VISIÓN POR COMPUTADOR

Reconocimiento de Objetos

Ingeniería de Sistemas y Automática

Universidad Miguel Hernández

Planteamiento General

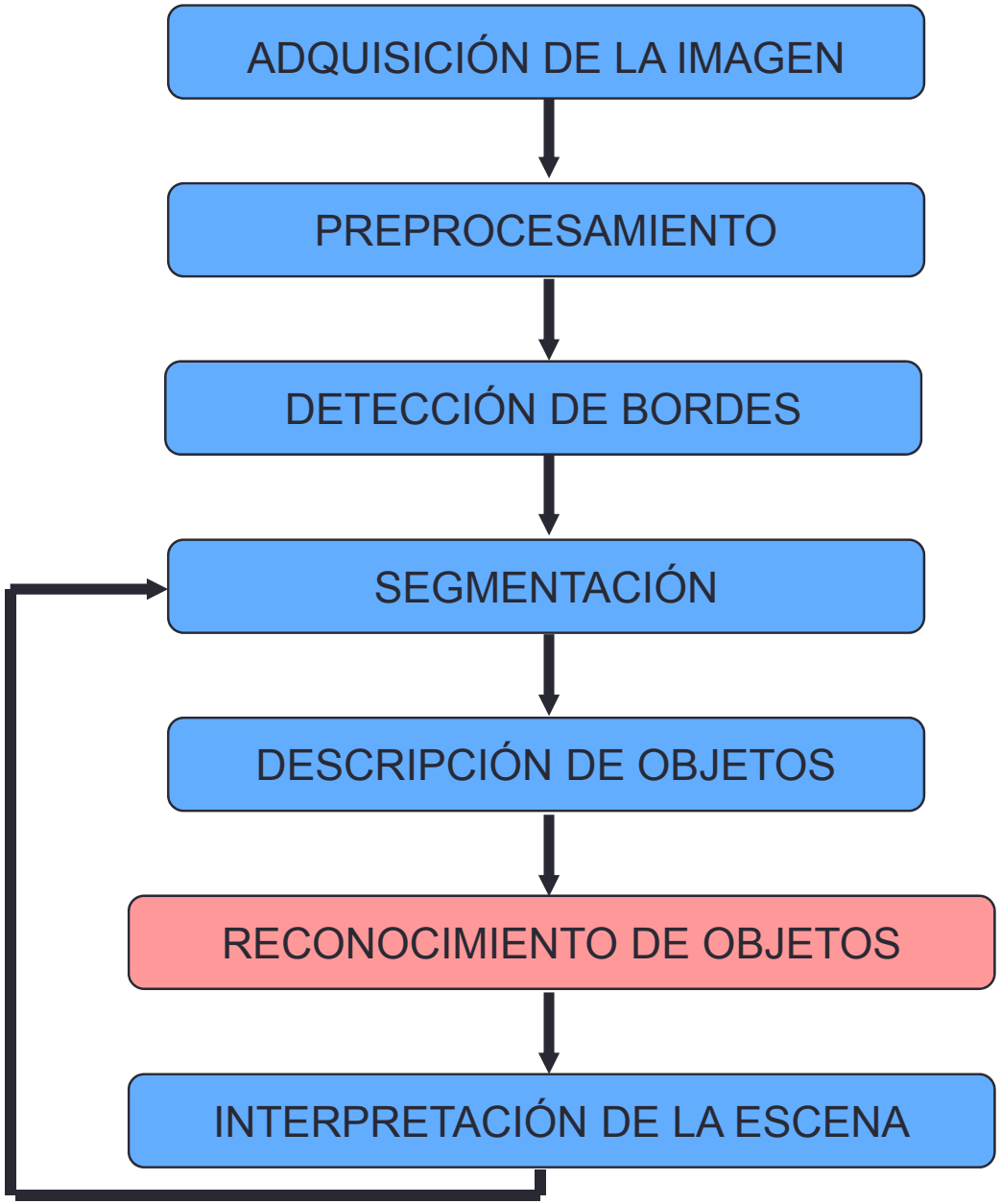
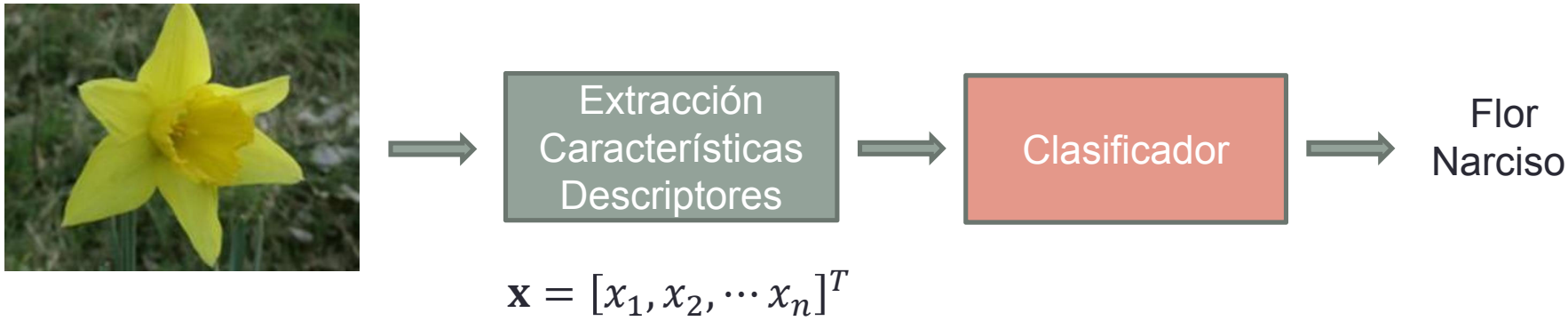


Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características. Validación
- Otros métodos de reconocimiento

Introducción

- Objetivo:
  - Dado un vector de características que proviene de un objeto de la escena nos proponemos identificarlo.
  - Lo podemos entender como un problema de clasificación de patrones.



Introducción

- Descripción
  - Se obtiene una descripción de los objetos mediante un ***vector de características.***
  - Se utilizan características invariantes ante:
    - Rotación, traslación, escalado.
    - Cambios de iluminación.
    - Oclusión.
    - Punto de vista…
  - Contamos con una serie de ***muestras*** que nos describen una clase de objetos.
- Reconocimiento ⇔ Clasificación
  - El enfoque del reconocimiento se puede entender como una clasificación.
  - Asociación de objetos con una representación conocida (PATRONES) a alguno de los grupos representativos (CLASES)
  - Supervisado / No supervisado

Introducción

- Ejemplo de Reconocimiento:

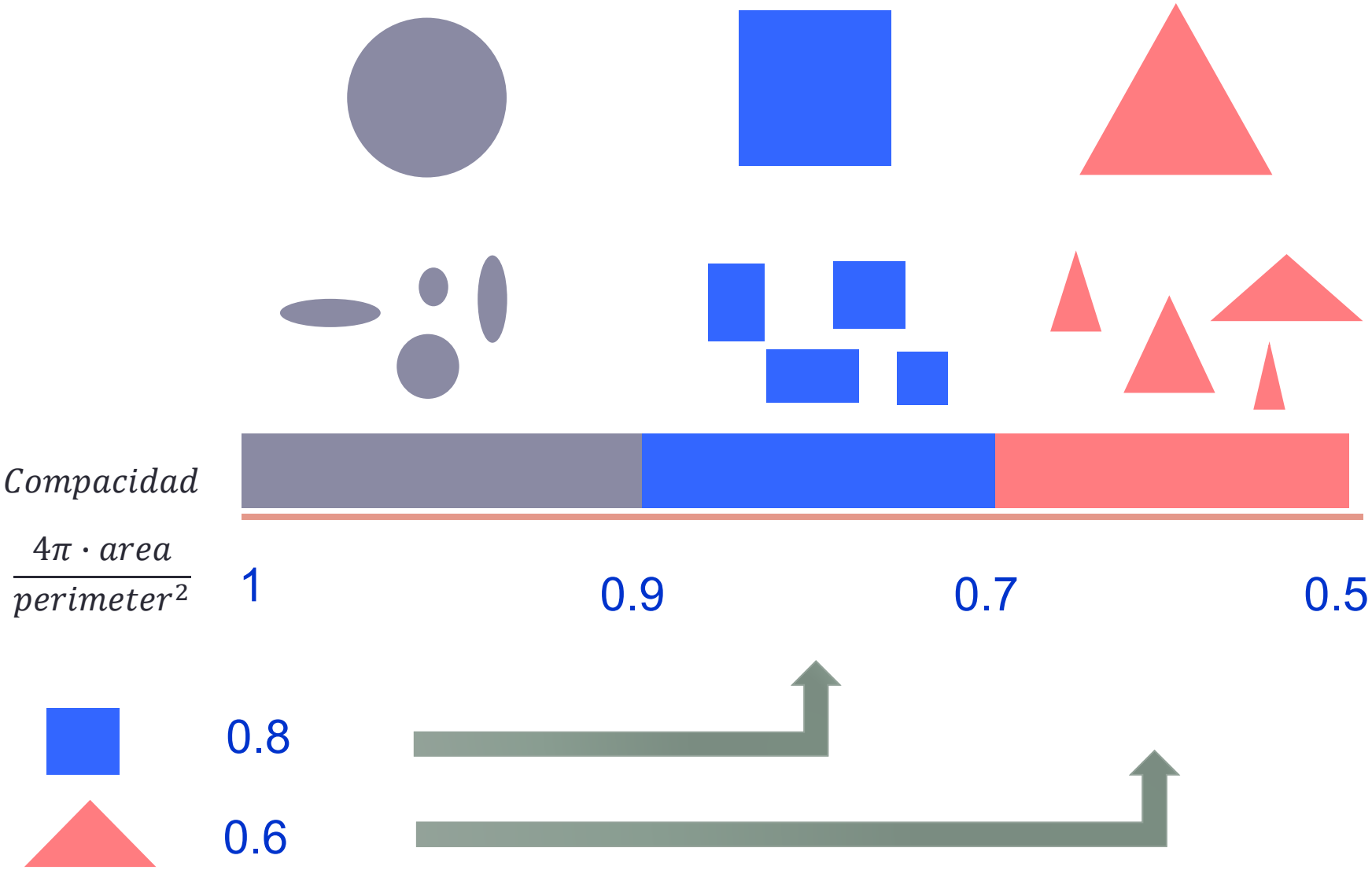
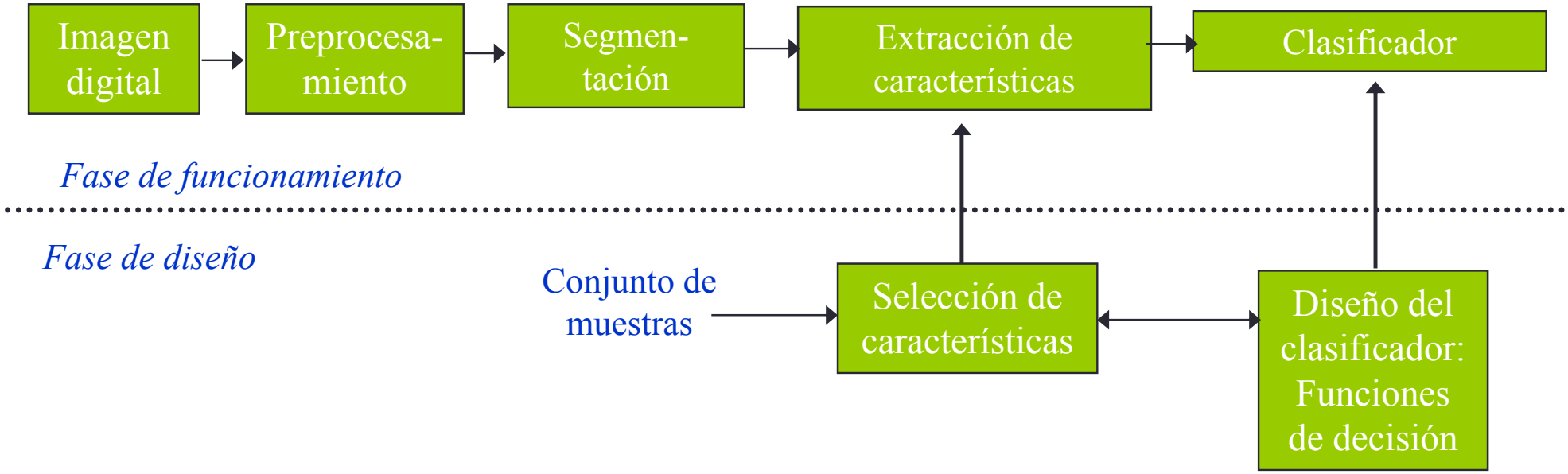


Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

Esquema general de un Sistema de Reconocimiento



- El diseño de un sistema de reconocimiento comprende:
  - Una fase de diseño (aprendizaje o entrenamiento del sistema de reconocimiento).
  - Una fase de funcionamiento ➔Se clasifican los objetos extraídos on-line.

Ciclo de Diseño de un Clasificador

- Etapas en el diseño:
  - Recogida de datos:
    - Recogida de una serie de características $\{x_1, x_2, x_3 \ldots\}$ para un conjunto de patrones (con clase conocida).
  - Elección de un conjunto de características:
    - Debemos elegir un conjunto de características más adecuado que nos permita separar mis muestras en una serie de clases.
    - Las características deberán tener valores similares para objetos pertenecientes a la misma clase, y diferentes para objetos pertenecientes a clases diferentes.
  - Cálculo de una función de decisión que me separe adecuadamente mis clases:
    - Dependiendo del número de características escogidas la complejidad de la decisión será mayor o menor.
    - Se plantea como un entrenamiento de mi sistema a partir de los datos de muestra disponibles.
  - Evaluación: Evaluación de los resultados de mi clasificador. Nº de aciertos y errores.

Ciclo de Diseño de un Clasificador

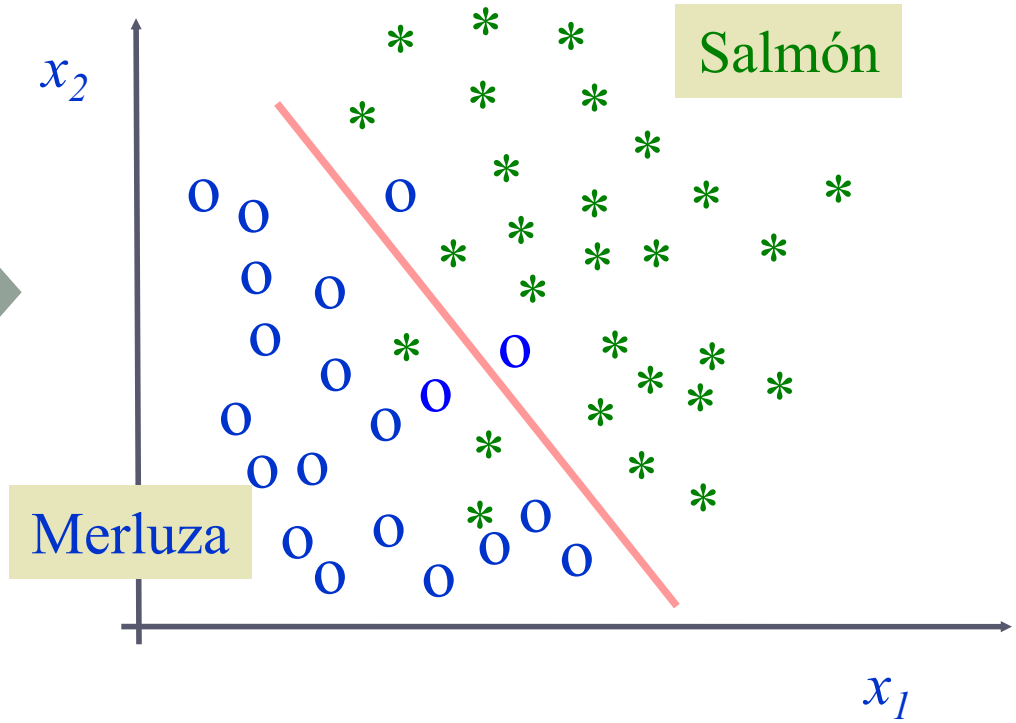
- Recogida de Muestras:
  - Recogemos una serie de datos a partir de unos ejemplos clasificados.
- Selección de Características:
  - Se nos plantea el problema de decidir qué conjunto de características de mi vector me separa mejor mis clases, ya que simplifica el diseño del clasificador.

- Práctica 4: Características
  - Área
  - Perímetro
  - Relación de compacidad
  - Momentos
  - … diferentes condiciones de iluminación
  - Diferentes proyecciones (vistas)
- ¿Cuáles elegimos?
  - Todas las que podamos →¿?
  - Pocas →¿?

Ciclo de Diseño de un Clasificador

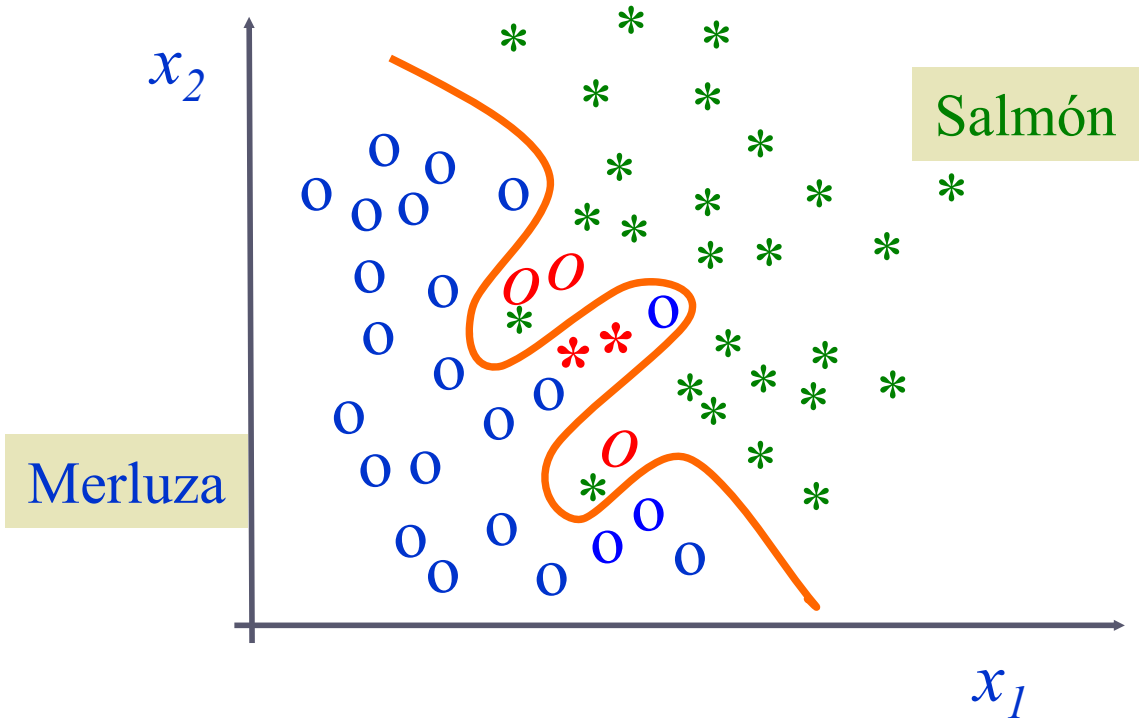
- Cálculo de la función de decisión. Entrenamiento:
  - A partir de una serie de datos, deseamos extraer una serie de reglas que nos separen de manera adecuada mis muestras.
  - Ejemplo de separación:

x_1: Brillo
x_2: Longitud



Ciclo de Diseño de un Clasificador

- Generalización:
  - Existe más de una manera de separar mis muestras. Mi sistema debe ser capaz de extraer una regla general a partir de la información.



  - La separación anterior es óptima para mi conjunto de muestras de entrenamiento, es decir, un caso particular.
  - Nuevas muestras no serán separadas correctamente usando esta función→No está generalizada para todos los casos. (Sobreajuste/Overfitting)
  - Buscamos una generalización a partir de una serie de muestras.

Métodos de Clasificación

- Métodos de Aprendizaje Inductivo.
  - Sin Modelo (*Lazy*):
    - Vecino más cercano (*K-NN, clustering)*
  - Construyen un modelo (*Eager*):
    - Teoría Bayesiana (conocemos las funciones de distribución de probabilidad)
    - Regresión Logística / SVM
    - Árboles de Decisión
    - Listas de Reglas
    - Redes Neuronales
- Métodos Basados en Apariencia
- Métodos Basados en Puntos Significativos

Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

Teoría Bayesiana de decisión

- Teoría Bayesiana de Decisión
  - Basamos nuestra decisión en el conocimiento de la probabilidad de que un determinado patrón pertenezca a una clase.
  - Objetivo: Asignar un patrón a una clase de manera que se minimice la probabilidad de error.
  - Cada patrón se representa mediante un vector de descripción, y cada clase por uno o varios patrones prototipo
  - El vector de descripción toma valores en el **Espacio de Características**, donde definiremos las clases que agrupan a patrones que pertenecen a la misma clase.

