llamada de **activación**.

## 4.1.1. El neurona

Algunas funciones de activación famosas.



(a) Sigmoid :  $\sigma(x) = \frac{1}{1 + \exp(-x/t)}$

(b) Tanh:  $\sigma(x) = \tanh(x)$

(c) ReLU :  $\sigma(x) = \max(0, x)$

La función ReLU (Rectifier Linear Unit) es la función de activación la mas popular en aprendizaje maquina aplicado.



## 4.1.2. Perceptron (Rosenblatt - 1958)

El perceptron es un algoritmo (viejo!) que permite hacer una tarea de estimación de  $w$  y  $v$  cuando

$$\sigma(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

y  $f(x) = \sigma(w^T x + v)$  para  $w \in \mathbb{R}^d$  y  $v \in \mathbb{R}$ .

Los datos son variables  $(X_1, Y_1), \dots, (X_n, Y_n) \in \mathbb{R}^d \times \{-1, 1\}$ .

La idea es encontrar  $w$  y  $v$  tal que para cada  $i$ ,  $Y_i = f(X_i)$ .

## 4.1.2. Perceptron (Rosenblatt - 1958)

**Input:** La muestra  $((X_i, Y_i))_{i \leq n}$ .

1. Inicialización :  $(w, v)$  al azar.
2. Mientras Mis ejecutar:  
**Si** existe  $i$  tq  $f(x_i) \neq Y_i$  **entonces**  
 $w = w + Y_i X_i$   
 $v = v + Y_i$   
**Si no,**  
Mis = False

**Output:**  $w, v$ .

- ▶ Si existe un hiperplano separador, el algoritmo encuentra la solución en tiempo finito.
- ▶ Si no, el algoritmo no termina!
- ▶ Con las herramientas de este curso, podemos hacer cosas mucho mejor!

## 4.1.2. Perceptron (Rosenblatt - 1958)

La teoría dice que si iniciamos con  $(w_0, v_0)$ , el algoritmo tomará a lo más

$$(\|w_*\|^2 + v_*^2) \frac{\|w_* - w_0\|^2 + (v_* - v_0)^2 + 2 \max_i \|X_i\|^2 + 2}{L^2}$$

donde

$$L = \min_i |w_*^T X_i + v_*|$$

**Problema:** Para funciones tan sencillas como XOR, el algoritmo no termina!

### 4.1.3. Shallow Neural Network

La función es de la forma

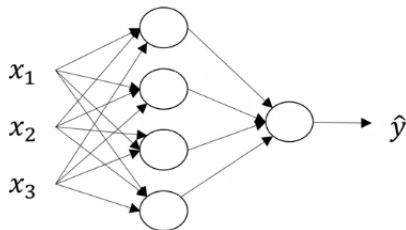
$$f(x) = \sum_{k=1}^K \alpha_k \sigma(w_k^T x + v_k)$$

Es una combinación lineal de neuronas (o perceptrons). Entonces la clase de funciones que podemos construir son

$$\mathcal{F}_{K,\sigma} = \left\{ x \mapsto \sum_{k=1}^K \alpha_k \sigma(w_k^T x + v_k) : w_k \in \mathbb{R}^d, v_k, \alpha_k \in \mathbb{R} \right\}$$

## 4.1.3. Shallow Neural Network

- ▶ El número de terminos en la suma = número de neuronas.
- ▶ Vocabulario:
  - ▶  $w_k$  : inner weights
  - ▶  $v_k$  : inner biases
  - ▶  $\alpha_k$  : outer weights
- ▶ La función de activación  $\sigma$  y el número de neuronas  $K$  son fijos y conocidos.
- ▶ Se tiene que estimar los  $w_k, v_k, \alpha_k$ .



## 4.1.4. Arquitectura general (multicapas)

### Arquitectura Red Neuronal $(L, k)$ :

- ▶ Un entero  $L$  que denota el número de capas escondidas (hidden layers).
- ▶ Un vector  $k = (k_1, \dots, k_L) \in \mathbb{N}^L$ .

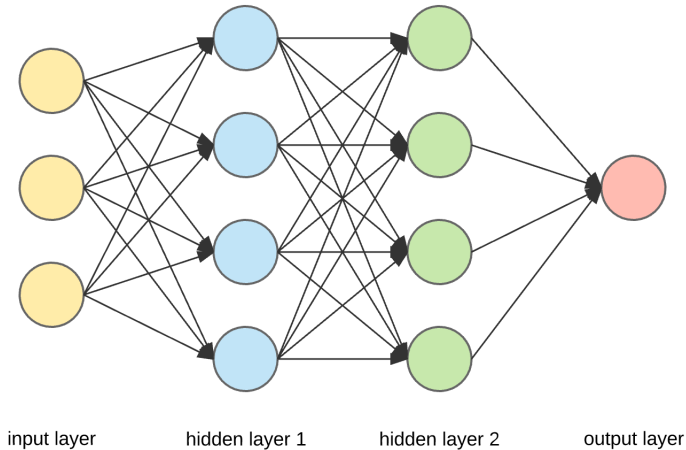
### Red Neuronal con Arquitectura de Red Neuronal $(L, k)$ :

$$f : x \mapsto W_{L+1} \sigma_{v_L} W_L \sigma_{v_{L-1}} \dots W_2 \sigma_{v_1} W_1 x$$

### Parámetros de la Red Neuronal:

- ▶ Matrices  $W_i$  de tamaño  $k_i \times k_{i-1}$ . (Sup:  $k_0 = d$ ,  $k_{L+1} = 1$ )
- ▶ Sesgos  $v_i \in \mathbb{R}^{k_i}$ .

## 4.1.4. Arquitectura general (multicapas)



## 4.1.4. Arquitectura general (multicapas)

- ▶ Para distinguir con otras estructuras, esas redes neuronales se llaman **Multilayer Feedforward Neural Networks**.
- ▶ La arquitectura está **dada**.

Casos particulares:

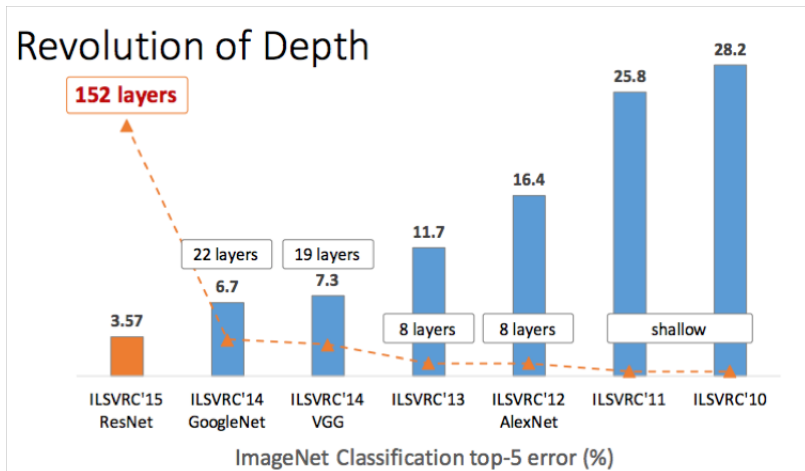
- ▶  $L = 1$ , **Shallow neural network** (poco profundos, llanos).
- ▶  $L \geq 2$ , **Deep neural network**.
- ▶ Si un parámetro  $s \in \mathbb{N}$  acota el número de pesos en la red entonces se habla de **Sparse neural networks**.
- ▶ Sin restricción sobre la conectividad, se llaman **Fully connected neural networks**.

Otras estructuras populares:

- ▶ **Convolutional neural networks**
- ▶ **Transformers, Autoencoders**



## 4.1.4. Arquitectura general (multicapas)



## 4.1.5. Aproximación univariada

### Retorno a las Redes Neuronales poco profundas

Las funciones son

$$\mathcal{F}_{K,\sigma} = \left\{ x \mapsto \sum_{k=1}^K \alpha_k \sigma(w_k^T x + v_k) : w_k \in \mathbb{R}^d, v_k, \alpha_k \in \mathbb{R} \right\}$$

- ▶ Para determinar una función hay que determinar  $K(d+2)$  parámetros.
- ▶ Los espacios son crecientes en  $K$  en el sentido

$$\mathcal{F}_{K,\sigma} \subset \mathcal{F}_{K',\sigma} \quad \text{cuando } K \leq K'.$$

## 4.1.5. Aproximación univariada

Hay un resultado importante para la validación de las redes neuronales:

### Proposición

Sea  $g : [0, 1] \rightarrow \mathbb{R}$  una función  $\rho$ -lipschitz. Entonces, para cualquier  $\epsilon > 0$ , existe una red neuronal  $f$  poco profunda ( $L = 1$ ) tal que  $f \in \mathcal{F}_{K,\sigma}$  con  $\sigma(x) = \mathbb{1}_{x \geq 0}$  y  $K = \lceil \rho/\epsilon \rceil$  tal que