Teoría Bayesiana de decisión

- Vector de Características, propiedades: $\{x_1, x_2, x_3 \ldots x_n\}$
  - ***Discriminación***: Para las diferentes clases, las características deben tomar valores distintos
  - ***Fiabilidad***: Para las mismas clases, las características deben tomar valores similares
  - ***Independencia***: No debe haber relación entre las distintas características, ya que podrían reflejar la misma característica del objeto
  - ***Número***: Debe ser el menor número posible, ya que la complejidad aumenta considerablemente

- Las herramientas de la Teoría Bayesiana nos permitirán la selección óptima del vector de características => mínimo número de características con mayor capacidad discriminante

Teoría Bayesiana de decisión

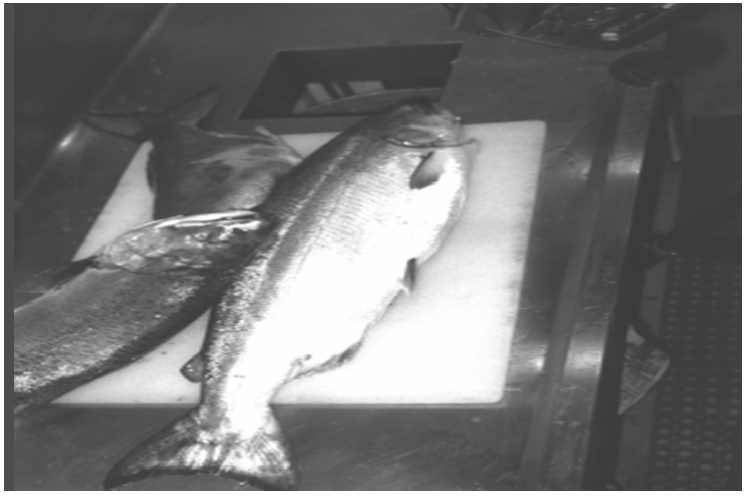
- Ejemplo: Clasificador de peces. Suponemos la existencia de dos clases.
  - Debemos decidir a qué clase ω pertenece cada objeto (pez) en base a una serie de medidas tomadas por el sistema de visión.
  - Describiremos la clase como:
    - Merluza: $\omega = \omega_1$
    - Salmón: $\omega = \omega_2$
  - Si la cantidad total de cada tipo es conocida, entonces podemos asumir cierto conocimiento *a priori*. Probabilidades a priori:

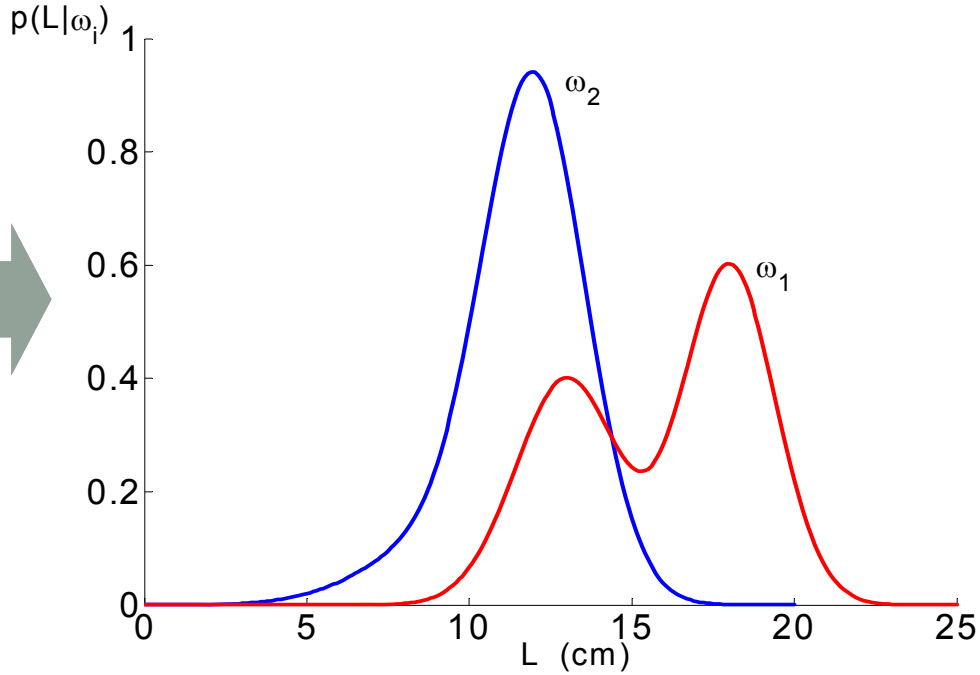
$$P(\omega_1) \qquad\qquad P(\omega_2)$$

  - Además, nuestro sistema de visión nos proporciona cierta información: P. e. la longitud $\mathbf{L}$ del pez. Suponemos que $\mathbf{L}$ es una variable aleatoria continua. Su distribución depende de la clase de pez.

Teoría Bayesiana de decisión

- Realizamos una medida $x=L$ con nuestro sistema de visión (longitud del objeto observado).
  - Consideramos que $\mathbf{L}$ es una variable aleatoria que se distribuye de manera diferente dependiendo de la clase.
  - $\boldsymbol{p}(L|\omega_i)$ es la función de densidad de probabilidad condicionada a la clase *i*. Describe la longitud de la población de ω_1 y ω_2





Teoría Bayesiana de decisión

- Utilizando la regla de Bayes podemos calcular:

$$P(\omega_i|x) = \frac{p(x|\omega_i) \cdot P(\omega_i)}{p(x)}$$

  - Siendo:
    - Probabilidad de x condicionado a la clase ω_i: $p(x|\omega_i)$
    - Probabilidad a priori de la clase ω_i: $P(\omega_i)$
    - Probabilidad total de ocurrencia de x (normalizador): $p(x)$

$$p(x) = \sum_{j=1}^{n} p(x|\omega_j)P(\omega_j)$$

- La regla de Bayes nos permite calcular $P(\omega_i|x)$:
  - Probabilidad de que el objeto pertenezca a la clase ω_i dado que hemos observado un valor de la característica x.
- De momento no hemos dicho nada sobre la distribución de probabilidad $p(x|\omega_i)$. P.e. una distribución normal.

Teoría Bayesiana de decisión

- Decisión:
  - Consideramos que el patrón x pertenece a la clase ω_1 si:

$$P(\omega_1|x) > P(\omega_2|x)$$

    - En caso contrario, pertenece a la clase ω_2.
  - Es decir:

$$\frac{p(x|\omega_1)\cdot P(\omega_1)}{p(x)} > \frac{p(x|\omega_2)\cdot P(\omega_2)}{p(x)}$$

  - *p(x)* actúa como un normalizador, asegura que:

$$P(\omega_1|x) + P(\omega_2|x) = 1$$

  - Podemos prescindir de él para la decisión:

$$p(x|\omega_1)\cdot P(\omega_1) > p(x|\omega_2)\cdot P(\omega_2)$$

  - En definitiva, elegimos la pertenencia a la clase ω_j de manera que:

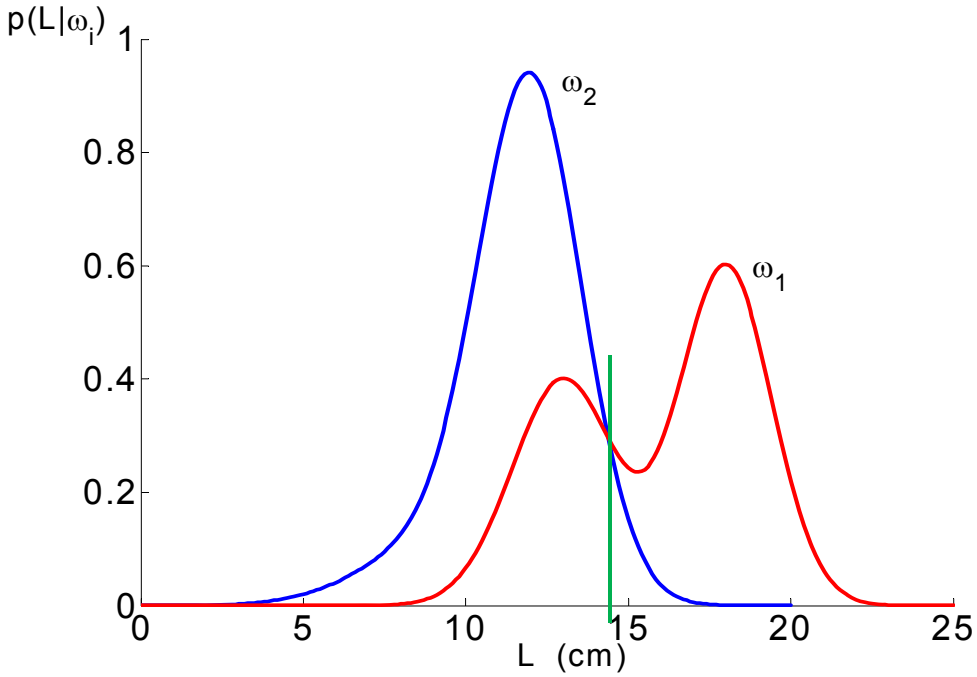
$$\boxed{p(x|\omega_j)\cdot P(\omega_j) > p(x|\omega_i)\cdot P(\omega_i)}$$

Teoría Bayesiana de decisión

- Si las probabilidades a priori son iguales, entonces elegimos la clase ω_j si:

$$p(x|\omega_j) \cdot P(\omega_j) > p(x|\omega_i) \cdot P(\omega_i)$$

$$p(x|\omega_j) > p(x|\omega_i)$$



Teoría bayesiana de decisión

- Ejercicio:
  - Tenemos una captura de peces: Salmones y Merluzas.
  - Las probabilidades a priori, son:
    - P(SALMÓN)=0.2 ← nº total de salmones/nº total de peces
    - P(MERLUZA)=0.8. ← nº total de merluzas/nº total de peces
  - Nos hemos dedicado a pesar los peces con una báscula. Decimos que el peso es ALTO, cuando supera 5 kg. BAJO, en caso contrario.
  - Con esto, hemos calculado las siguientes probabilidades.
    - P(ALTO|SALMÓN)=0.8, probabilidad de que el peso sea alto, dado que sabemos que estamos pesando un salmón
      → cogemos todos los salmones P(ALTO|SALMÓN)=nº de salmones con peso alto/nº total de salmones
    - P(BAJO|SALMÓN)=0.2
    - P(ALTO|MERLUZA)=0.6
    - P(BAJO|MERLUZA)=0.4
  - La ley de probabilidad nos dice (Regla de Bayes): $P(\omega_i|x) = \frac{p(x|\omega_i) \cdot P(\omega_i)}{p(x)}$
  - Para el caso de salmones y merluzas:

$$P(salmon|ALTO) = \frac{p(ALTO|salmón) \cdot P(salmón)}{p(ALTO)}$$

$$P(merluza|ALTO) = \frac{p(ALTO|merluza) \cdot P(merluza)}{p(ALTO)}$$

$$P(salmón|BAJO) = \frac{p(BAJO|salmón) \cdot P(salmón)}{p(BAJO)}$$

$$P(merluza|BAJO) = \frac{p(BAJO|merluza) \cdot P(merluza)}{p(BAJO)}$$

Teoría bayesiana de decisión

- Imaginad que pesamos un pescado y obtenemos que el peso es ALTO.
  - ¿Qué pez es? ¿Apuestas?
  - ¿Con qué probabilidad es un salmón?
  - ¿Con qué probabilidad es una merluza?
  - ¿Por qué opción apostaríais?
    - Por la más probable… a la larga ganaremos un mayor número de apuestas así…

P(SALMÓN)=0.2
P(MERLUZA)=0.8

P(ALTO|SALMÓN)=0.8
P(BAJO|SALMÓN)=0.2
P(ALTO|MERLUZA)=0.6
P(BAJO|MERLUZA)=0.4

- Solución:
  - Probabilidad total de ocurrencia:
    - P(ALTO)=P(ALTO|SALMÓN)*P(SALMÓN)+P(ALTO|MERLUZA)*P(MERLUZA)=
      = 0.8*0.2 *0.6*0.8 = 0.64
    - P(BAJO)=P(BAJO|SALMÓN)*P(SALMÓN)+P(BAJO|MERLUZA)*P(MERLUZA)=
      = 0.2*0.2 *0.4*0.8 = 0.36

  - Hemos pesado un pescado y nos da Peso=ALTO.
  - Con qué probabilidad es un SALMÓN, con qué una merluza?
    - P(SALMÓN|ALTO)=P(ALTO|SALMÓN)*P(SALMÓN)/P(ALTO)=0,8*0,2/0,64=0,25
    - P(MERLUZA|ALTO)=P(ALTO|MERLUZA)*P(MERLUZA)/P(ALTO)=0,6*0,8/0,64=0,75
    - P(MERLUZA|ALTO)>P(SALMÓN|ALTO)

  - En resumen elegiremos la clase: $P(w_i|ALTO) > P(w_j|ALTO)$
  - Podemos prescindir del normalizador P(ALTO) en la decisión.
    - P(ALTO|SALMÓN)P(SALMÓN)>P(ALTO|MERLUZA)P(MERLUZA)

Teoría bayesiana de decisión

- Repetimos ahora la medida, y obtenemos que el peso es BAJO.
  - ¿Qué pez es?
  - P(SALMÓN|BAJO)=P(BAJO|SALMÓN)*P(SALMÓN)/P(BAJO)
    =0,2*0,2/0,36=0,11
  - P(MERLUZA|BAJO)=P(BAJO|MERLUZA)*P(MERLUZA)/P(BAJO)
    =0,4*0,8/0,36=0,89
  - P(MERLUZA|BAJO)>P(SALMÓN|BAJO)
  - P(MERLUZA|ALTO)>P(SALMÓN|ALTO)

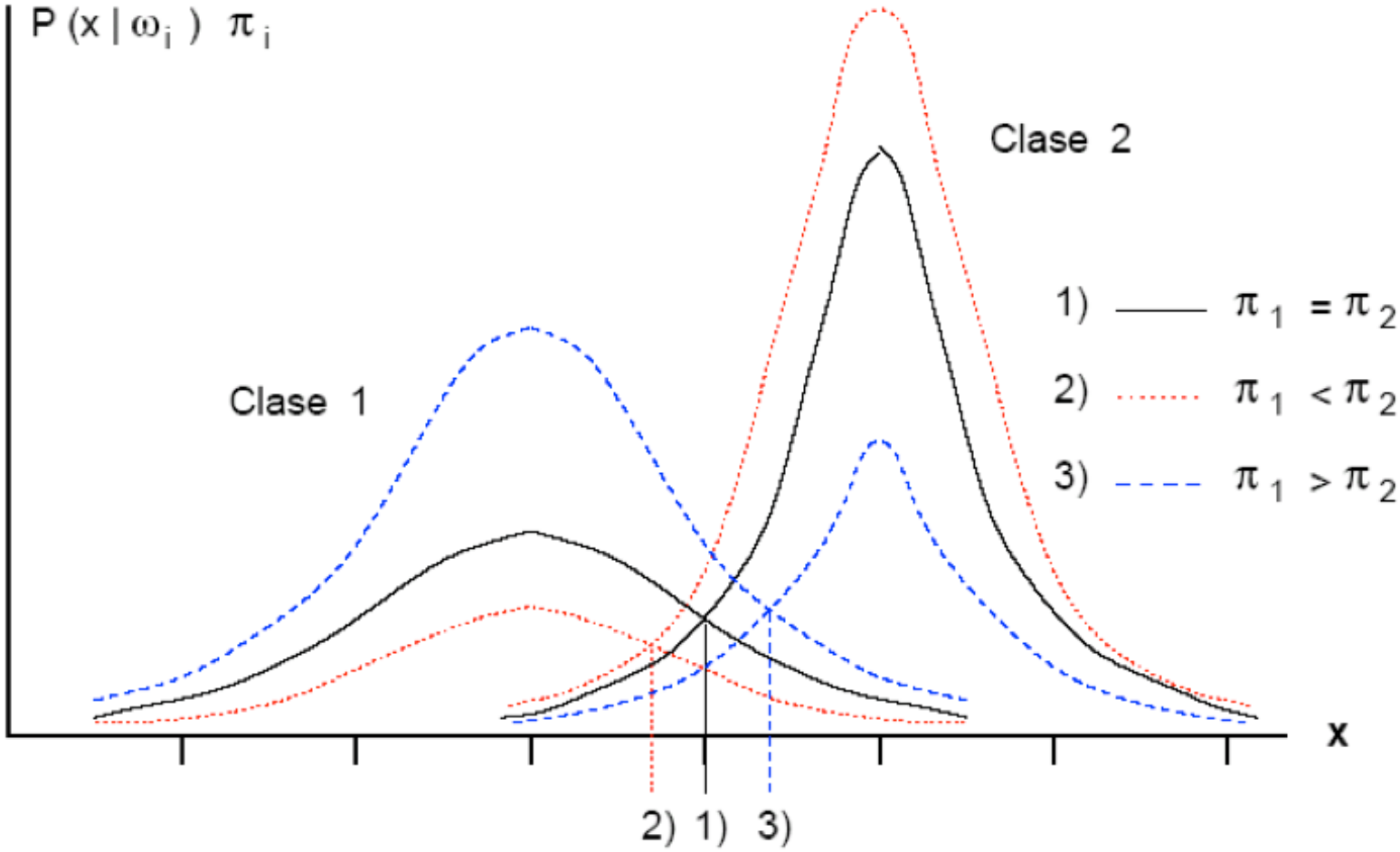
P(SALMÓN)=0.2
P(MERLUZA)=0.8

P(ALTO|SALMÓN)=0.8
P(BAJO|SALMÓN)=0.2
P(ALTO|MERLUZA)=0.6
P(BAJO|MERLUZA)=0.4

P(ALTO)=0.64
P(BAJO)=0.36

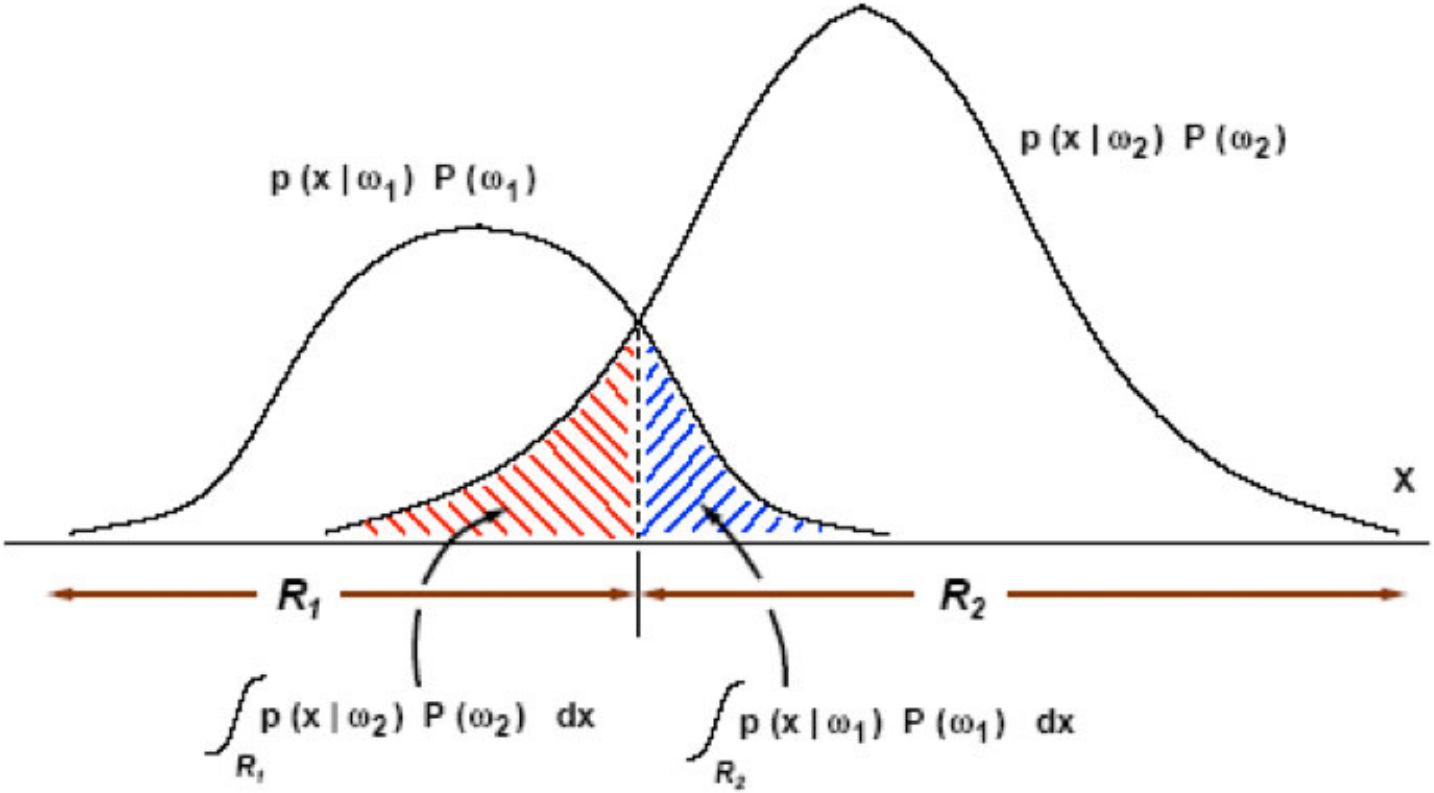
Teoría Bayesiana de decisión

- Influencia de la probabilidad a priori de cada clase en la clasificación:



Teoría Bayesiana de decisión

- Probabilidad de Error en la clasificación:



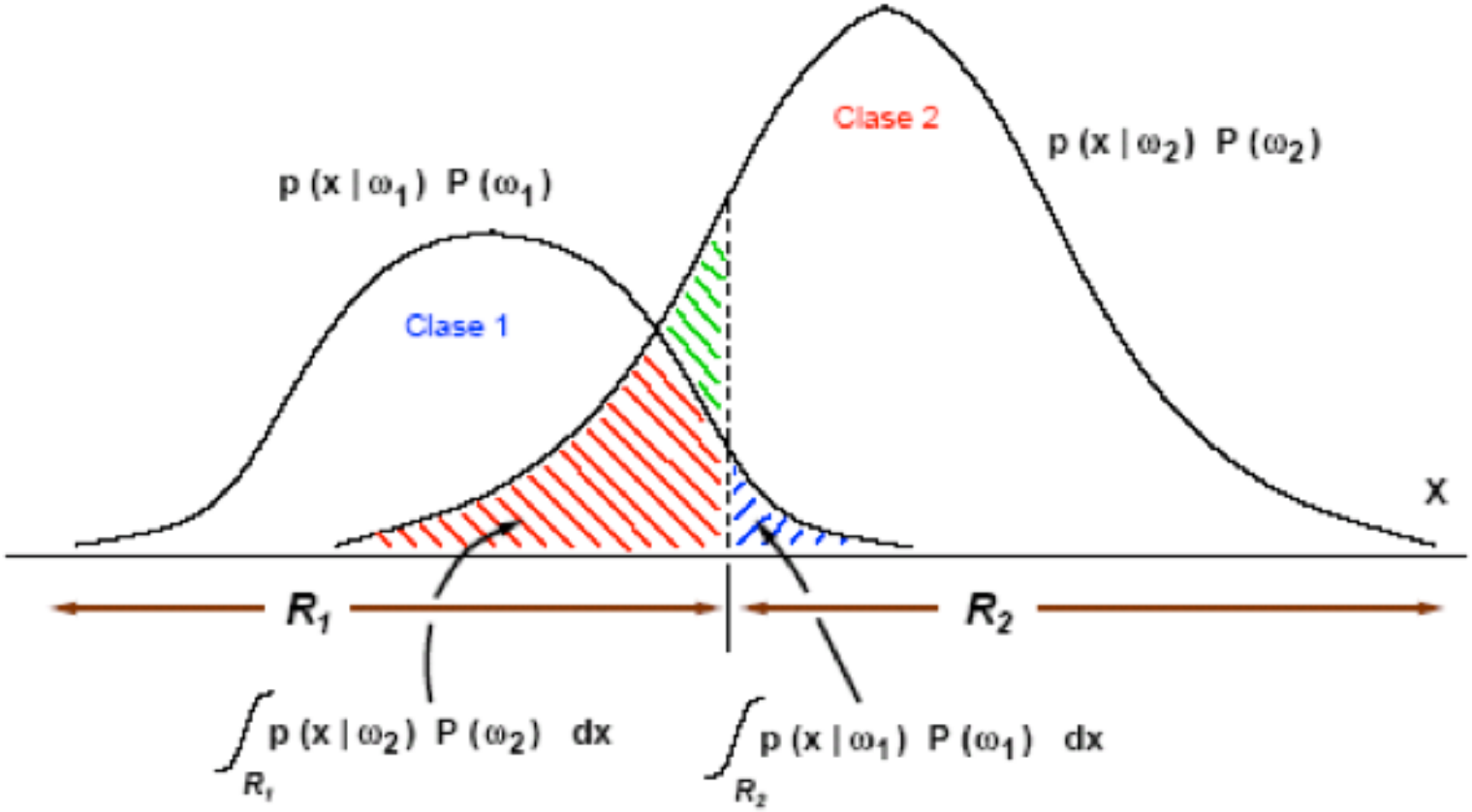
$$P(error) = P(x \in R_2 \mid \omega_1) + P(x \in R_1 \mid \omega_2)$$

$$P(error) = p(x \in R_2 | \omega_1) \cdot P(\omega_1) + p(x \in R_1 | \omega_2) \cdot P(\omega_2)$$

$$P(error) = \int_{R_2} p(x \mid \omega_1) \cdot P(\omega_1)\, dx + \int_{R_1} p(x \mid \omega_2) \cdot P(\omega_2)\, dx$$

Teoría Bayesiana de decisión

- Probabilidad de Error en la clasificación:



La probabilidad de Error no es mínima

$$P(acierto) = 1 - P(error)$$

Teoría Bayesiana de decisión

- Generalizamos:
  - Podemos generalizar el problema para el caso de que contemos con un vector de características $\mathbf{x}$ de dimensión d y que las posibles clases se representan como $\{\boldsymbol{\omega}_1 \dots \boldsymbol{\omega}_c\}$ (c clases diferentes).
  - Espacio de características: $\mathbf{x} = [x_1, x_2 \dots x_d]$
  - La probabilidad de pertenencia a la clase i (ω_i) condicionada a la observación del patrón $\mathbf{x}$ es entonces.