$$\sup_{x \in [0,1]} |f(x) - g(x)| \leq \epsilon.$$

## 4.1.5. Aproximación univariada

### Prueba:

Podemos discretizar el intervalo  $[0, 1]$  de manera regular. Sean  $K = \lceil \rho/\epsilon \rceil$ ,  $b_i = i\epsilon/\rho$  para  $i \in \{0, \dots, K-1\}$ ,  $a_0 = g(0)$  y  $a_i = g(b_i) - g(b_{i-1})$ . Definimos la función

$$f(x) = \sum_{i=0}^{K-1} a_i \mathbb{1}_{x \geq b_i}.$$

Es una función "red neuronal". Para  $x \in [0, 1]$  sea  $k$  el índice máximo tal que

$$x \geq b_k$$

y entonces  $f$  es constante sobre el intervalo  $[b_k, x]$ . Entonces,

$$\begin{aligned} |g(x) - f(x)| &\leq |g(x) - g(b_k)| + |g(b_k) - f(b_k)| + |f(b_k) - f(x)| \\ &\leq \rho|x - b_k| + |g(b_k) - \sum_{i=0}^k a_i| \leq \rho \frac{\epsilon}{\rho} = \epsilon \end{aligned}$$

## 4.Nociones de redes neuronales

4.2.Curso del 21 de noviembre

## 4.2.1. Aproximación universal

De hecho, el resultado es más general. Hay una propiedad de aproximación universal.

### Teorema

*Las redes neuronales poco profundas definidas con una función de activación  $\sigma \in C_\infty$  tal que  $\sigma$  **no** es un polinomio tiene la propiedad de aproximación universal:*

*Para cada  $\epsilon > 0$  y cada función  $g$  continua sobre  $[0, 1]^d$ , existe un  $K = K(g, \epsilon)$ , tal que*

$$\inf_{f \in \mathcal{F}_{K, \sigma}} \|f - g\|_\infty \leq \epsilon.$$

## 4.2.1. Aproximación universal

**Prueba:** Tratamos el caso  $d = 1$ .

Definimos  $\Delta_h^1 \sigma(t) = (\sigma(t + xh) - \sigma(t))/h$  y de manera recursiva  $\Delta_h^k \sigma(t) = \Delta_h^1(\Delta_h^{k-1} \sigma)(t)$ .

Por definición de las derivadas de orden  $k$ , tenemos,  $\forall t \in [0, 1]$ ,

$$\left| \Delta_h^k \sigma(t) - x^k \sigma^{(k)}(t) \right| \rightarrow 0, \quad \text{cuando } |h| \rightarrow 0.$$

Como  $\sigma$  no es un polinomio, existe un valor  $t_k \in [0, 1]$  tq

$$\sigma^{(k)}(t_k) \neq 0$$

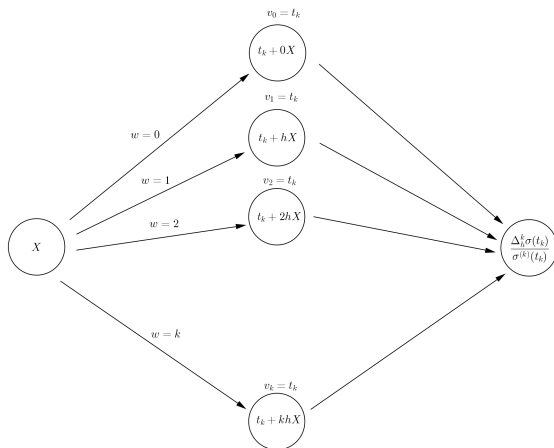
y entonces

$$\left| \frac{\Delta_h^k \sigma(t_k)}{\sigma^{(k)}(t_k)} - x^k \right| \rightarrow 0, \quad \text{cuando } |h| \rightarrow 0.$$

## 4.2.1. Aproximación universal

Pero, como función de  $x$  la función  $\Delta_h^k \sigma(t_k)$  es una combinación lineal de los  $\sigma(t), \sigma(t + hx), \sigma(t + 2hx), \dots, \sigma(t + khx)$ . Entonces, para cualquier  $h$ ,

$$f_h : x \mapsto \Delta_h^k \sigma(t_k) \in \mathcal{F}_{k+1, \sigma}$$





## 4.2.1. Aproximación universal

Entonces la función  $x \mapsto x^k$  se aproxima por las funciones  $f_h$ . La convergencia ocurre uniformemente en  $x$ . (Convergencia de funciones continuas en un compacto.)

Por el teorema de aproximación de Weierstrass, sabemos que cualquier función continua  $f$  se puede aproximar uniformemente por funciones polinomio

$$P_K(f) \xrightarrow{K \rightarrow \infty} f$$

Entonces tenemos

$$(P_K(f))_h \xrightarrow{h \rightarrow 0} P_K(f) \xrightarrow{K \rightarrow \infty} f$$

$(P_K(f))_h$  es una función de red neuronal poco profunda con  $K + 1$  neuronas.