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

  - Igual que en el ejemplo, la decisión es:

$$\boxed{p(\mathbf{x}|\omega_j)P(\omega_j) > p(\mathbf{x}|\omega_i)P(\omega_i)}$$

Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

Funciones de decisión

- Existen diferentes maneras de representar un clasificador. La más habitual es mediante c funciones de decisión.

$$g_i(\mathbf{x}) \quad i = 1, \ldots, c$$

- En este caso, el clasificador asigna un patrón $\mathbf{x}$ a la clase ω_i si:

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$
$$\forall\, j \neq i$$

- La elección de un conjunto de funciones de decisión no es única.
  - P.e. podemos multiplicar todas las funciones por una constante y no afectar al resultado.
  - Podemos transformarlas por una función monótona creciente:

    $f(g)$

Funciones de decisión

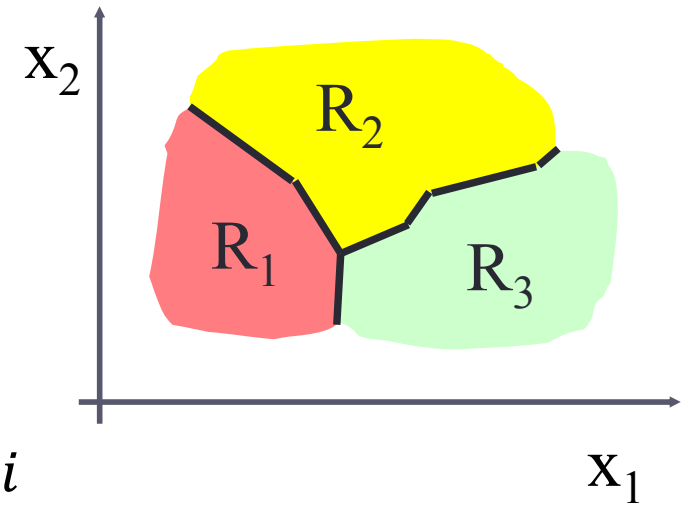
- Podemos elegir, por ejemplo:

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i)$$

$$g_i(\mathbf{x}) = \ln p\,(\mathbf{x}|\omega_i) + \ln P\,(\omega_i)$$

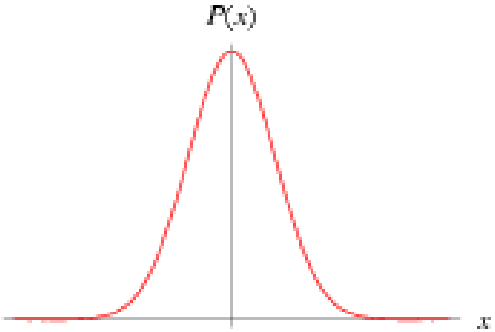
- Las funciones de decisión dividen el espacio de características en *c* regiones.



- Si: $g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall\, j \neq i$
  - Entonces el patrón $\mathbf{x}$ se encuentra en la región $\boldsymbol{R_i}$ y debemos asignarlo a la clase $\boldsymbol{\omega_i}$

Funciones de decisión

- Por ejemplo, supongamos que $d=1$ (tenemos una única característica)
- Suponemos también que $p(x|\ \omega_i)$ es una función de probabilidad gaussiana.



$$p(x|\omega_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(x-\mu_i)^2}{2\sigma^2}\right)}$$

- Si tenemos d características, entonces la función se generaliza en una multinormal.

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right)$$

- Matriz de covarianzas: $\boldsymbol{\Sigma}_i = \frac{1}{m}\sum_{j=1}^{m}(\mathbf{x}-\boldsymbol{\mu}_i)\cdot(\mathbf{x}-\boldsymbol{\mu}_i)^T$

Funciones de decisión

- Por ejemplo, si suponemos una distribución multinormal para la clase ω_i.

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right)$$

- La función de decisión se convierte en:

$$g_i(\mathbf{x}) = \ln p\,(\mathbf{x}|\omega_i) + \ln P\,(\omega_i)$$
$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\,\pi - \frac{1}{2}\ln|\,\boldsymbol{\Sigma}_i| + \ln P\,(\omega_i)$$

- Que es más sencilla de calcular…
- ¿Qué más se puede obviar de $g_i(x)$?
- ¿Y si las clases son equiprobables…? $P(\omega_i)=P(\omega_j)$

Funciones de decisión

- Si todas las características son independientes entre sí y tienen la misma varianza, entonces:

$$\mathbf{\Sigma}_i = \sigma^2 I \qquad\qquad p(\mathbf{x}|\omega_i) = \prod_{j=1}^{n} p\left(x_j|\omega_i\right)$$

- En este caso particular, las funciones de decisión son lineales (*Naive Bayes*):

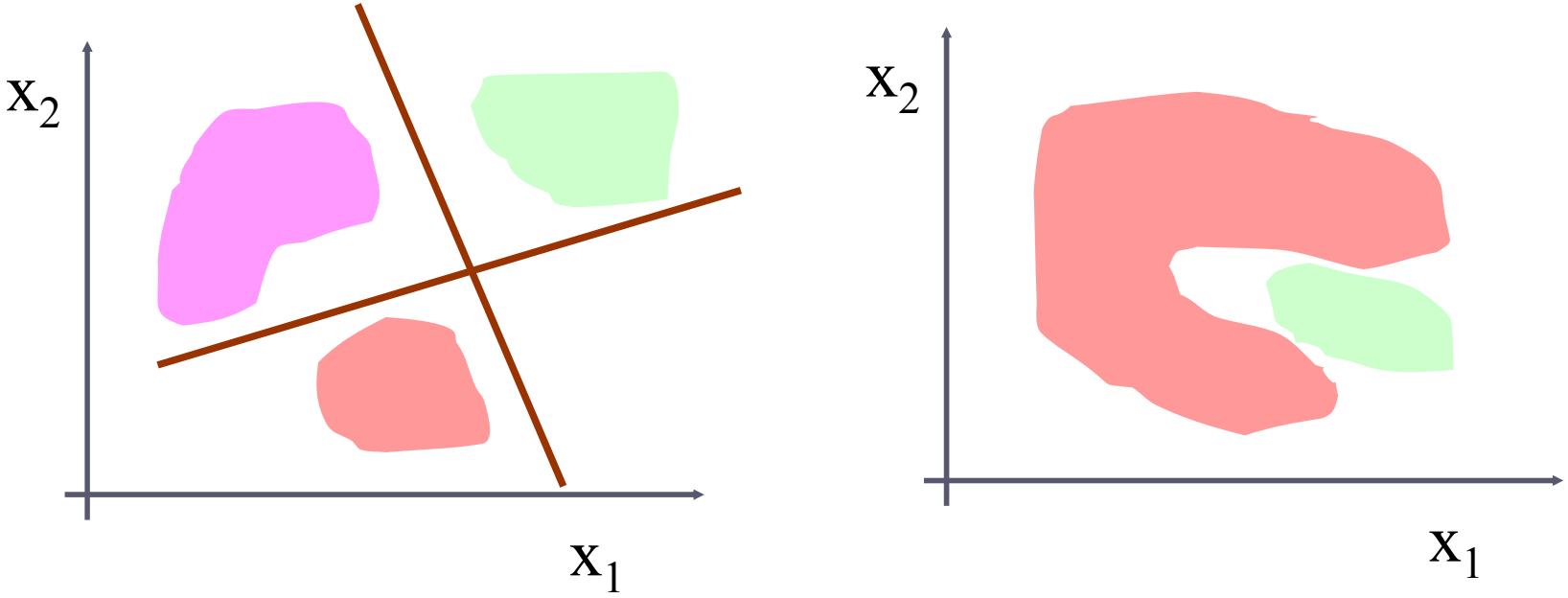
$$g_i(\mathbf{x}) = \theta_i^t \mathbf{x} + k_{i0}$$

$$\theta_i = \frac{1}{\sigma^2}\mu_i$$

$$k_{i0} = \frac{-1}{2\sigma^2}\mu_i^t \mu_i + \ln P\left(\omega_i\right)$$

Funciones de decisión

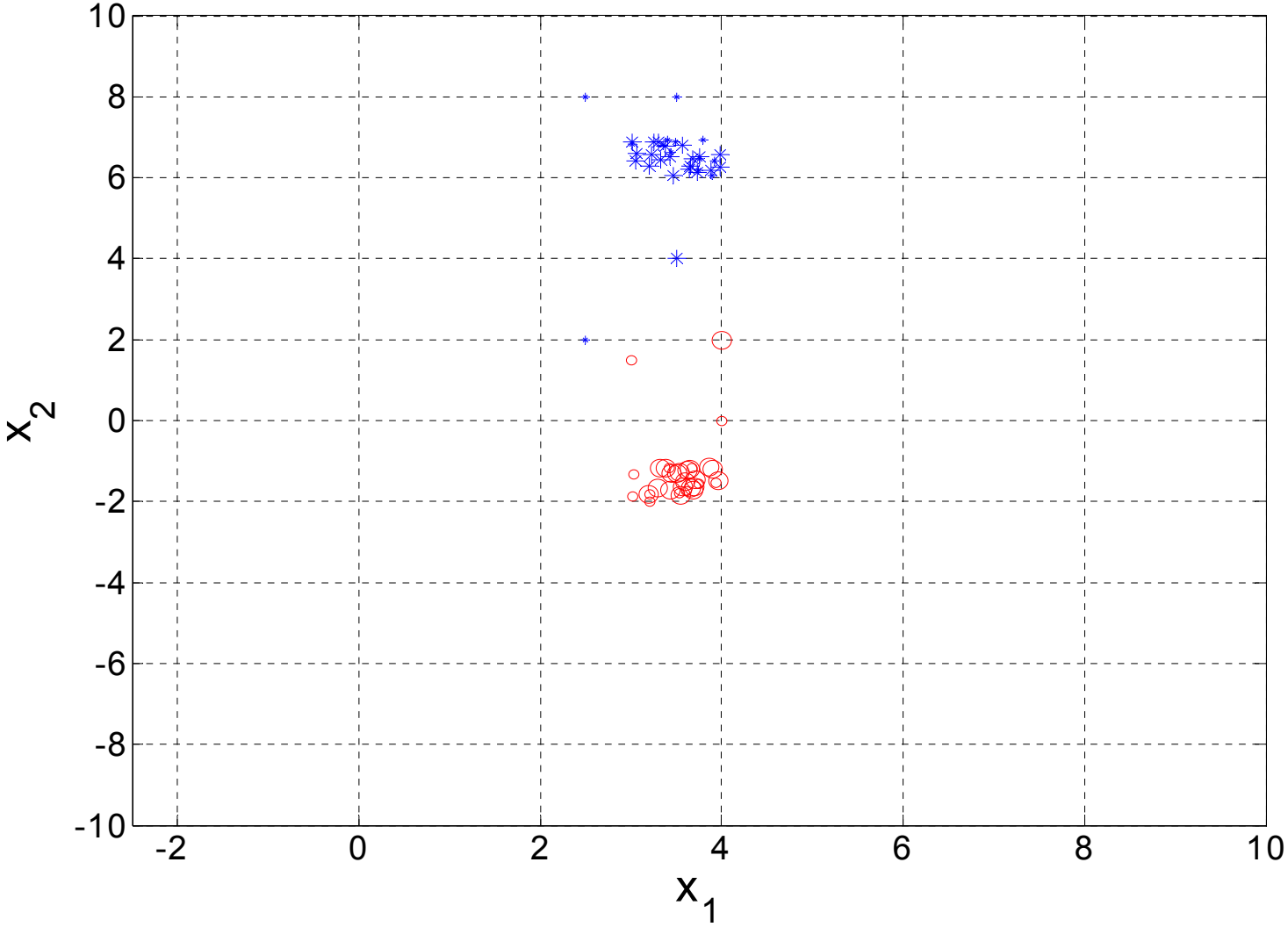
- En general, las clases cuya envolvente conexa no se corte se pueden separar mediante funciones de decisión lineales



- Un clasificador basado en funciones lineales se denomina: "*Máquina Lineal*"

Funciones de decisión

- Ejemplo: Queremos obtener una función de separación entre dos clases supuestas gaussianas.



Funciones de decisión

- Suponemos que los patrones se distribuyen según una binormal:

$$\mu_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix} \quad ; \quad \mu_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\Sigma_1^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$

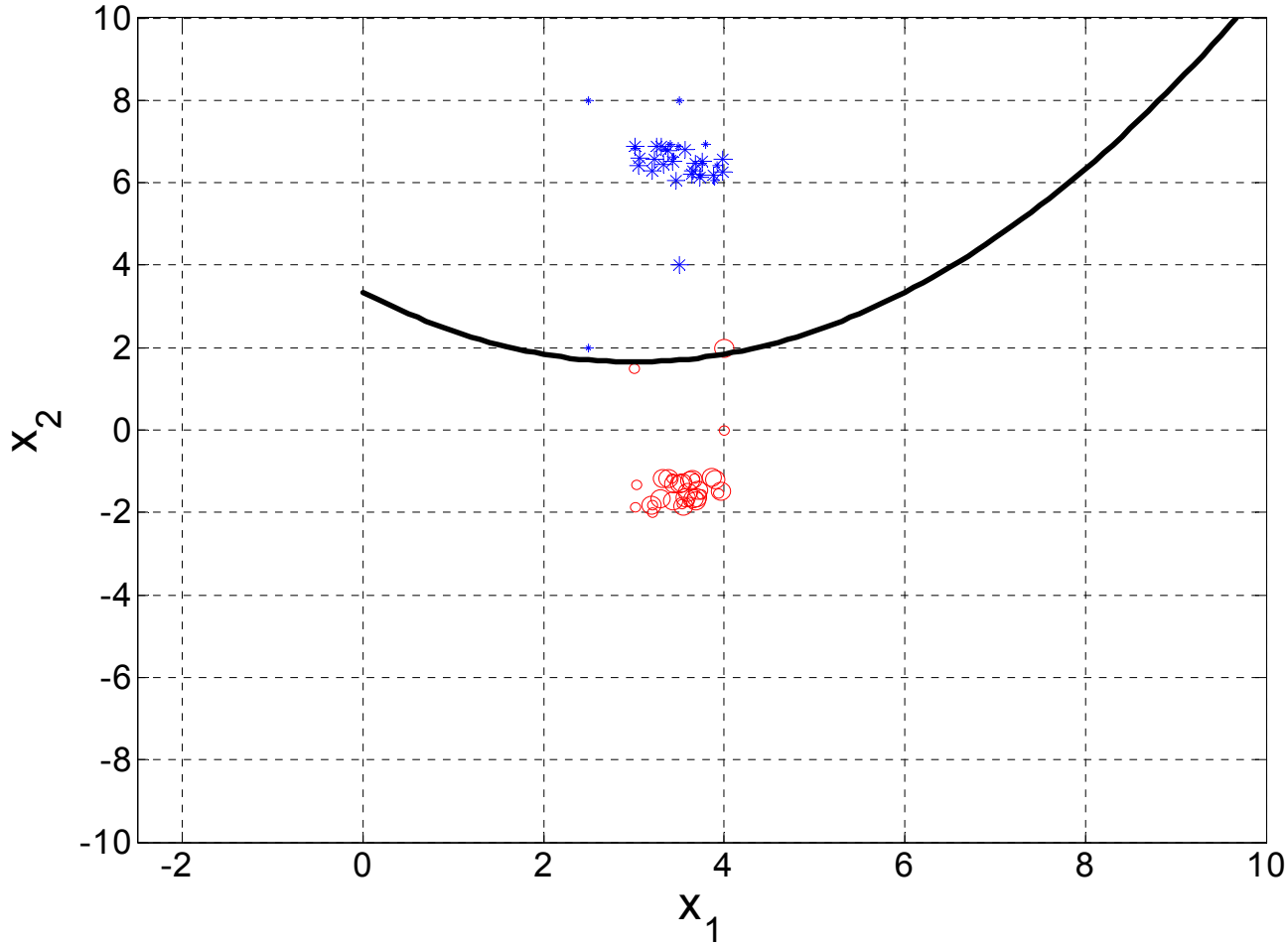
$$g_1(\mathbf{x}) = -\frac{1}{2}[2(x_1 - 3)^2 + 0.5(x_2 - 6)^2] - \frac{2}{2}\ln 2\pi - \frac{1}{2}\ln 1 + \ln 0.5$$

$$g_2(\mathbf{x}) = -\frac{1}{2}[0.5(x_1 - 3)^2 + 0.5(x_2 + 2)^2] - \frac{2}{2}\ln 2\pi - \frac{1}{2}\ln 4 + \ln 0.5$$

- Elegimos la clase 1 si: $g_1(\mathbf{x}) > g_2(\mathbf{x})$
- En caso contrario elegimos la clase 2

Funciones de decisión

- Para extraer la frontera entre las dos clases hacemos: $g_1(\mathbf{x}) = g_2(\mathbf{x})$
- De donde se obtiene: $x_2 = 3.34 - 1.125x_1 + 0.1875x_1^2$



Funciones de decisión

- Suponemos que los patrones se distribuyen según una binormal con:

$$\mu_1 = \begin{bmatrix} 1 \\ 5 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma_1 = \Sigma_2 = \Sigma \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad \Sigma^{-1} = 2 \cdot I$$

$$P(\omega_1) = 0.2 \quad P(\omega_2) = 0.8$$

- ¿Cómo es la función que separa las dos clases?

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \cancel{\frac{d}{2}\ln 2\pi} - \cancel{\frac{1}{2}\ln|\boldsymbol{\Sigma}_i|} + \ln P(\omega_i)$$

$$g_1(\mathbf{x}) = -[(x_1 - 1)^2 + (x_2 - 5)^2] + \ln 0.2$$

$$g_2(\mathbf{x}) = -[{x_1}^2 + {x_2}^2] + \ln 0.8$$

$$g_1(\mathbf{x}) = g_2(\mathbf{x}) \Rightarrow$$

$${x_1}^2 + {x_2}^2 - \ln 0.8 = (x_1 - 1)^2 + (x_2 - 5)^2 - \ln 0.2$$

$$0 = -2x_1 - 10x_2 + 26 + \ln\left({}^{0.8}/_{0.2}\right)$$

$$\boxed{x_1 = -5x_2 + (13 + 0.5\ln 4)}$$

Funciones de decisión: distribuciones no Gaussianas

- Si la función de densidad de probabilidad de las características no siguen una Gaussiana:
  - Transformar la característica x_i de forma que se ajuste a una gaussiana
    - Función logarítmica:

$$z = \log(x)$$

    - Potencia fraccionaria (n>=2)

$$z = x^{1/n}$$

  - Ajuste de una función de decisión minimizando una función de coste sin ajustar a un modelo estadístico (FDP). Algoritmos Computacionales:
    - Regresión Logística
    - Maquinas de Vector Soporte (SVM) ('*Support Vector Machines*')

Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

Selección de características

- ¿Qué características me separan correctamente mis datos?→ Selección de características… Problema complejo.

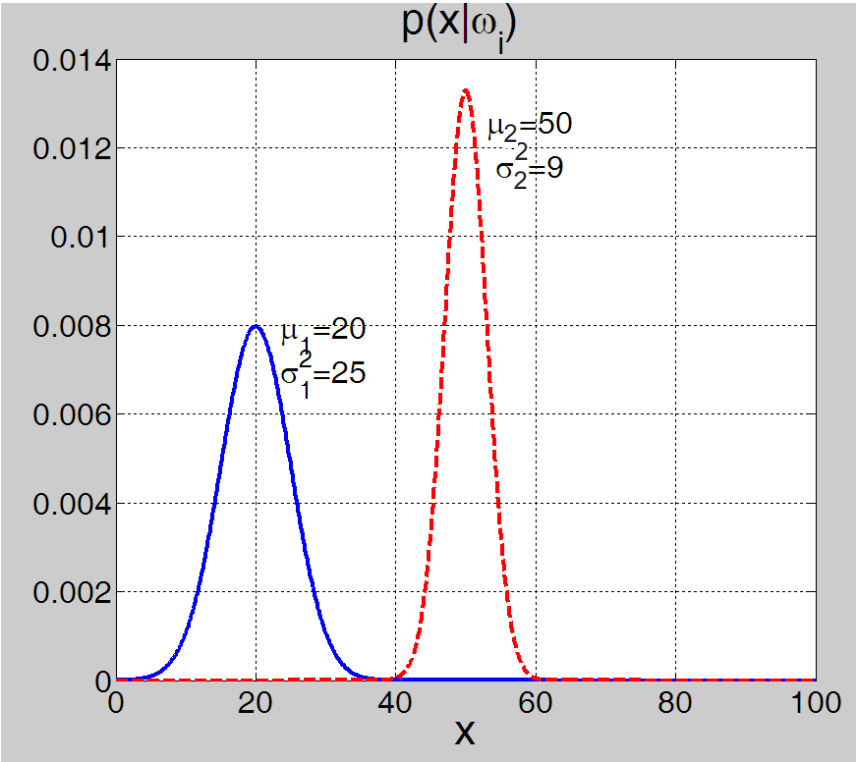
- Métodos:
  - Distancia entre distribuciones (cuando podemos modelarlas)
  - Test Chi-cuadrado (χ^2) (medida de la correlación entre el atributo y las clases)
    - Cuanto mayor es la correlación mayor es la capacidad discriminante
  - Análisis discriminante de Fisher
    - Una característica es tanto más discriminante cuanto mayor es su ratio de Fisher
    - Para una característica k y dos clases i,j el ratio de Fisher viene dado por:

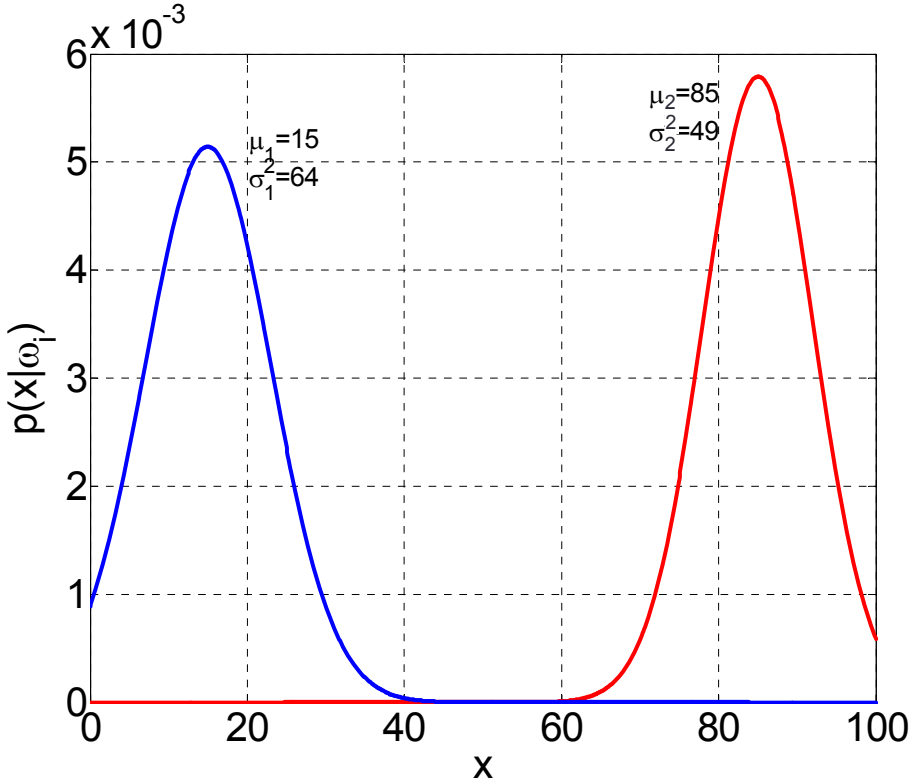
$$F_{ij}^{k} = \frac{\left(\mu_i^k - \mu_j^k\right)^2}{\sigma_i^{k^2} + \sigma_j^{k^2}} \qquad F^k = \frac{\frac{1}{n}\sum_{j=1}^{n}\left(\mu_i^k - \mu_j^k\right)^2}{\frac{1}{n}\sum_{j=1}^{n}\sigma_j^{k^2}}$$

Selección de características

- Podemos definir el siguiente coeficiente de separación de clases para la característica *k (con distribución gaussiana)*:

$$C_{ij}^{k} = \frac{MinDist\{(\mu_i^k \pm 3\sigma_i^k)|(\mu_j^k \pm 3\sigma_j^k)\}}{|\mu_i^k - \mu_j^k|} = \frac{|\mu_i^k - \mu_j^k| - 3\cdot(\sigma_i^k + \sigma_j^k)}{|\mu_i^k - \mu_j^k|}$$





$$C_{ij}^{1} = \frac{(50 - 3*3) - (20 + 3*5)}{|50 - 20|} = 0.2$$

$$C_{ij}^{2} = \frac{(85 - 3*7) - (15 + 3*8)}{|85 - 15|} = 0.357$$

Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Funciones de decisión: Algoritmos computacionales
  - Regresión Logística, SVM
  - Validación
- Otros métodos de reconocimiento

Regresión Logística

- Función de decisión (Hipótesis): $h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^T \cdot \mathbf{x}) \quad 0 \leq h_\theta(\mathbf{x}) \leq 1$

$$\mathbf{x} = [1 \;\; x_1 \;\; \cdots \;\; x_n]^T$$

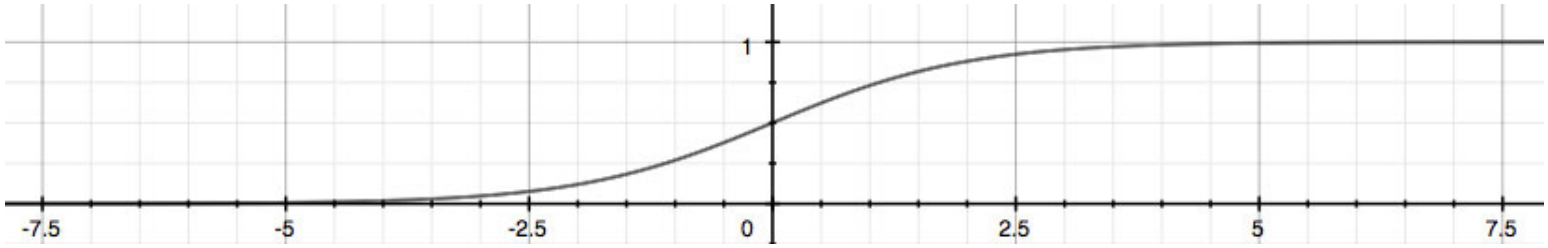
Patrón: vector descripción

$$\boldsymbol{\theta} = [\theta_0 \;\; \theta_1 \;\; \cdots \;\; \theta_n]^T$$

Modelo

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

Función sigmoide (Logística)



$$y \in \{\, 0, \; 1\}$$

Salida: Clasificación binaria

$$\begin{cases} h_\theta(\mathbf{x}) \geq 0.5 \rightarrow y = 1 \\ h_\theta(\mathbf{x}) < 0.5 \rightarrow y = 0 \end{cases}$$

$$y = \; (\boldsymbol{\theta}^T \cdot \mathbf{x} > 0)$$

Regla de decisión lineal

Regresión Logística

- Ajuste del modelo ($\boldsymbol{\theta}$): Minimización función de coste
- Datos entrenamiento:

$$\mathbf{X}_{(m\times n+1)} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$\mathbf{x}^{(i)} = \begin{bmatrix} 1 & x_1^{(i)} \cdots & x_n^{(i)} \end{bmatrix}^T$$

$$\boldsymbol{\theta} = [\theta_0 \;\; \theta_1 \;\; \cdots \;\; \theta_n]^T$$

$$y \in \{0,\; 1\}$$

- Función de Coste:

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(\mathbf{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(\mathbf{x}^{(i)}\right)\right)\right] + \lambda\sum_{j=1}^{n}\theta_j^2$$

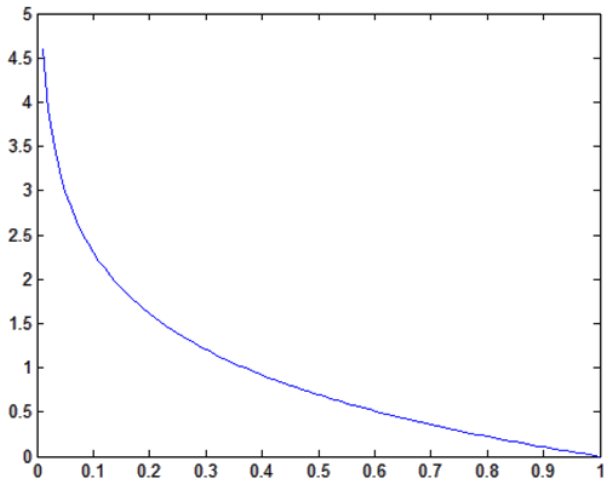
Entropía cruzada
'*cross-entropy*'

$y = 1$

$y = 0$

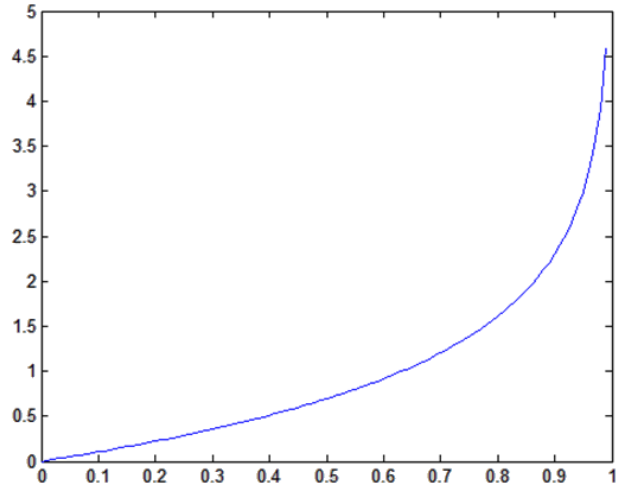
Regularización

$-\log(h_\theta(\mathbf{x}))$



$h_\theta(\mathbf{x})$

$-\log(1-h_\theta(\mathbf{x}))$



$h_\theta(\mathbf{x})$

$$0 \le h_\theta(\mathbf{x}) \le 1$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Regresión Logística

- Minimización función de coste:

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(\mathbf{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(\mathbf{x}^{(i)}\right)\right)\right] + \lambda\sum_{j=1}^{n}\theta_j^2$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Descenso del gradiente (iterativo):

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha\ \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \Longrightarrow \quad {\theta_j}^{t+1} = {\theta_j}^{t} - \alpha\ \frac{\partial}{\partial\theta_j} J(\boldsymbol{\theta})$$

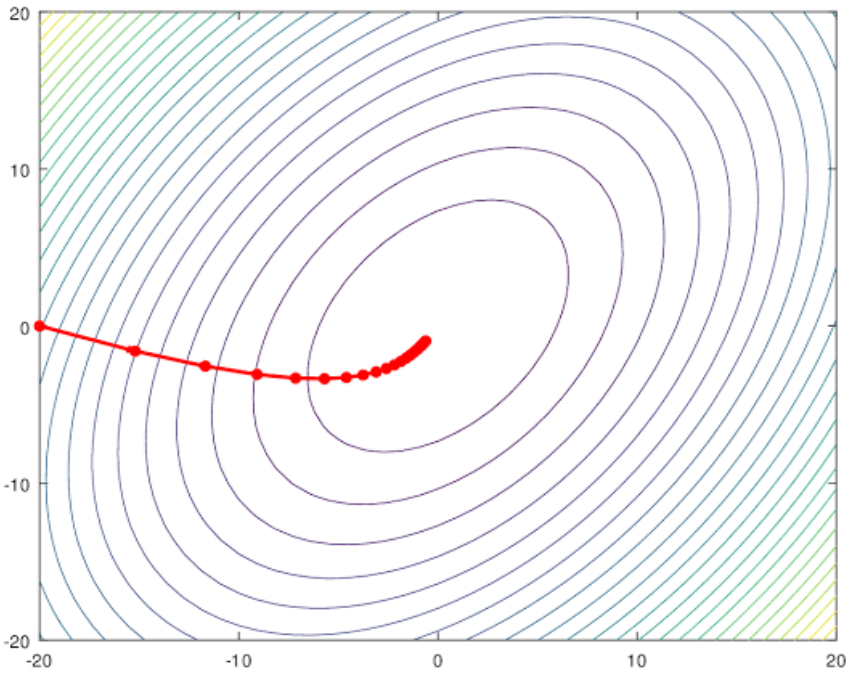
$${\theta_0}^{t+1} = {\theta_0}^{t} - \alpha\ \left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\cdot x_0^{(i)}\right]$$

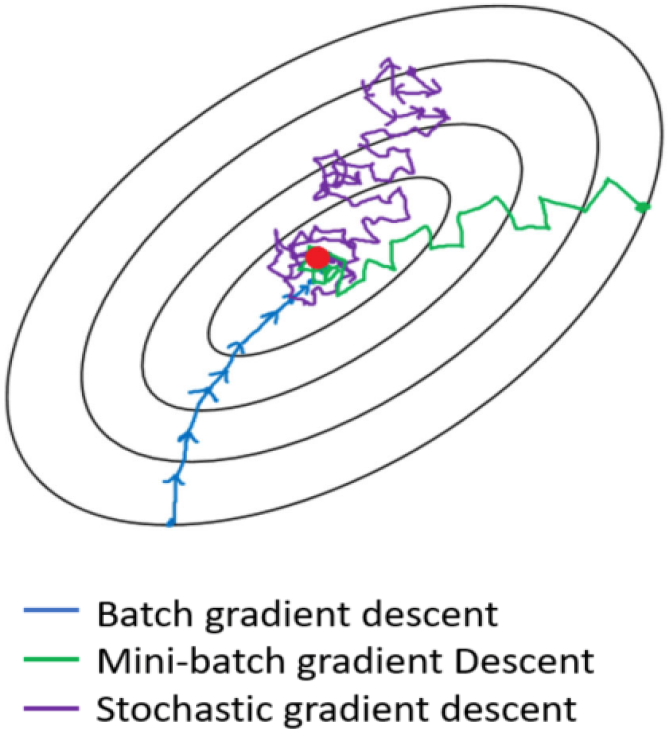
$${\theta_j}^{t+1} = {\theta_j}^{t} - \alpha\ \left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\cdot x_j^{(i)} + \frac{\lambda}{m}\theta_j\right] \quad j \in \{1,2\cdots n\}$$

Regresión Logística

- Minimización función de coste: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha \, \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

$$\theta_j^{\,t+1} = \theta_j^{\,t} - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \cdot x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad j \in \{1,2 \cdots n\}$$





- Mejoras para acelerar el algoritmo:
  - Usar solo una pequeña parte de los datos ($l<m$) en cada iteración (‘***mini-batch***’)
  - Seleccionar aleatoriamente una muestra de los datos en cada iteración (‘***Stochastic Gradient Descent***’)

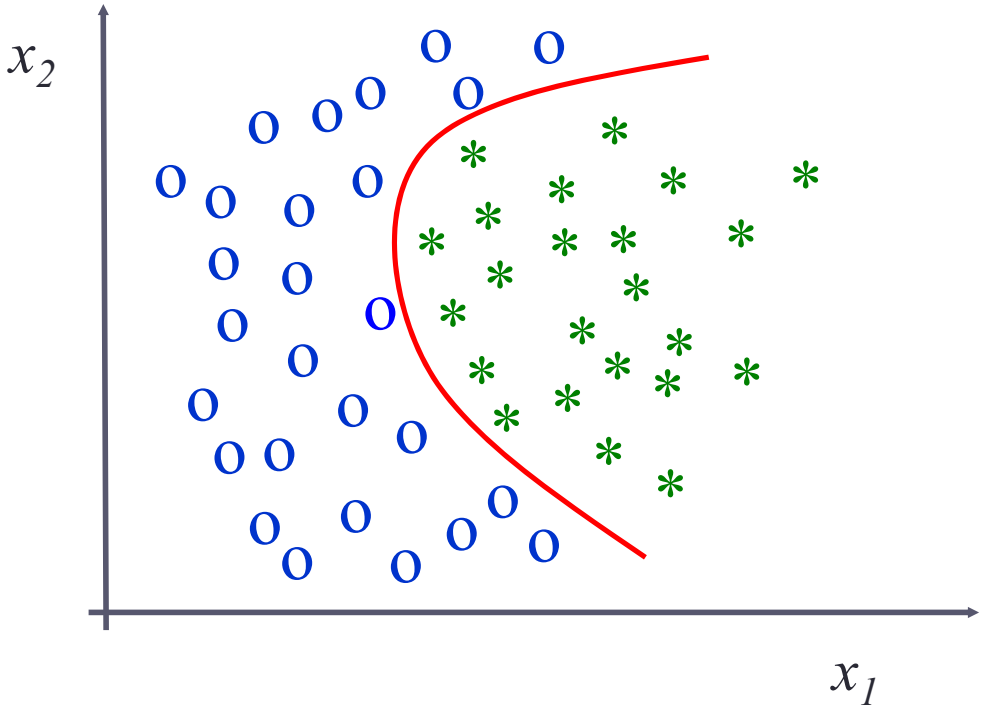
Regresión Logística

- Reglas de decisión no lineales:
  - Añadir variables polinómicas adicionales al vector de descripción

$$\mathbf{x} = [1 \quad x_1 \quad x_2]^T$$

$$\mathbf{x}' = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 \cdot x_2]^T \Longrightarrow \mathbf{x}' = [1 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5]^T$$

  - Permite seguir utilizando el mismo algoritmo de ajuste del modelo
  - El vector de descripción puede crecer mucho pudiendo generar problemas de '*overfitting*'



Regresión Logística

- Clasificación Multi-clase:
  - Estrategia Uno contra Todos (*'one-vs-all'*, *'one-vs-rest'*):

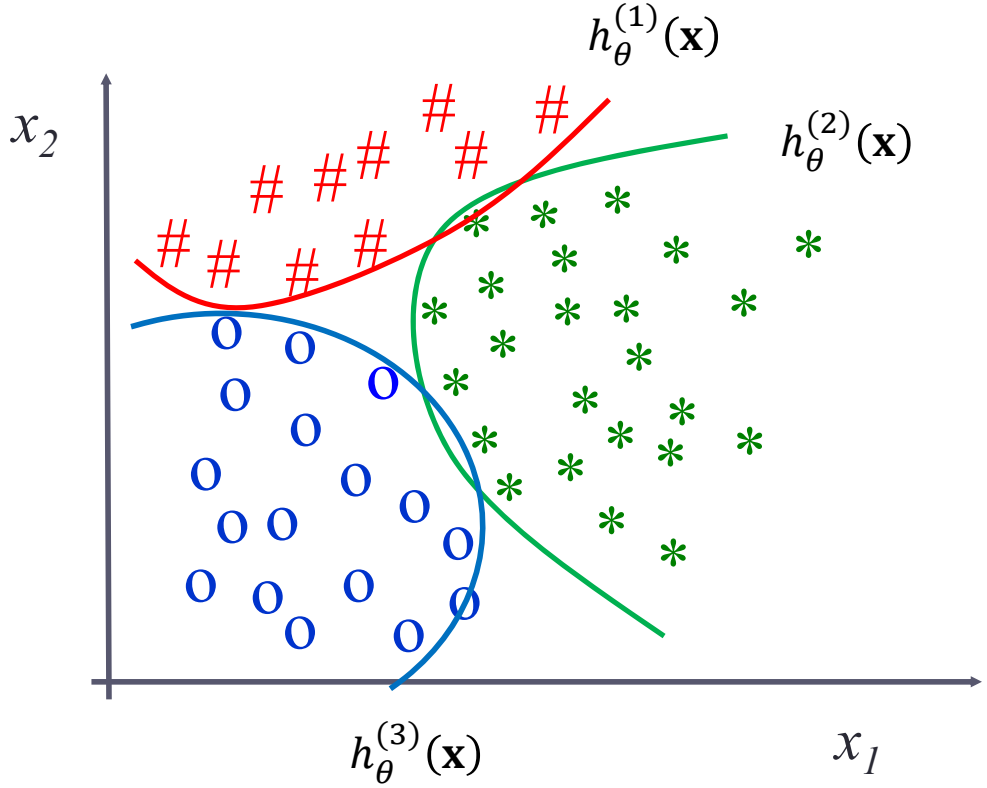
$$\mathbf{x}^{(i)} = \begin{bmatrix} 1 & x_1^{(i)} \cdots & x_n^{(i)} \end{bmatrix}^T \qquad y \in \{1, \cdots d\}$$

  - Se divide en ***d*** problemas de clasificación binaria de un subconjunto de ejemplos ***(y = i)*** frente a todos los demás:

$$h_\theta^{(i)}(\mathbf{x}) = P(y = i | x\,; \theta) \quad (i = 1,2 \cdots d)$$

- Escogeríamos como predicción la de máxima probabilidad:

$$\max_i h_\theta^{(i)}(\mathbf{x})$$



Regresión Logística

- Problemas
  - Sesgo (High Bias) -> Subajuste (Underfitting) -> pocas características
  - Alta Varianza -> Sobreajuste (Overfitting) -> demasiadas características

- Validación del modelo:
  - Error de clasificación:
    - Función de coste: $J(\boldsymbol{\theta})$
    - Tasa de acierto/error (Accuracy): Aciertos/Total de ejemplos
    - Matriz de confusión: Precision/Recall

$$error = \frac{\left(y \neq (h_{\boldsymbol{\theta}}(\mathbf{x}) > 0.5)\right)}{m}$$

  - Debemos dividir los datos disponibles:
    - Datos Entrenamiento: (~60%)
      - Se utilizarán para calcular los parámetros del modelo $\boldsymbol{\theta}$
    - Datos Validación Cruzada (cross-validation): (~20%)
      - Se utilizarán para seleccionar el tipo de modelo y los parámetros de ajuste del modelo: $\mathbf{x}, \lambda, \alpha$
    - Datos de Test: (~20%)
      - Se utilizarán para validar el modelo ajustado

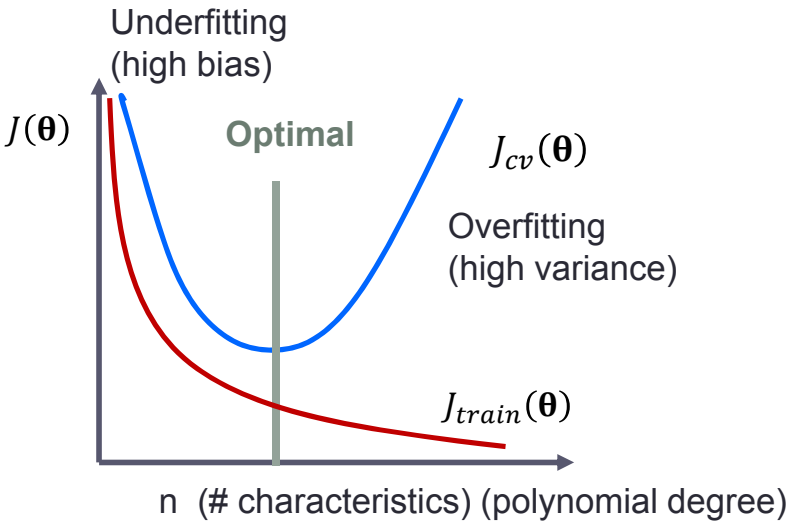
$$\min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta})$$

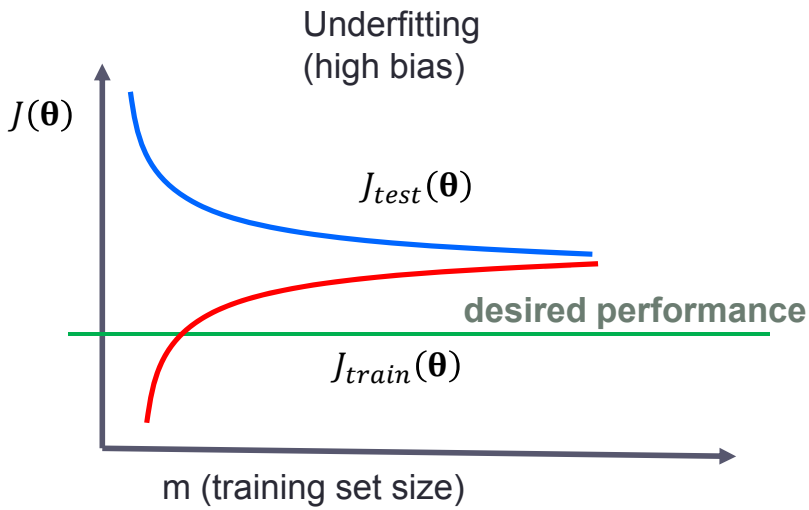
$$\min_{\mathbf{x},\lambda,\alpha} J_{cv}(\boldsymbol{\theta})$$