## 4.2.1. Aproximación universal

### Comentarios:

- ▶ Vemos porque  $\sigma$  no puede ser un polinomio. En efecto, las combinaciones lineales de polinomios de grado  $r$  son polinomios de grado a lo más  $r$ .
- ▶ Más capas de neuronas ayuda en este caso! Ayuda a generar polinomios de grados crecientes!
- ▶ La construcción es explícita en función de las derivadas de las funciones.

## 4.2.2. Caso multivariado: Prueba

El resultado de la prueba es que

$$\overline{\bigcup_K \mathcal{F}_{K,\sigma}} = \mathcal{C}[0, 1]$$

Para el resultado de dimensión general, se usa que la clase de funciones Ridge

$$x \mapsto \sum_{j=1}^K f_j(w_j^T x)$$

son densas en la clase de funciones continuas, donde cada  $f_j : \mathbb{R} \rightarrow \mathbb{R}$  es una función continua. El caso unidimensional de la aproximación da que

$$f_j \approx \sum_{i=1}^{K'} \sigma(a_{ji} \cdot + b_{ji})$$

## 4.2.2. Caso multivariado: Prueba

Entonces para una función continua  $f : [0, 1]^d \rightarrow \mathbb{R}$ ,

$$f \approx \sum_{j=1}^K f_j(w_j^T \cdot) \approx \sum_{i=1}^{K'} \sum_{j=1}^K \sigma(a_{ji} w_j^T \cdot + b_{ji}).$$

La aproximación se hace con una red neuronal poco profunda con  $K \times K'$  neuronas.

## 4.2.3. Un teorema general de aproximación universal

En general hay un teorema importante para la aproximación de funciones:

### Teorema

Sea  $\mathcal{F}$  una clase de funciones que satisface las condiciones:

1. Cada función  $f \in \mathcal{F}$  es continua.
2. Para cada  $x$ ,  $\exists f \in \mathcal{F}$  tal que  $f(x) \neq 0$ .
3. Para cada  $x \neq x'$ ,  $\exists f \in \mathcal{F}$  tal que  $f(x) \neq f(x')$ . (Separación de los puntos)
4.  $\mathcal{F}$  es una algebra.

Entonces  $\mathcal{F}$  es una clase de aproximación universal.

## 4.2.3. Un teorema general de aproximación universal

Entonces tenemos fácilmente

### Proposición

*La clase*

$$\mathcal{F}_{\cos,d} = \{x \in \mathbb{R}^d \mapsto \sum_i \alpha_i \cos(w_i^T x)\}$$

*es una clase de aproximación universal.*

- ▶ Entonces una red neuronal poco profunda con  $\sigma = \cos$  es una clase de aproximación universal.

## 4.2.4. Estimación de los pesos

- ▶ La estimación de los pesos  $W_i$  es el reto mayor para la estimación de las redes neuronales (profundas o poco profundas).
- ▶ El problema de estimación es no-lineal (altamente no-lineal para redes profundas).
- ▶ Se basa sobre la técnica general de minimización de función de pérdida.

$$L(\theta) = \sum_{i=1}^n \ell(Y_i, f_{\theta}(X_i)).$$

- ▶ Una manera general de solucionar este problema es el famoso **descenso del gradiente**.
  1. Elegir un parámetro  $\theta_0 \in \mathbb{R}^d$  y un tasa de aprendizaje  $\eta > 0$ .
  2. Secuencialmente, mover

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t)$$

## 4.2.4. Estimación de los pesos

- ▶ El aprendizaje de una red neuronal requiere muchos datos y en general es (extremadamente) costoso de calcular

$$\nabla_{\theta_t} L(\theta) = \sum_{i=1}^n \nabla_{\theta_t} \ell(Y_i, f_{\theta}(X_i))$$

- ▶ Una alternativa es SGD (Stochastic gradient descent), descenso del gradiente aleatorio. Se elige un dato  $(X_{t(i)}, Y_{t(i)})$  al azar y se calcula

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \ell(Y_{t(i)}, f_{\theta}(X_{t(i)}))$$

donde  $t(i)$  es el índice aleatorio del dato seleccionado.

- ▶ Es computacionalmente mucho más ligero pero introduce mucho ruido.
- ▶ En la práctica se utiliza el compromiso:

$$\theta_{t+1} = \theta_t - \eta \sum_{i \in B_t} \nabla_{\theta_t} \ell(Y_i, f_{\theta}(X_i))$$

con  $B_t$  aleatorio y  $|B_t| \ll n$ .