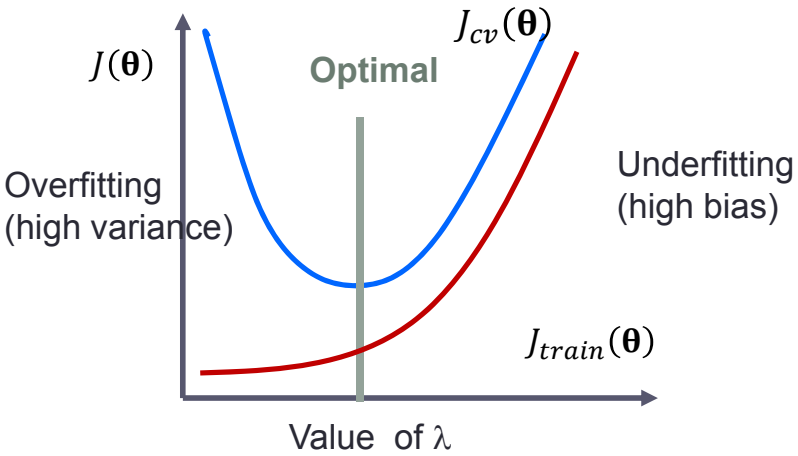
$$J_{test}(\boldsymbol{\theta})$$

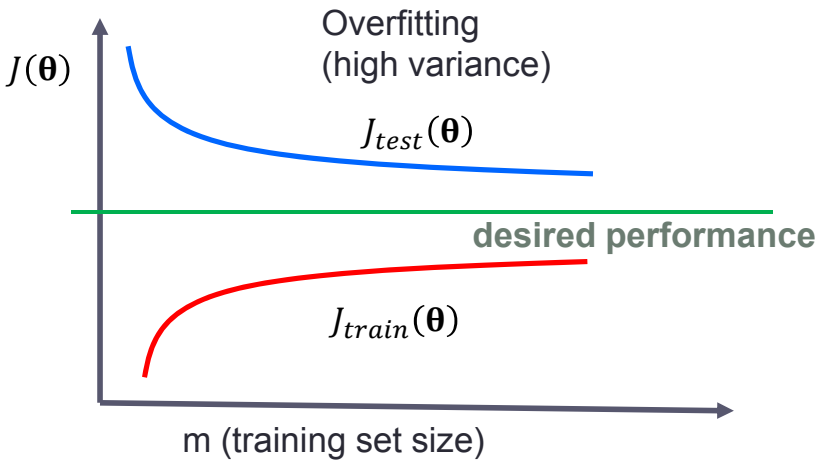
Regresión Logística

- Problemas
  - Sesgo (Bias) -> Subajuste (Underfitting) -> pocas características
  - Alta Varianza -> Sobreajuste (Overfitting) -> demasiadas características
- Análisis: curvas de aprendizaje



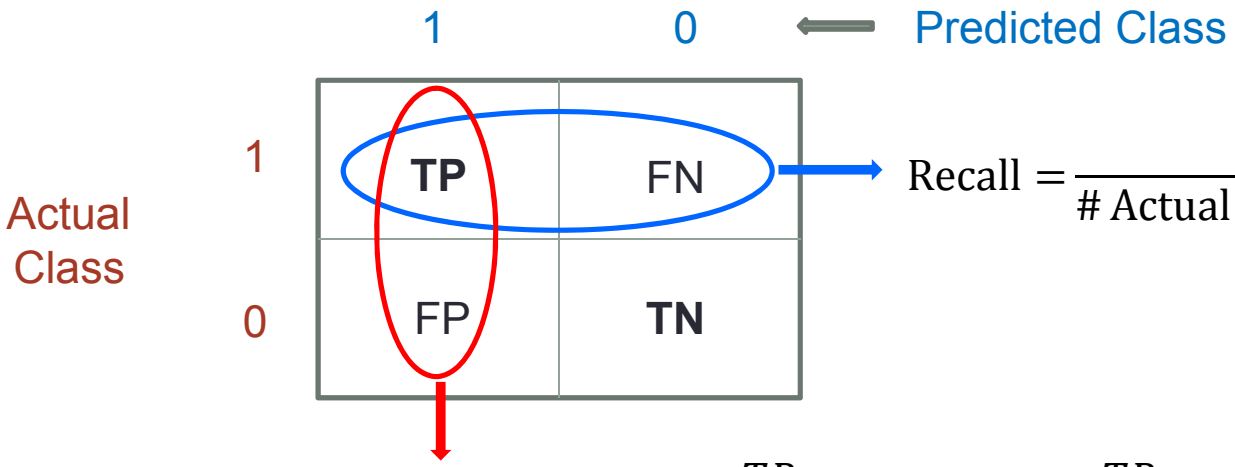






Validación del modelo:

- Matriz de confusión: Precision/Recall, F_1-score
  - Permite una mejor estimación del error cuando hay sesgo en los datos de entrenamiento (muchos datos de una clase y pocos de otra)



$$\text{Recall} = \frac{TP}{\#\,\text{Actual Positives}} = \frac{TP}{TP + FN}$$

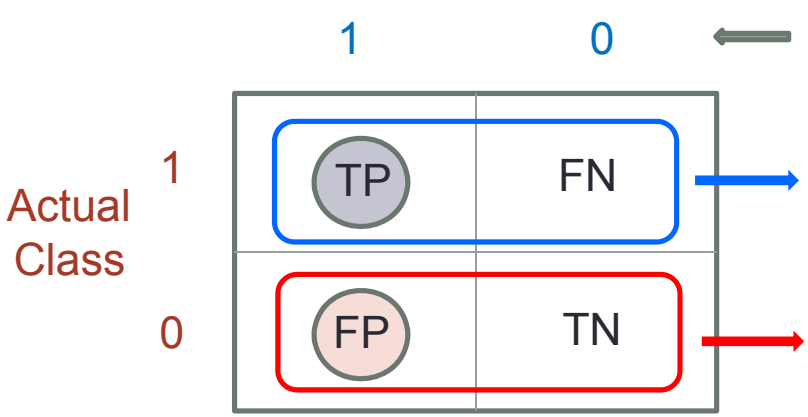
$$\text{Precission} = \frac{TP}{\#\,\text{Predicted Positives}} = \frac{TP}{TP + FP}$$

TP (True Positives): Clasificado correctamente como $\boldsymbol{c_i}$ siendo de la clase $\boldsymbol{c_i}$
FP (False Positive): Clasificado incorrectamente como $\boldsymbol{c_i}$ siendo de la clase $\boldsymbol{c_j}$

$$F_1 - score = 2\frac{P \cdot R}{\mathrm{P} + R}$$

Validación del modelo:

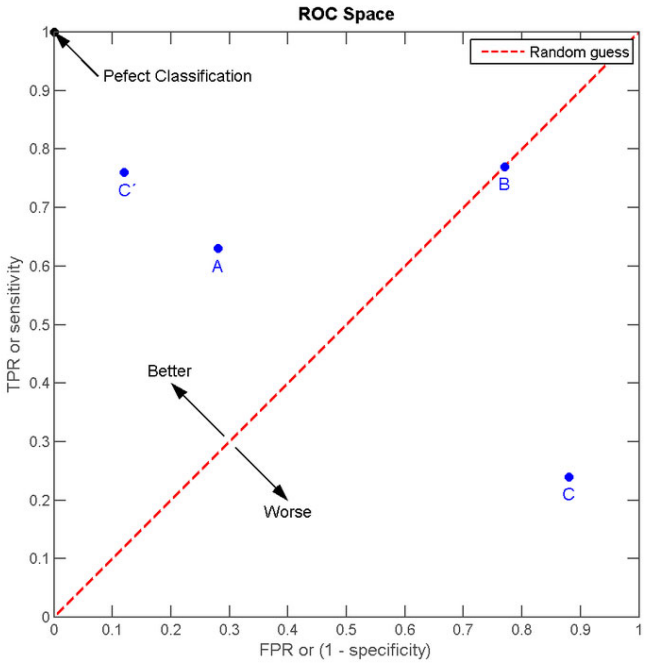
- Curvas **ROC** (‘*Receiver Operator Characteristic*’):

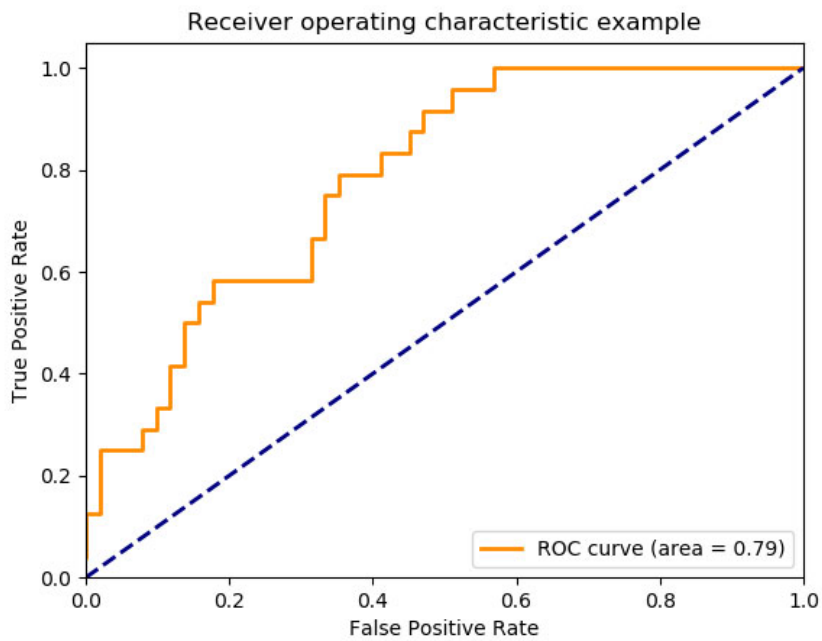


$$TPRate = \frac{TP}{\#\,\text{Actual Positives}} = \frac{TP}{TP + FN}$$ (Recall)

$$FPRate = \frac{FP}{\#Actual\ Negatives} = \frac{FP}{FP + TN}$$

Objetivo: Maximizar el área debajo de la curva





Validación del modelo:

- Matriz de confusión: Multiclase
  - Obtendremos un valor Precission/Recall, F_1-score, curvas ROC por clase

Predicted Class

Actual Class

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & c_{11} & c_{12} & c_{13} \\ 2 & c_{21} & c_{22} & c_{23} \\ 3 & c_{31} & c_{32} & c_{33} \end{array}$$

$$TPRate(i) = \frac{c_{ii}}{\sum_{j=1}^{nc} c_{ij}}$$

TP Rate (i) (True Positives): Clasificado correctamente como $\boldsymbol{c_i}$ siendo de la clase $\boldsymbol{c_i}$ / número de ejemplos de $\boldsymbol{c_i}$

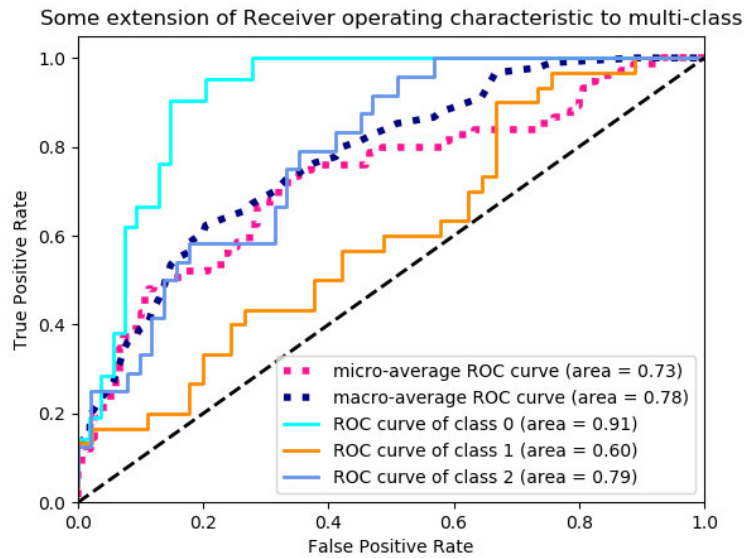
$$FPRate(i) = \frac{\sum_{j \neq i} c_{ji}}{\sum_{k \neq i}^{nc} \sum_{j=1}^{nc} c_{ki}}$$

FP Rate (i) (False Positives): Clasificado incorrectamente como $\boldsymbol{c_i}$ siendo de otra clase / número de ejemplos otras clases $\boldsymbol{c_j}$

$$\text{Precission}(i) = \frac{TP\ c_i}{\#\text{Predicted Positives } c_i} = \frac{c_{ii}}{\sum_{j=1}^{nc} c_{ji}}$$

$$\text{Recall}(i) = \frac{TP\ c_i}{\#\text{Actual Positives } c_i} = \frac{c_{ii}}{\sum_{j=1}^{nc} c_{ij}} = TPRate(i)$$

$$F_1 - score = 2\frac{P \cdot R}{P + R}$$



Ejercicio:

- Dada la siguiente matriz de confusión, calcular las tasas de éxito y error en la clasificación:

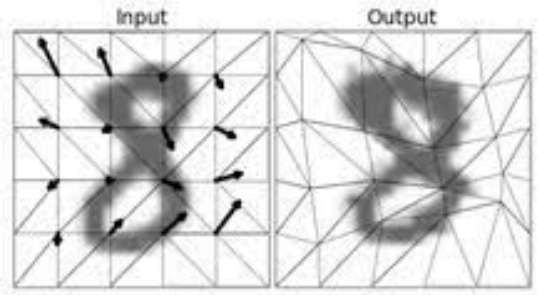
```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
  9  3  0  0  0 |  a = 15
  2 11  0  0  0 |  b = 35
  1  4  7  0  1 |  c = 62
  0  0  0 13  0 |  d = 69
  0  0  2  0 16 |  e = 950
```

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.75	0.053	0.75	0.75	0.75	15
0.846	0.125	0.611	0.846	0.71	35
0.538	0.036	0.778	0.538	0.636	62
1	0	1	1	1	69
0.889	0.02	0.941	0.889	0.914	950

Aumento de Datos: "Data-augmentation"

- Para entrenar los clasificadores se requiere de un gran numero de muestras no sesgadas
- Solución: aplicar trasformaciones geométricas a las imágenes para aumentar los datos de entrenamiento
  - Giros
  - Traslaciones
  - Escalados
  - Espejo vertical y horizontal
  - Trasformaciones elásticas (warping)
  - Borrado aleatorio de partes de la imagen
  - Modificación del color
  - Añadir ruido



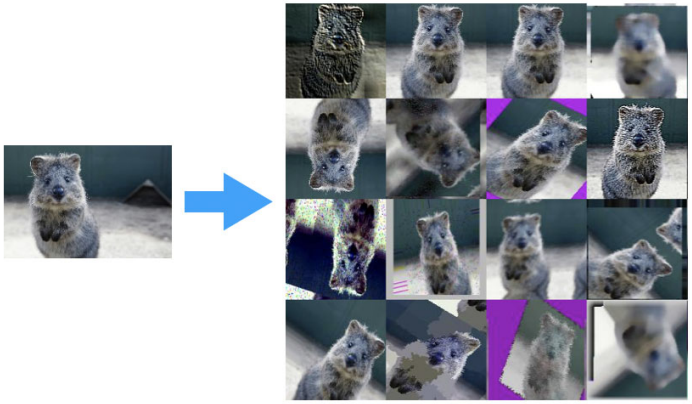


Figure credit: https://github.com/aleju/imgaug

Máquinas de Vector Soporte (SVM)

- Características
  - La base es similar a la regresión logística
  - Diferencias:
    - Función de coste: aproximación lineal en dos tramos
    - Maximiza el margen de separación de los datos a la regla de decisión
    - Introduce el concepto de **kernel** que permite transformar el espacio de características.
    - Se proyecta cada vector de datos **x** sobre un conjunto de puntos en el espacio de características (**vectores soporte**) a través del **kernel**.
    - Permite reglas de decisión no lineales sin necesidad incrementar el tamaño del espacio de características.
    - Precisa escalar y normalizar el vector de características.

  - Regla Decisión/Hipótesis:

$$h_\theta(\mathbf{x}) = \begin{cases} 1 & si \quad \boldsymbol{\theta}^T \cdot \mathbf{x} \geq 0 \\ 0 & si \quad \boldsymbol{\theta}^T \cdot \mathbf{x} < 0 \end{cases}$$

"Support-Vector Networks", Cortes, C., Vapnik, V. Machine Learning 20, 273–297 (1995)

Máquinas de Vector Soporte (SVM)

- Función de coste

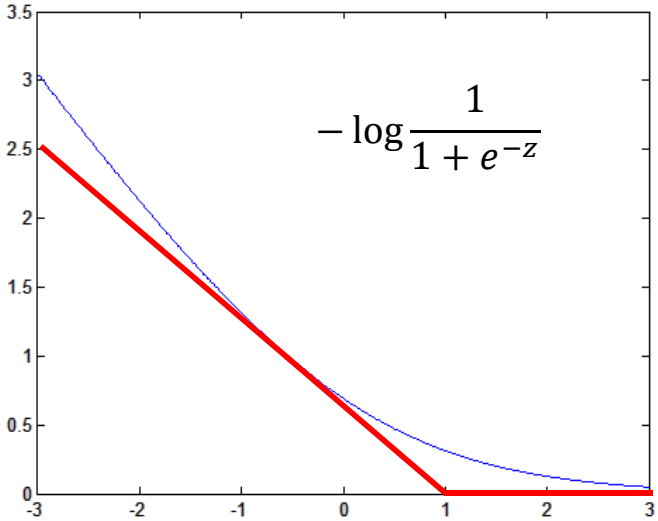
$$J(\boldsymbol{\theta}) = C \sum_{i=1}^{m} \left[y^{(i)} \operatorname{cost}_1\left(\boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \operatorname{cost}_0\left(\boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)}\right) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

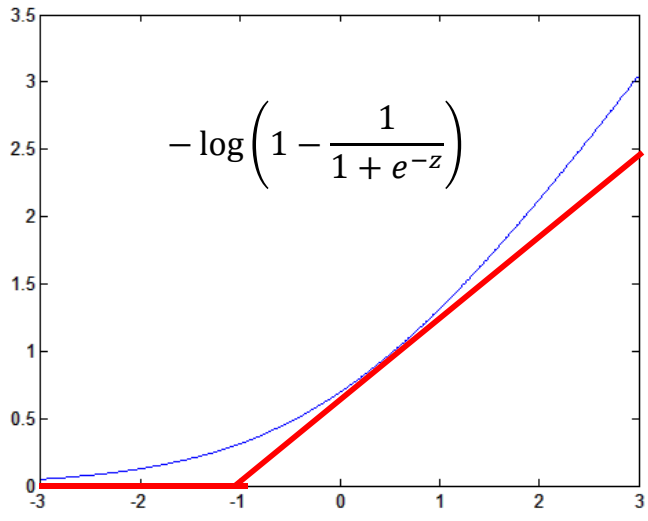
Coeficiente Regularización

$y = 1$

$y = 0$

Regularización





$$\operatorname{cost}_1(\boldsymbol{\theta}^T \cdot \mathbf{x}) = \max\left(0,\ k(1 - \boldsymbol{\theta}^T \cdot \mathbf{x})\right)$$

$$\operatorname{cost}_0(\boldsymbol{\theta}^T \cdot \mathbf{x}) = \max\left(0,\ k(1 + \boldsymbol{\theta}^T \cdot \mathbf{x})\right)$$

$$y = 1 \Rightarrow \boldsymbol{\theta}^T \cdot \mathbf{x} > 1$$

$$y = 0 \Rightarrow \boldsymbol{\theta}^T \cdot \mathbf{x} < -1$$

Maximiza el margen de separación entre clases

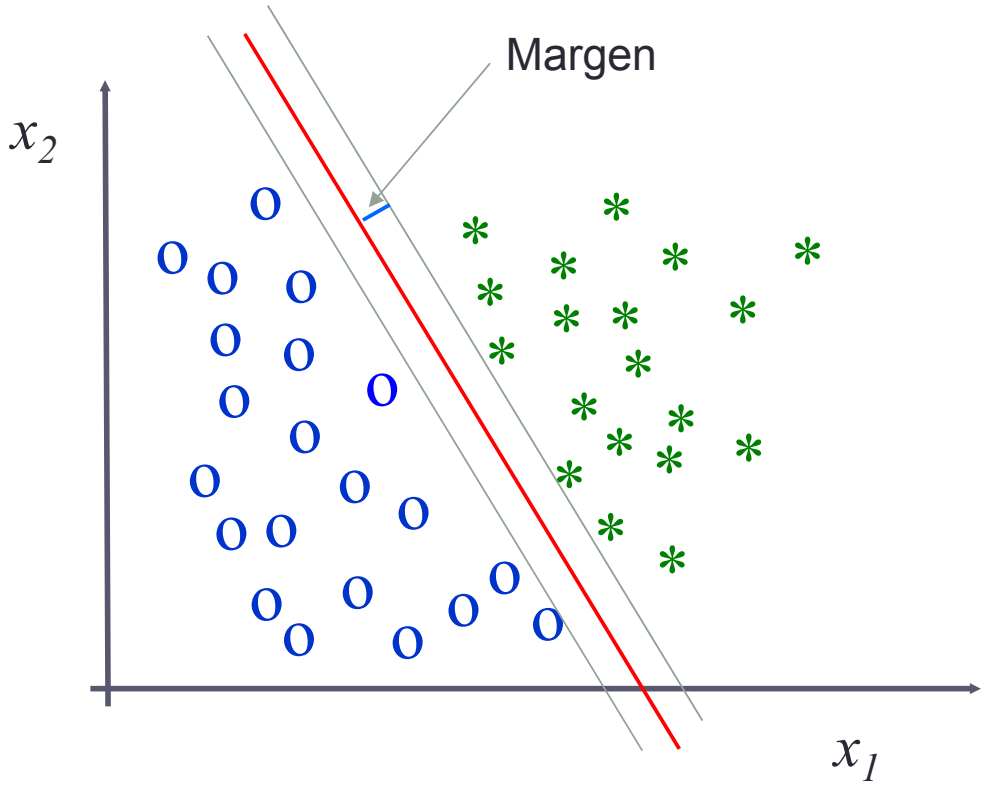
Máquinas de Vector Soporte (SVM)

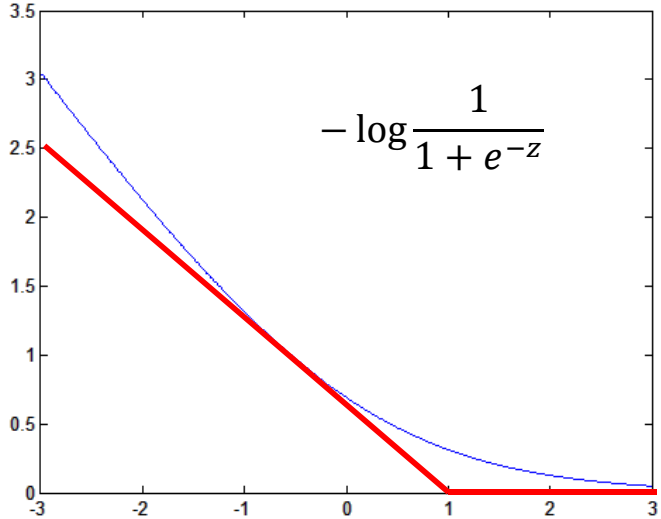
- Margen máximo:

$$y = 1 \Rightarrow \boldsymbol{\theta}^T \cdot \mathbf{x} > 1$$

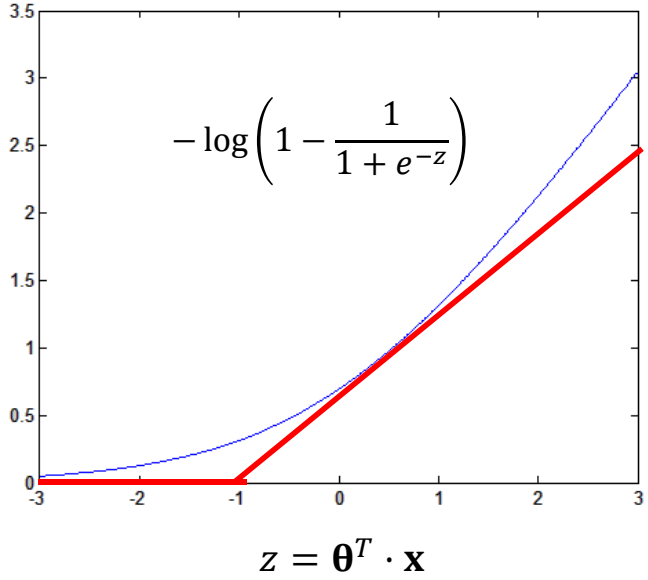
$$y = 0 \Rightarrow \boldsymbol{\theta}^T \cdot \mathbf{x} < -1$$

Maximiza el margen
de separación entre clases





$y = 1$



$y = 0$

Máquinas de Vector Soporte (SVM)

- Transformación mediante Kernels y vectores soporte
  - **Kernel Lineal**: equivalente a no usar kernel (**n** grande y **m** pequeño)
  - **Kernel Gaussiano**: (**n** pequeño y **m** grande)
    - Se toma *s*=*m*: un vector soporte por cada muestra de entrenamiento, (o los vectores más cercanos a la regla de decisión lineal – Vectores Soporte)

$$f_i = \text{kernel}\ \left(\mathbf{x},\ \mathbf{l}^{(i)}\right) \quad = \quad e^{-\frac{\left\|\mathbf{x}-\ \mathbf{l}^{(i)}\right\|^2}{2\sigma^2}}$$

$$\mathbf{x} \rightarrow \mathbf{f} = [1\ f_1 \quad \cdots \quad f_m]$$

Vector de características transformado $\mathbf{f} =\in \Re^{m+1}$

$$h_\theta(\mathbf{x}) = (\boldsymbol{\theta}^T \cdot \mathbf{f} > 0)$$

Clasificación

$$J(\boldsymbol{\theta}) = \ C \sum_{i=1}^{m} \left[y^{(i)} \operatorname{cost}_1\left(\boldsymbol{\theta}^T \cdot \mathbf{f}^{(i)}\right) + \left(1 - y^{(i)}\right) \operatorname{cost}_0\left(\boldsymbol{\theta}^T \cdot \mathbf{f}^{(i)}\right)\right] + \frac{1}{2}\boldsymbol{\theta}^T \mathbf{M} \boldsymbol{\theta}$$

- **Kernel Polinómico**: $f_i = \text{kernel}\ \left(\mathbf{x},\ \mathbf{l}^{(i)}\right) \quad = \quad \left(\mathbf{x}^T \cdot\ \mathbf{l}^{(i)} + c\right)^n$
- …….

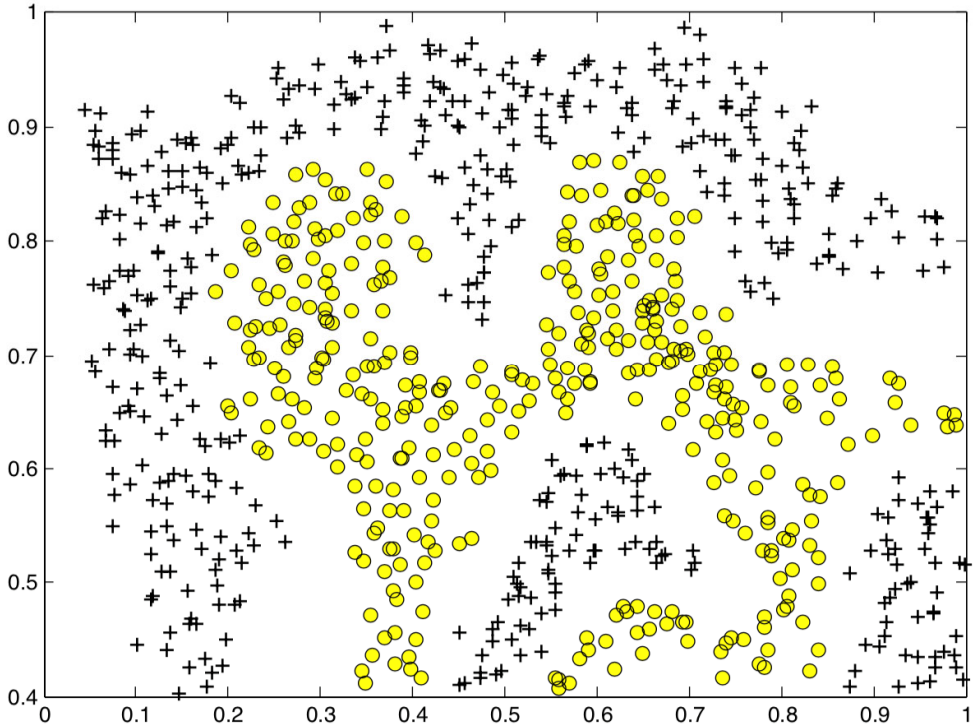
Máquinas de Vector Soporte (SVM)

- Ejemplo **Kernel Gaussiano**:

$$f_i = \text{kernel}\left(\mathbf{x},\ \mathbf{l}^{(i)}\right) = e^{-\frac{\left\|\mathbf{x}-\mathbf{l}^{(i)}\right\|^2}{2\sigma^2}}$$

$$\mathbf{x} \rightarrow \mathbf{f} = [1\ f_1 \quad \cdots \quad f_m]$$

$$h_\theta(\mathbf{x}) = (\boldsymbol{\theta}^T \cdot \mathbf{f} > 0)$$



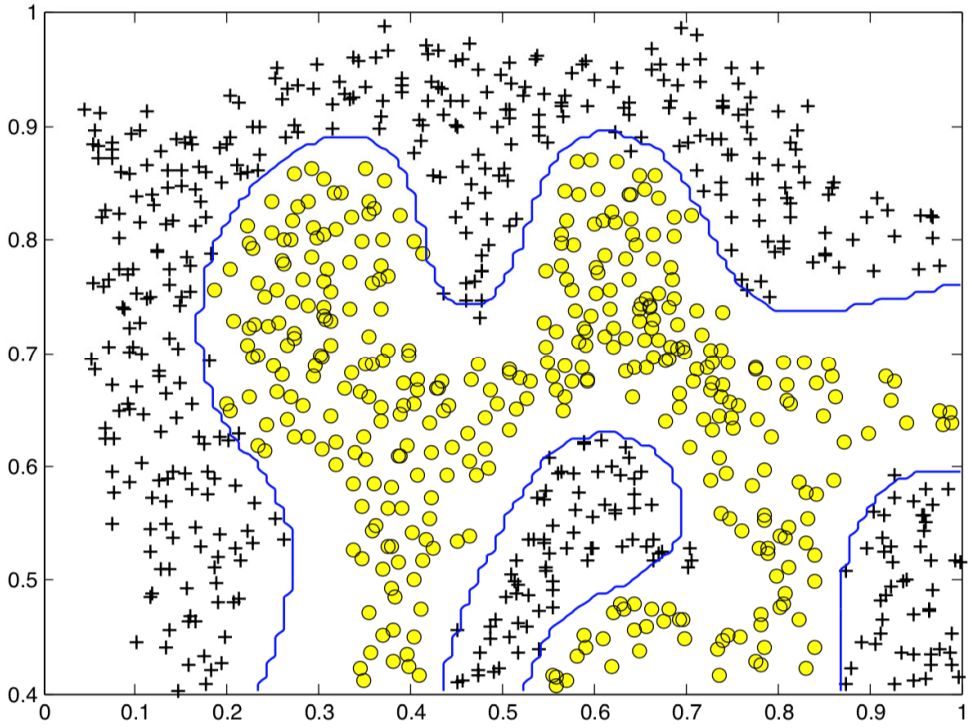


Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

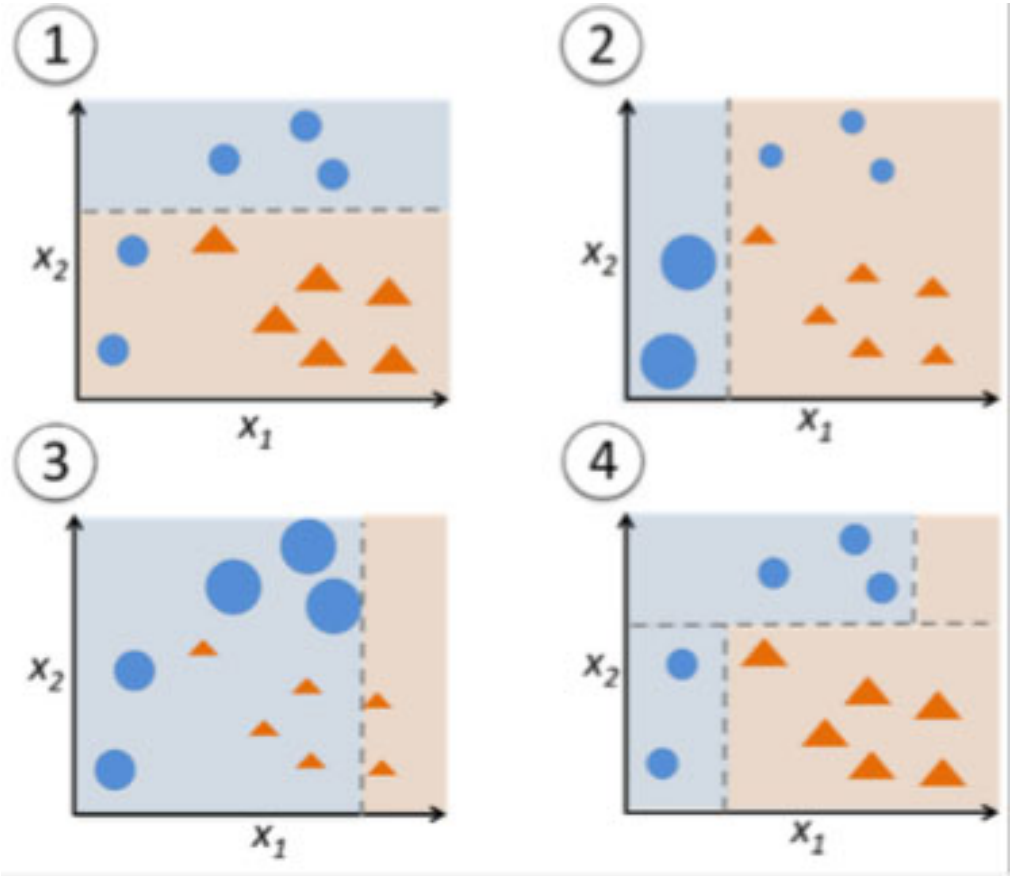
Otros Métodos de Reconocimiento

- Métodos de Aprendizaje Inductivo.
  - Adaboost
  - Vecino más cercano
  - Clustering (K-means – Mean-Shift)
  - Redes neuronales
  - Árboles de decisión
  - Listas de reglas

- Métodos Basados en Puntos Significativos (Bag of Words)

Algoritmo AdaBoost

- Clasificador fuerte a partir de varios clasificadores débiles
  - Se utiliza cuando el vector de descripción es muy grande
  - Permite algoritmos de clasificación sin problemas de overfitting limitando el número de características utilizadas.
- Fundamentos:
  - Uso de clasificadores simples: una sola característica
  - Cada clasificador se entrena dando un peso a cada ejemplo que se usa en las siguientes clasificaciones
    - Mayor peso a ejemplos mal clasificados
    - Menor peso a ejemplos bien clasificados
  - Se combinan los clasificadores débiles mediante un algoritmo de votación generando un clasificador fuerte



"A decision-theoretic generalization of on-line learning and an application to boostings", Freund, Y., Schapire, R.E Journal of Computer and System Sciences 55 (1), (1997)

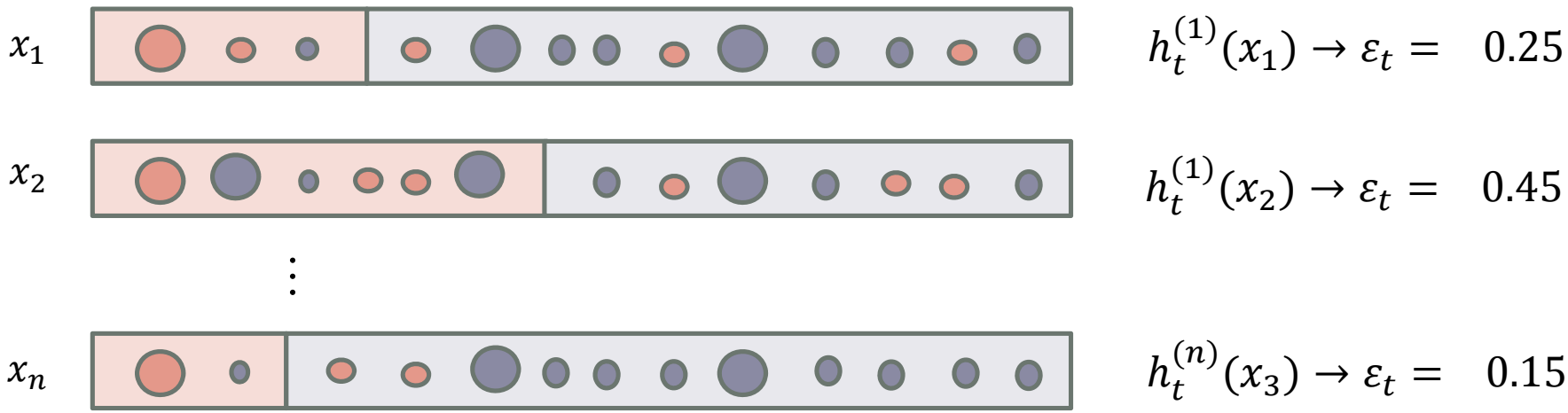
Algoritmo AdaBoost

- Clasificador débil
  - Datos de partida: vectores de descripción y etiquetas
$$\mathbf{X}_{mxn} = \left[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\right]^T \quad y^{(i)} \rightarrow \{-1,\ 1\}$$
  - Se inicializa el vector de pesos: $d_1^{(i)} = 1/m \quad | \quad i = 0, \cdots m$
  - Para la característica x_j Se selecciona el clasificador ($\boldsymbol{\theta}$) minimizando el error de clasificación ponderado por el vector de pesos anterior
$$\varepsilon_t = \sum_{i=1}^{m} d_t^{(i)} \cdot \cos t\left(h_t(x_j),\ y^{(i)}\right) \qquad h_t(x_j) \rightarrow \{-1,\ 1\}$$
  - Calcula la predicción con el clasificador débil y actualiza los pesos
$$\mathbf{D}_{t+1}^{(i)} = \frac{\mathbf{D}_t^{(i)} e^{-\alpha_t y^{(i)} h_t(x_j{}^{(i)})}}{Z_t} \qquad \alpha_{t+1} = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

- Clasificador fuerte:
  - Suma ponderada de cada una de las clasificaciones débiles en función del error de clasificación
$$H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$
  - Se continúa añadiendo clasificadores débiles hasta que el error de clasificación (validación cruzada) es bajo

Algoritmo AdaBoost

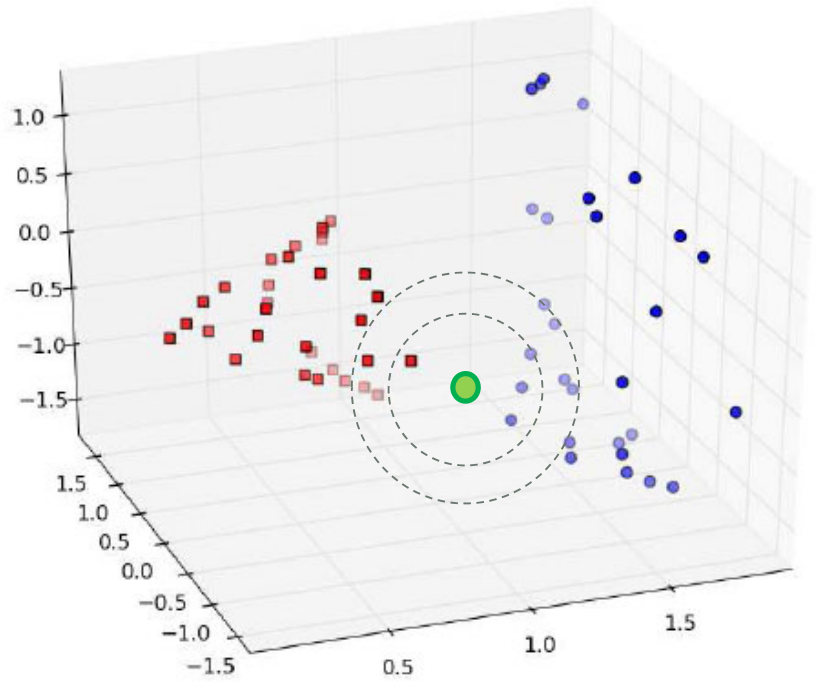
- Selección de Características
  - En cada iteración del proceso de aprendizaje, se entrena un clasificador débil para cada una de las características
  - Se selecciona el clasificador débil con error mínimo



- Aplicación
  - Clasificador utilizado en el reconocimiento de caras mediante descriptores Haar
  - Se complementa con un esquema de clasificación en cascada para diferentes ventanas en la imagen para localizar la posición de la cara.
    - Múltiples clasificadores fuertes con pocas características, es preciso que todos los clasificadores en cascada detecten la cara
    - Los clasificadores se entrenan en cascada de forma especializada con un objetivo de rendimiento

Vecino más Cercano

- Propiedades:
  - Aprendizaje: no construye un modelo (almacena todas las instancias)
  - Clasificación: una nueva instancia se clasifica en función de la clase de las instancias almacenadas más cercanas:
    - Algoritmo K-NN: clase mayoritaria en las k-instancias más cercanas



Vecino más Cercano

- Cálculo de distancias:
  - La distancia entre dos instancias se calcula a partir del valor de sus atributos.
  - Distancia Euclídea:
    - Precisa que el vector de características esté normalizado
    - En otro caso, los atributos con mayor valor absoluto siempre prevalecen sobre los demás.

$$d_e(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(x_{11} - x_{21})^2 + \cdots + (x_{1n} - x_{2n})^2}$$

  - Distancia Mahalanobis: distancias normalizadas según varianza

$$d_m(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\hat{x}_{11} - \hat{x}_{21})^2 + \cdots + (\hat{x}_{1n} - \hat{x}_{2n})^2} \qquad \hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

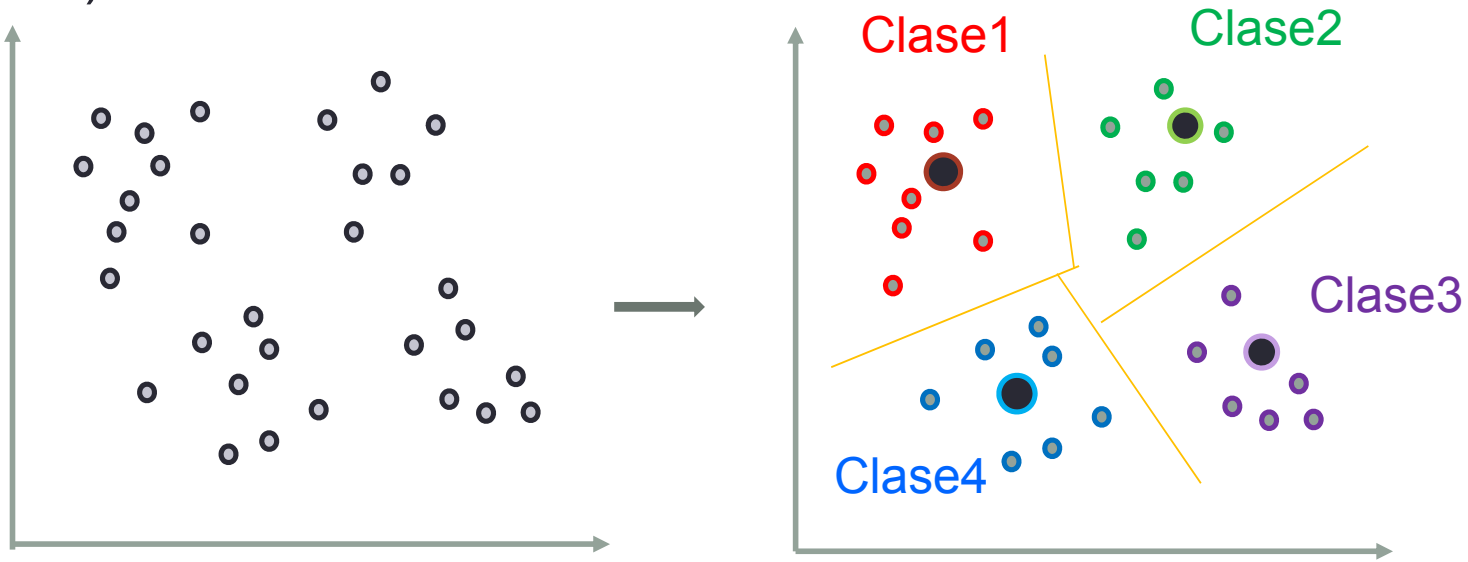
$$d_m(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\left(\frac{x_{11} - x_{21}}{\sigma_1}\right)^2 + \cdots + \left(\frac{x_{1n} - x_{2n}}{\sigma_n}\right)^2}$$

  - Distancia Coseno: ángulo entre los vectores

$$d_{cos}(\mathbf{x}_1, \mathbf{x}_2) = 1 - \cos(\mathbf{x}_1, \mathbf{x}_2) = 1 - \frac{\mathbf{x}_1^T \cdot \mathbf{x}_2}{|\mathbf{x}_1| \cdot |\mathbf{x}_2|}$$

Clustering: K-medias (‘*K-means*’)

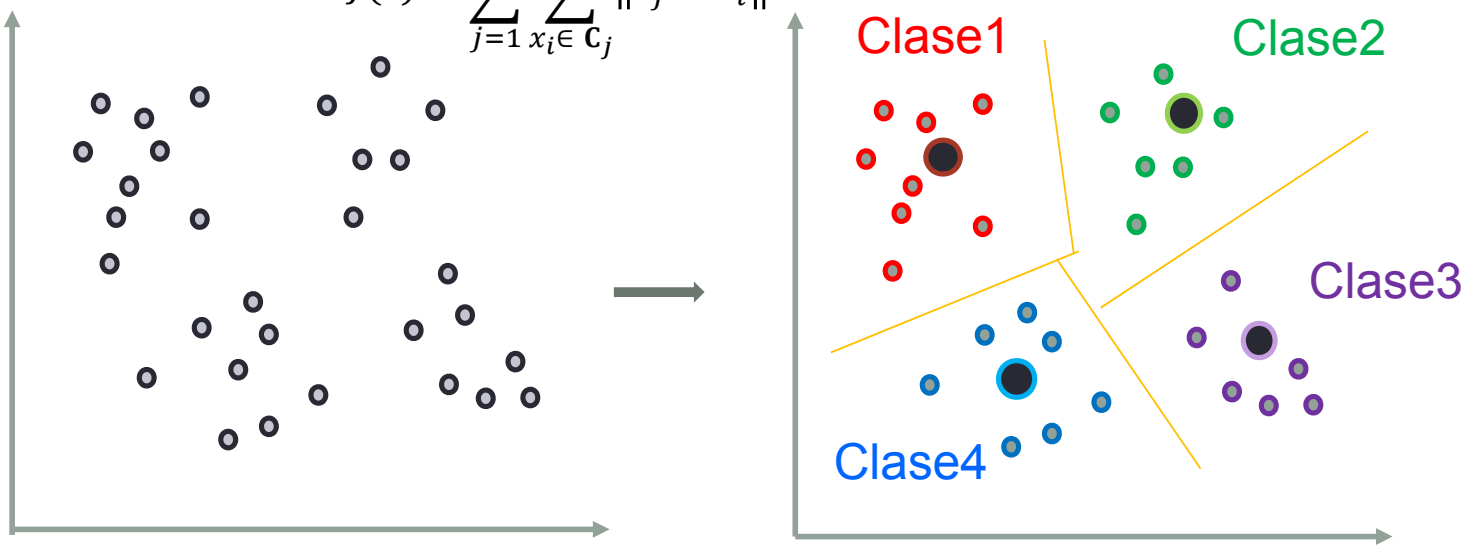
- Aprendizaje no supervisado
  - Datos de entrada: m-vectores de características (datos entrenamiento)
  - Parámetro de entrada: número de clases (clústeres) ***K***
  - Agrupación que minimiza la distancia respecto a la media
  - Resultado: medias de cada agrupación
  - La asignación a cada clúster se realiza como el clúster vecino (media) más cercano



Clustering: K-medias (‘*K-means*’)

- Algoritmo iterativo
  1. Se seleccionan aleatoriamente K vectores de los datos de entrenamiento y se toman como representantes de cada clúster: $\{c_1, c_2, \cdots c_k\}$
  2. Se asigna cada muestra al clúster más cercano:
  $$\mathbf{C}_j = \left\{ x_i \perp d(x_i, c_j) = \min_r d(x_i, c_r) \right\}$$
  3. Se modifican el representante de cada clase como la media de las muestras asignadas al clúster: $$c_j = \frac{1}{n_{\mathbf{C}_j}} \sum_{x_i \in \mathbf{C}_j} x_i$$
  4. Volver al paso 2 hasta que no haya cambios de asignación
- El resultado depende de la selección inicial de clústeres:
  - Probar varias y quedarse con el que genere la mínima suma de distancias a las medias

$$J(\mathbf{c}) = \sum_{j=1}^{k} \sum_{x_i \in \mathbf{C}_j} \left\| \mathbf{c}_j - \mathbf{x}_i \right\|^2$$

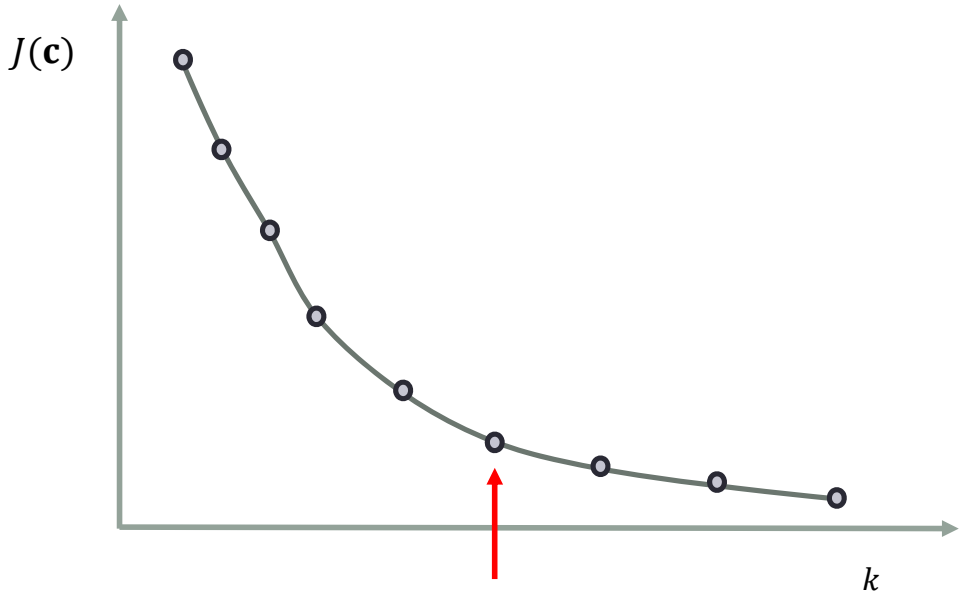


Clustering: K-medias (‘*K-means*’)

- Elección del valor de k
  - Cuanto mayor es el valor de k menor es la distancia a la media (***Heterogeneidad***)

$$J(\mathbf{c}) = \sum_{j=1}^{k} \sum_{x_i \in \mathbf{c}_j} \left\| \mathbf{c}_j - \mathbf{x}_i \right\|^2$$

  - Regla heurística:
    - Regla del codo: tomar el valor de k cuando la diferencia en Heterogeneidad empieza a ser pequeña (codo de la curva)



Clustering: Mean-Shift

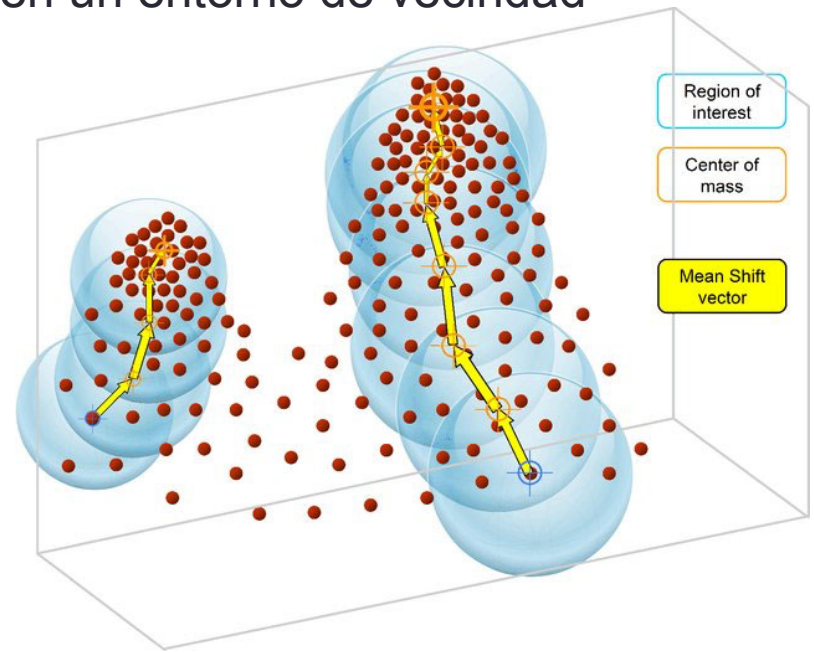
- Busca los puntos de máximo gradiente de densidad
  - Se define:
    - Un kernel de distancia (el más común es el kernel Gaussiano)

$$K\left(x^{(i)},\mathbf{x}\right) \quad = \quad e^{-\frac{\left\|x^{(i)}-\mathbf{x}\right\|^2}{2\sigma^2}}$$

    - Media ponderada (kernel) de la densidad en un entorno de vecindad $N(\mathbf{x})$:

$$m(\mathbf{x}) = \frac{\sum_{x^{(i)}\in N(x)} K\left(x^{(i)},\mathbf{x}\right)\cdot x^{(i)}}{\sum_{x^{(i)}\in N(x)} K\left(x^{(i)},\mathbf{x}\right)}$$

$$Mean-Shift = \; m(\mathbf{x}) - \mathbf{x}$$

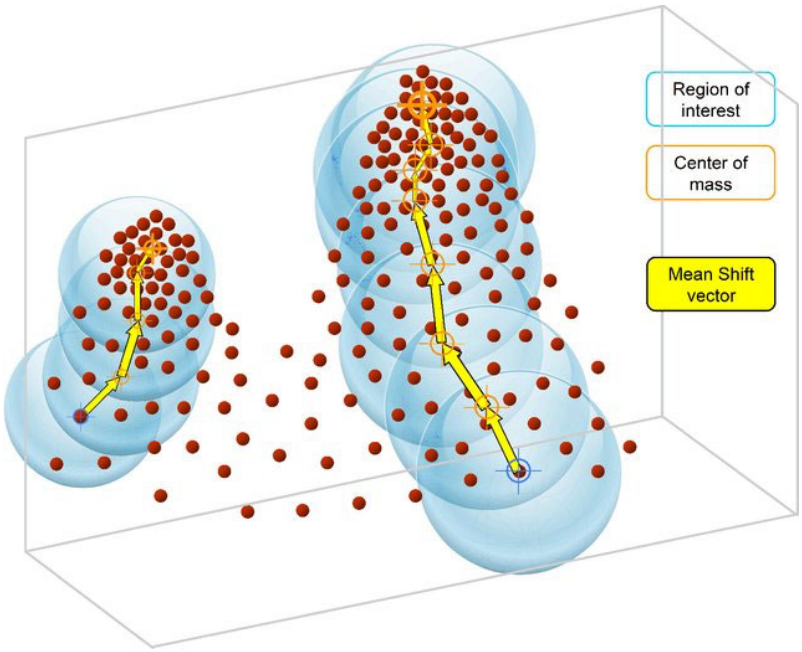


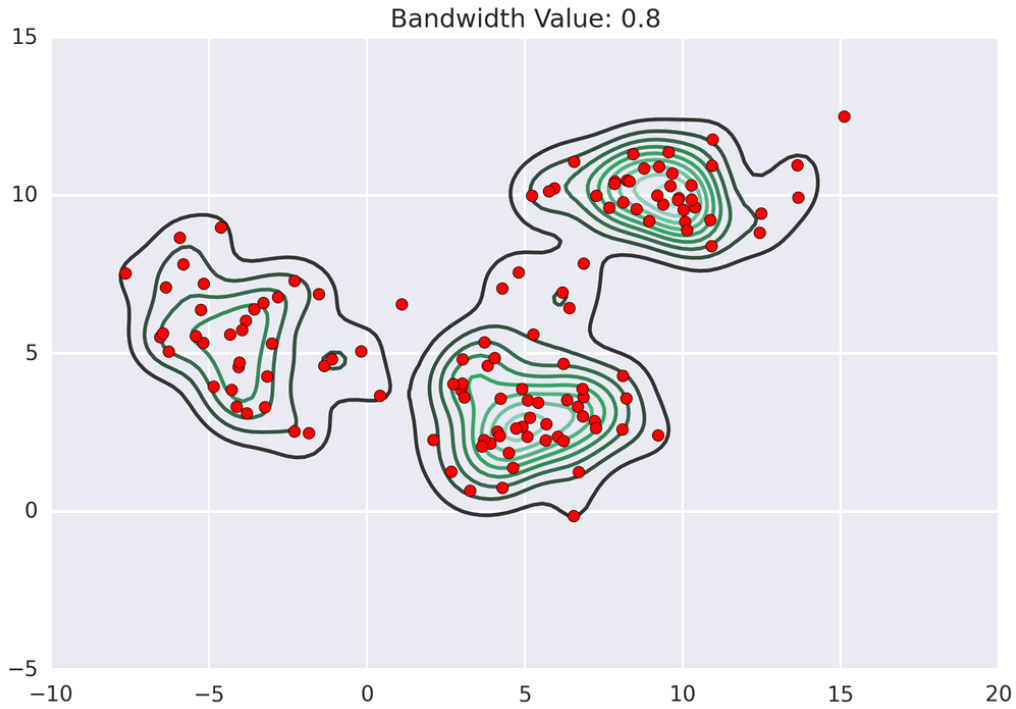
"Mean Shift, Mode Seeking, and Clustering". Cheng, Yizong
IEEE Transactions on Pattern Analysis and Machine Intelligence. 17 (8): 790–799) (August 1995)

"The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition" Fukunaga, Keinosuke
Larry D. Hostetler IEEE Transactions on Information Theory. 21 (1): 32–40. (January 1975)

Clustering: Mean-Shift

- Busca los puntos de máximo gradiente de densidad
  - Proceso iterativo:
    - Desplaza cada punto a su media en el entorno de vecindad: $m(\mathbf{x})$
  - Resultado:
    - Centros de máxima densidad



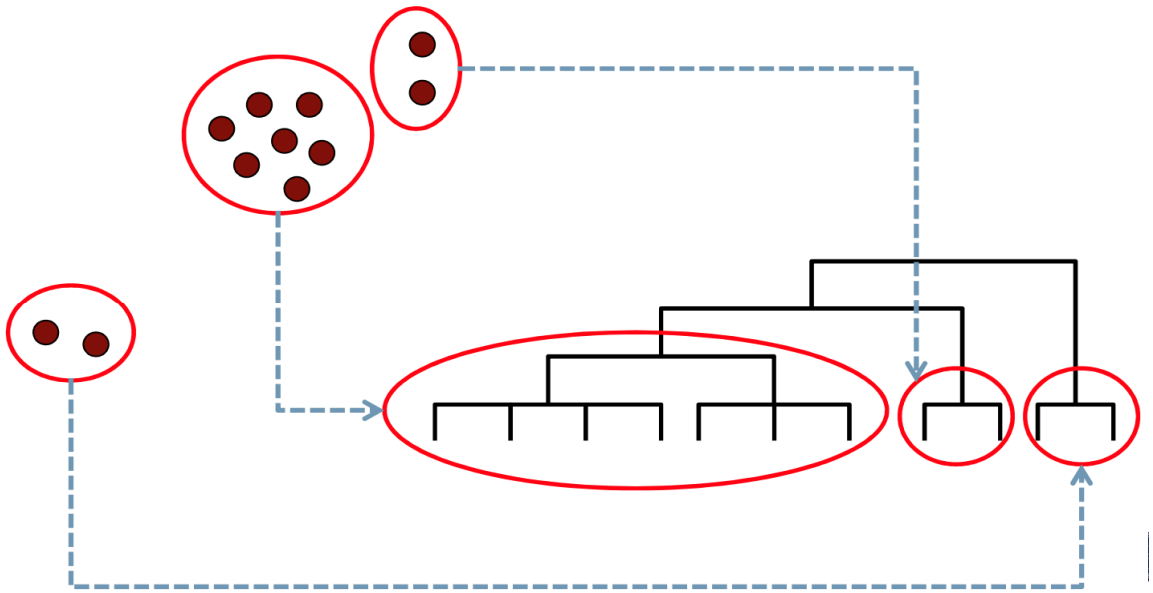


"Mean Shift, Mode Seeking, and Clustering". Cheng, Yizong
IEEE Transactions on Pattern Analysis and Machine Intelligence. 17 (8): 790–799) (August 1995)

Clustering Jerárquico

- Clustering Jerárquico: (AGNES: AGglomerative NESting)
  - Los métodos previos sólo son validos cuando el descriptor es escalar.
  - Si el descriptor es **binario** (ORB) tenemos una secuencia de '1's y '0's $\mathbf{x} = [01010 \cdots 101]^T$
    No tiene sentido calcular calcular medias o centroides.
  - La distancia Euclídea se sustituye por la distancia **Hamming**: (número de bits distintos)

- Etapas:
  - Se realiza en primer lugar un proceso de agrupamiento: agregación de muestras por distancia
  - Se genera un árbol ***dendrograma*** con las agregaciones realizadas en el paso previo
  - Se seleccionan los clusters desde la raíz del árbol



"Finding Groups in Data: An Introduction to Cluster Analysis". Kaufman, L. and Rousseeuw, P.J. (Wiley 1990)

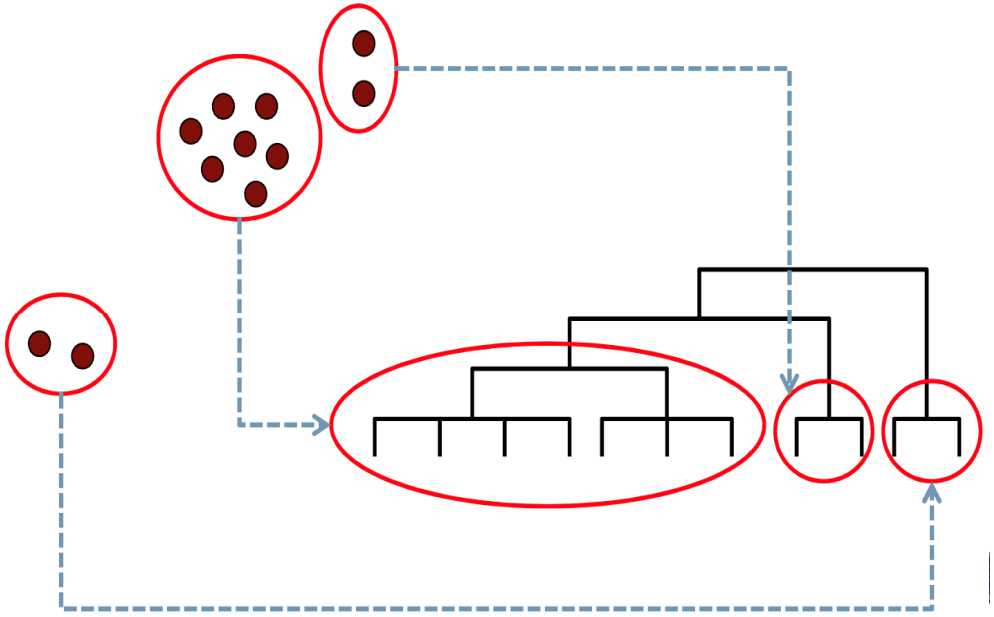
Clustering Jerárquico

- Clustering Jerárquico: (AGNES: AGglomerative NESting)

- Algoritmo:
  1. Se construye la matriz de distancias para cada par de datos
  2. Se parte de tantos clusters como datos
  3. Elegir el par de clusters/datos más cercanos (mínima distancia en la matriz D_{nxn}) y juntarlos en un solo clúster.
  4. Recalcular la distancia del nuevo clúster al resto de clusters (distancia media)
  5. Eliminar de la matriz D_{nxn} las filas y columnas de los dos datos/clusters agregados y sustituirlas por los del nuevo cluster $D_{nxn} \rightarrow D'_{n-1xn-1}$
  6. Volver al paso 3 subiendo de nivel hasta que tengamos un solo clúster

  7. Se crea un ***dendrograma***: grafo del árbol revelando las agrupaciones.
  8. Se seleccionan los clusters desde la raíz del árbol

$$D_{nxn} = \begin{bmatrix} d_{11} & \dots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \dots & d_{nn} \end{bmatrix}$$



"Finding Groups in Data: An Introduction to Cluster Analysis". Kaufman, L. and Rousseeuw, P.J. (Wiley 1990)

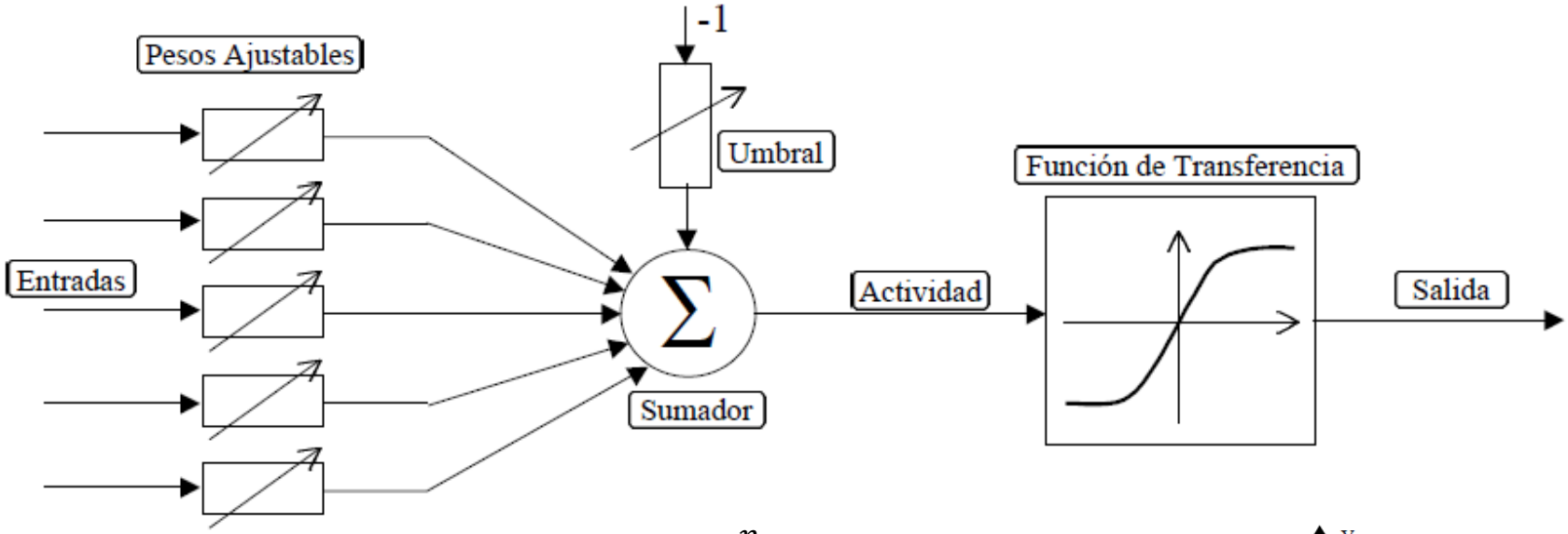
Redes Neuronales

- Definición:
  - Una red neuronal es una estructura de procesamiento de información paralela y distribuida, que intenta emular las funciones computacionales elementales de la red nerviosa del cerebro humano, en base a la interconexión de multitud de elementos de procesamiento, cada uno de los cuales presenta un comportamiento completamente local

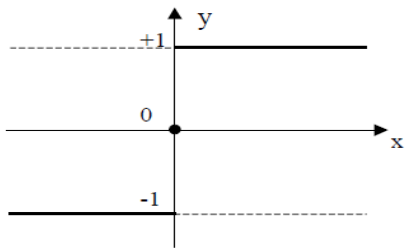
- Fundamentos:
  - Mecanismo o procedimiento matemático que transforma un conjunto de variables numéricas (entradas) en un conjunto de variables numéricas (salidas).
  - Imitan la estructura neuronal del cerebro y desarrollan un equivalente algorítmico de los procesos de reconocimiento y aprendizaje

Redes Neuronales

- Modelo de funcionamiento de una neurona:



- Salida de la neurona: $a = f\left(\sum_{i=1}^{n} \omega_i\, x_i - \theta\right)$
- Actividad interna: $z = \sum_{i=1}^{n} \omega_i\, x_i - \theta$
- Umbral: θ
- Pesos sinápticos (memoria): ω_i
- Función de transferencia: sigmoide(a), signo(a),…



$$\mathrm{sgn}(z) = \begin{cases} +1 & si \quad z > 0 \\ 0 & si \quad z = 0 \\ -1 & si \quad z < 0 \end{cases}$$

$$a^{(k)} = f\left(z^{(k)}\right) = f\left(\mathbf{W}^{(k-1)} \cdot a^{(k-1)}\right)$$

$$\mathrm{sigmoide}(z) = \frac{2}{1 + e^{-z}} - 1$$

Redes Neuronales

- Red de neuronas:
  - La función que cada neurona desarrolla en la red es un elemento básico de procesamiento.
  - Es la ***cooperación*** de las neuronas la que produce las respuestas complejas
  - Las neuronas de la red se agrupan en capas, recibiendo la misma información que procesan de manera independiente. Sistema masivamente ***paralelo***.

  - Memoria a corto plazo: Nivel de actividad de cada neurona.
  - Memoria a largo plazo: Valores de los pesos sinápticos

Redes Neuronales

- Aprendizaje
  - La red neuronal no se programa, sino que se entrena
  - Solución para problemas complejos difícilmente programables
    - Multitud de elementos muy simples interconectados.
    - Procesamiento no lineal.
  - Se denomina aprendizaje o entrenamiento de la red al ajuste de los pesos sinápticos, que determina el grado de conexión entre las neuronas de la red.
  - Se distinguen dos tipos de aprendizaje:
    - **Supervisado**: Típico de las redes concebidas para el ajuste de datos o la clasificación de objetos etiquetados
    - **No-Supervisado**: Redes en las que los datos no tienen a priori ninguna clasificación (Redes auto-organizativas)
  - El número de capas y neuronas de la red se selecciona tras una serie de ensayos y observando sus resultados.

Redes Neuronales (Tipos)

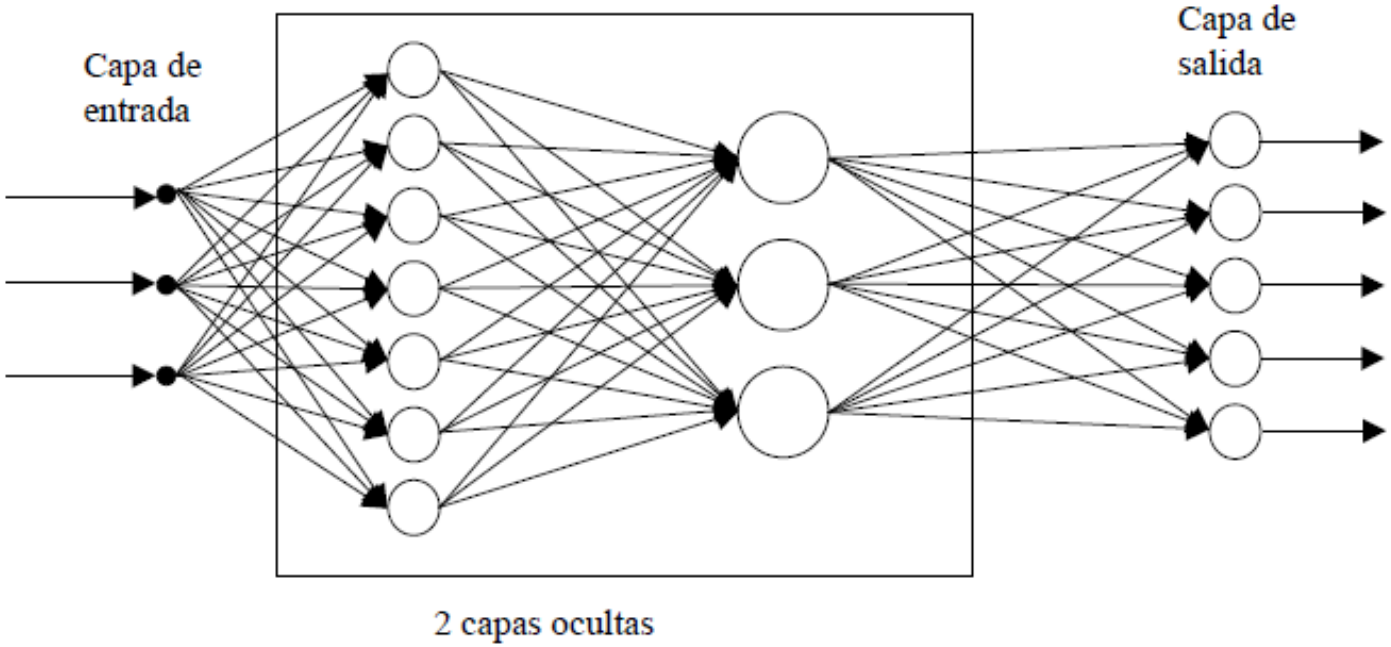
- **Redes Unidireccionales** (*Feedforward Networks*)
  - Usadas para transformar un conjunto de datos especificado en otro también especificado.
  - El aprendizaje es supervisado y tiene lugar a través de un proceso de ajuste de los pesos sinápticos de las neuronas de la red.
  - Su arquitectura típica es de una red multicapa.
  - Prototipo: Perceptrón multicapa (MLP)
- **Redes Recurrentes** (*Feedback Networks*)
  - Concebidas para almacenar eficientemente información (**memoria asociativa**)
  - La red funciona como un sistema dinámico cuyos puntos de equilibrio representan los registros almacenados.
  - Su arquitectura típica es la de una red monocapa con una gran realimentación.
  - Prototipo: Red de Hopfield.
- **Redes Auto-Organizadas** (Self-Organizing Networks)
  - Uso: Estructurar la información que se presenta a la red en conjuntos a priori desconocidos (**clustering**)
  - El entrenamiento es mediante un aprendizaje no supervisado denominado competitivo.
  - Cada neurona tiene un cierto grado de conexión con neuronas colaterales.

Redes Neuronales

- Aplicaciones:
  - Clasificación:
    - Decidir a qué clase, de una serie dada, se asigna un dato de entrada a la red.
  - Asociación:
    - La red funciona como un proceso de recuperación de un dato a partir de una entrada relacionada con el dato almacenado.
  - Agrupamiento (Clustering):
    - Utilizada cuando no se conoce una clasificación de los datos de entrada, y se espera que la red genere esta clasificación.
  - Optimización:
    - Permite solucionar problemas de optimización de naturaleza complicada.
  - Predicción y Control:
    - En tareas de predicción para anticipar el estado futuro de un sistema.
    - Tareas de control de sistemas dinámicos complejos.

Redes Neuronales

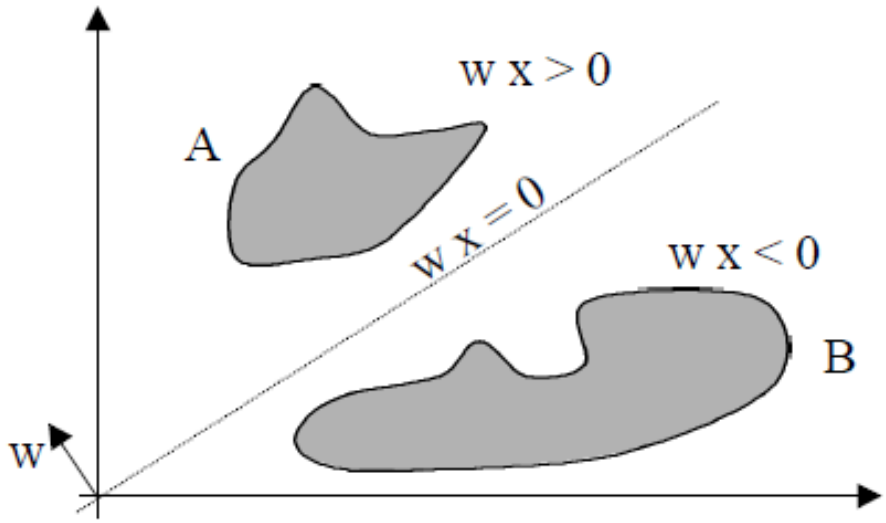
- Perceptrón Multicapa aplicado a reconocimiento
  - Red multicapa.
    - *Capa de Entrada*: vector de descripción
    - *Capa de Salida*: clases de objetos
    - *Capas Ocultas*: capas intermedias (memoria)
  - Las neuronas de cada capa reciben información de las de la capa precedente y la transmiten a la capa siguiente.
  - Las neuronas no están interconectadas dentro de cada capa de la red.
  - El perceptrón más común incluye sólo una capa oculta.



Redes Neuronales

- Perceptron Simple:
  - Una sola capa (no hay capas ocultas), reciben todas las neuronas las mismas entradas
  - Se usa como función de transferencia de la neurona la función signo (+1,0, -1)
  - La utilización del perceptrón simple como función discriminante, equivale a utilizar formas lineales como funciones discriminantes
  - Matemáticamente el aprendizaje del perceptrón consiste en la determinación de un vector $\boldsymbol{w}$ que discrimine entre las clases de salida

- Dos capas:
  - No lineales, Conjunto convexo
- Tres Capas:
  - Conjuntos generales



Redes Neuronales

- Operación:
  - Aprendizaje (off-line)
    - Estimación de los pesos $\boldsymbol{w}_i$ de cada neurona
    - Aplicamos un conjunto de patrones de entrenamiento a la entrada, de clases conocidas
    - Algoritmos de optimización para la estimación de los pesos que maximicen la respuesta en las neuronas de salida de cada clase
      - *Algoritmo iterativo* ***Back Propagation***: retro-propagación del error para cada ejemplo k

$$\delta_k{}^{(S)} = a_k{}^{(S)} - y_k$$

$$\delta_k{}^{(l)} = \left(\mathbf{W}^{(l)}\right)^T \cdot \delta_k{}^{(l+1)} \cdot\!* \, f'\left(z_k{}^{(l)}\right)$$

$$\Delta^{(l)} = \Delta^{(l)} + \delta_k{}^{(l+1)}\left(a_k{}^{(l)}\right)^T$$

$$\frac{\partial}{\partial \omega_{ij}^{(l)}} J(\mathbf{W}) = \frac{1}{m}\Delta_{ij}^{(l)} + \frac{\lambda}{m}\omega_{ij}^{(l)}$$

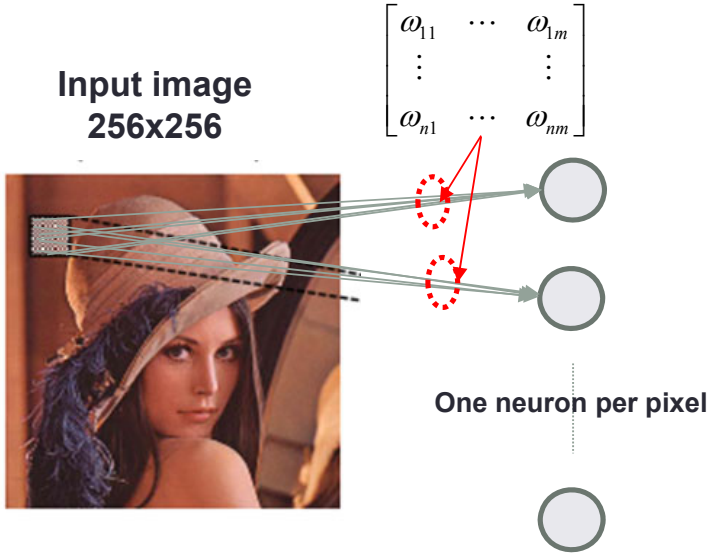
$$\omega_{ij}{}^{(l)} = \omega_{ij}{}^{(l)} - \alpha \frac{\partial}{\partial \omega_{ij}^{(l)}} J(\mathbf{W})$$

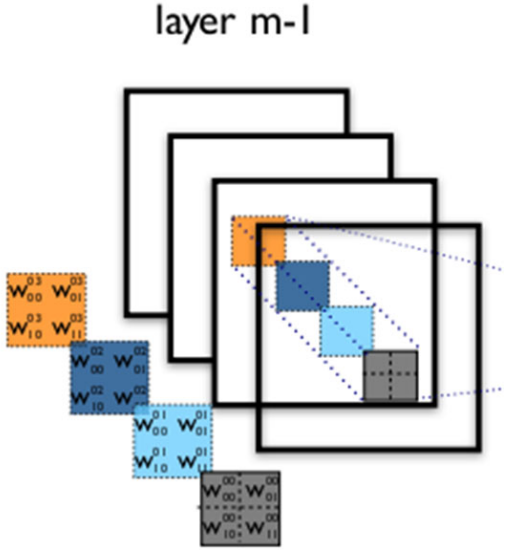
  - Clasificación (on-line)
    - Aplicar a la entrada un vector de un nuevo patrón
    - Ejecutar la propagación por las neuronas (***Forward Propagation***)
    - La salida más alta indica el objeto clasificado
    - Rápido

$$a^{(k)} = f\left(z^{(k)}\right) = f\left(\mathbf{W}^{(k-1)} \cdot a^{(k-1)}\right)$$

Redes Neuronales Convolucionales (CNN)

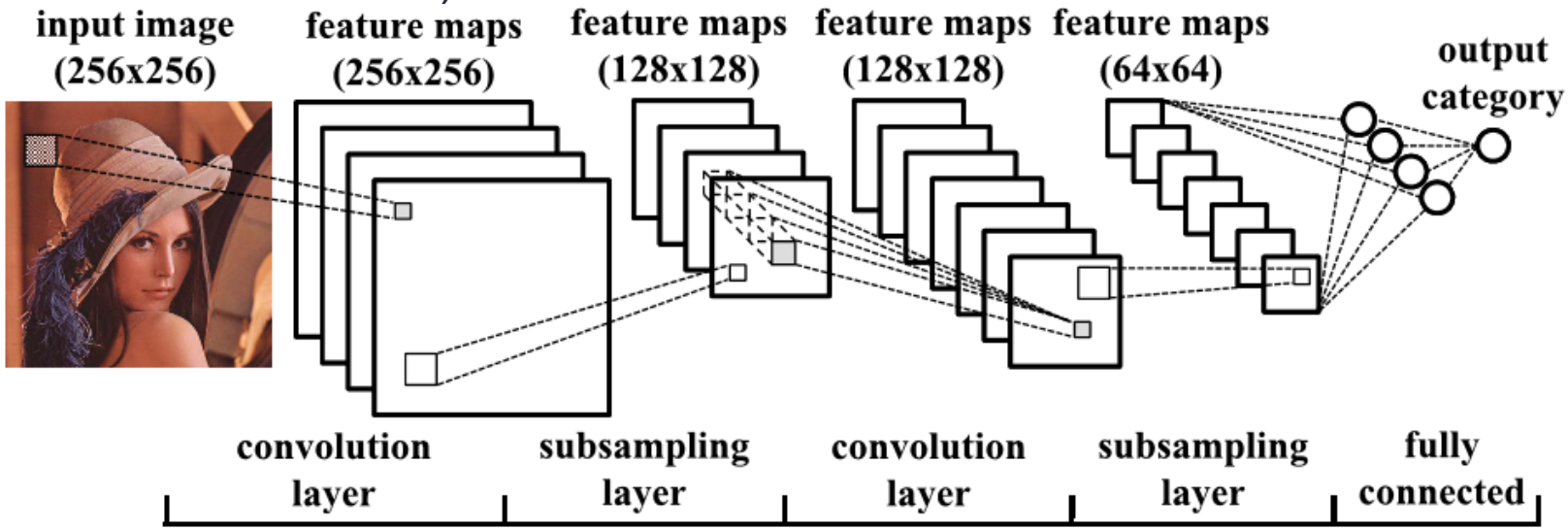
- Clasificación de imágenes sin usar descriptores:
  - La capa de entrada es la propia imagen
  - Segunda capa: una neurona por cada pixel de la imagen
    - En una Red Neuronal tradicional (MLP) todas las neuronas utilizarían toda la imagen (*explosión de parámetros*)
    - En una Red Neuronal Convolucional (CNN) cada neurona actúa sobre una ventana (entorno de vecindad de cada pixel de la imagen)
    - Los pesos en todas las neuronas en la misma capa son iguales
    - En cada capa hay varios niveles que realizan filtrados diferentes
  - Cada nivel actúa como un filtro convolucional extrayendo características (la red aprende su propias características)





Redes Neuronales Convolucionales (CNN)

- Implementación (LeNET CNN):
  - La capa de entrada es la propia imagen
  - Cada neurona actúa sobre una ventana distinta de la imagen.
  - Cada capa esta formada por varios niveles (agrupación de neuronas)
  - Las neuronas dentro del mismo nivel utilizan los mismo pesos (mismo filtrado)
  - Diferentes niveles de una capa realizan filtrados diferentes.
  - Se incorporan capas de submuestreo ('*max-pooling*') para reducir al dimensión de la red neuronal (máximo de la ventana)
  - Las capas de salida actúan como una capa normal de clasificación (MLP) (totalmente conectada)



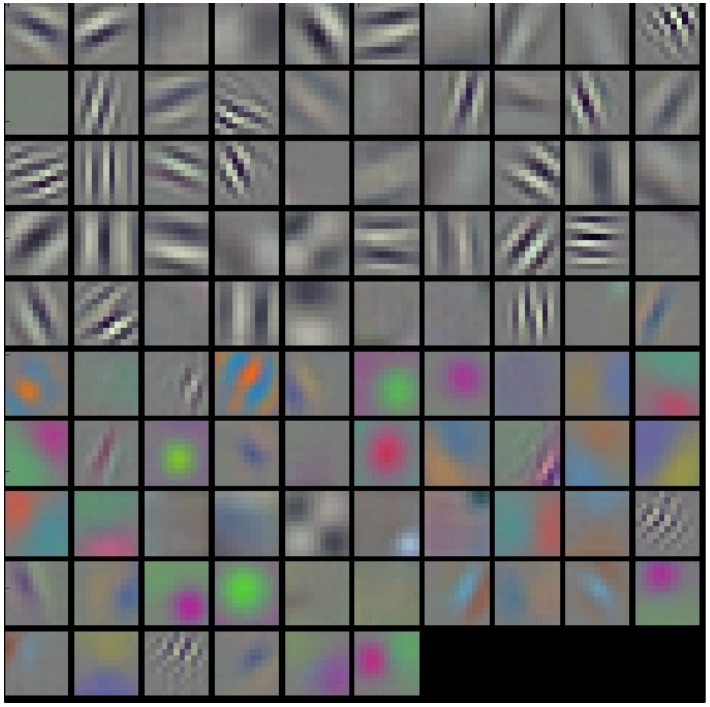
"Convolutional networks for images, speech, and time-series" Y. LeCun and Y. Bengio, The Handbook of Brain Theory and Neural Networks. MIT Press (1995)

LeNet CNN

Redes Neuronales Convolucionales (CNN)

- Propiedades:
  - El número de parámetros de la red crece exponencialmente
  - El ajuste suele presentar problemas de overfitting y requiere de un conjunto de entrenamiento muy grande
  - Los pesos de las capas de extracción de características se pueden representar como filtros de imagen:

Primera capa de convolución en AlexNet



27 capas, filtros 3x3, 7x7
12M parámetros

19 capas, filtros 3x3
144M parámetros

7 capas, filtros 3x3, 5x5
60M parámetros

Image classification

Year	Codename	Error (percent)	99.9% Conf Int
2014	**GoogLeNet**	**6.66**	**6.40 - 6.92**
2014	VGG	7.32	7.05 - 7.60
2014	MSRA	8.06	7.78 - 8.34
2014	AHoward	8.11	7.83 - 8.39
2014	DeeperVision	[illegible]	[illegible]
2013	Clarifai†	[illegible]	[illegible]
2014	CASIAWS†	[illegible]	[illegible]
2014	Trimps†	11.46	11.13 - 11.80
2014	Adobe†	11.58	11.25 - 11.91
2013	**Clarifai**	**11.74**	**11.41 - 12.08**
2013	NUS	12.95	12.60 - 13.30
2013	ZF	[illegible]	[illegible]
2013	AHoward	[illegible]	[illegible]
2013	OverFeat	[illegible]	[illegible]
2014	Orange†	14.80	14.43 - 15.17
2012	SuperVision†	15.32	14.94 - 15.69
2012	**SuperVision**	**16.42**	**16.04 - 16.80**
2012	ISI	26.17	25.71 - 26.65
2012	VGG	26.98	26.53 - 27.43
2012	XRCE	27.06	26.60 - 27.52
2012	UvA	29.58	29.09 - 30.04

ImageNet Large Scale Visual Recognition Challenge
http://image-net.org/challenges/LSVRC/

"ImageNet Classification with Deep Convolution" Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, lournal of Computer and System Sciences 55 (1), (1997)

Redes Neuronales Convolucionales (CNN)

- YOLO CNN Classifier

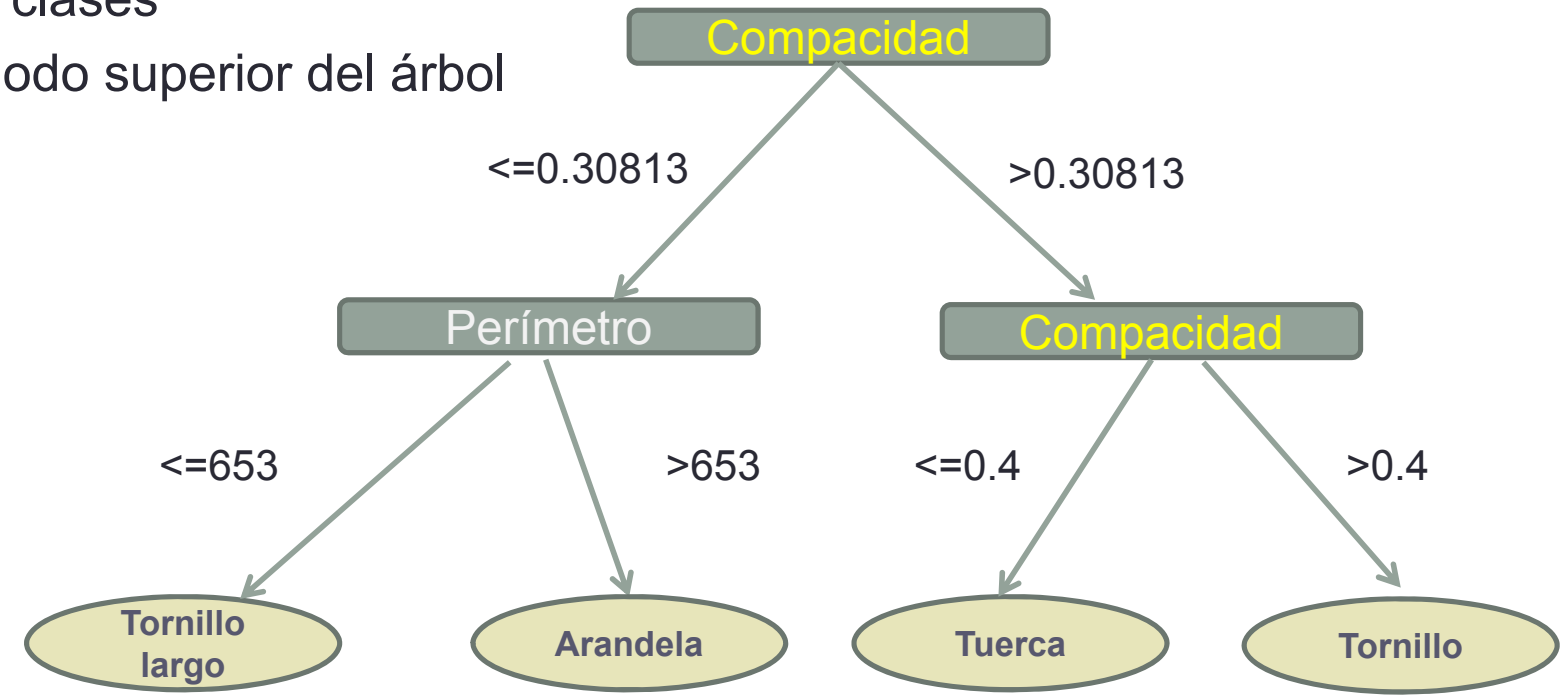
Video

"You Only Look Once: Unified, Real-Time Object Detection"
Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi CVPR (2016)

YOLO9000: Better, Faster, Stronger
Joseph Redmon, Ali Farhadi CVPR (2017)

Árboles de Decisión

- Estructura:
  - Representan funciones lógicas (if-then-else)
  - Compuestos de nodos y ramas
  - Nodos internos = atributos (medidas)
  - Nodos hoja = clases
  - Nodo raíz = nodo superior del árbol



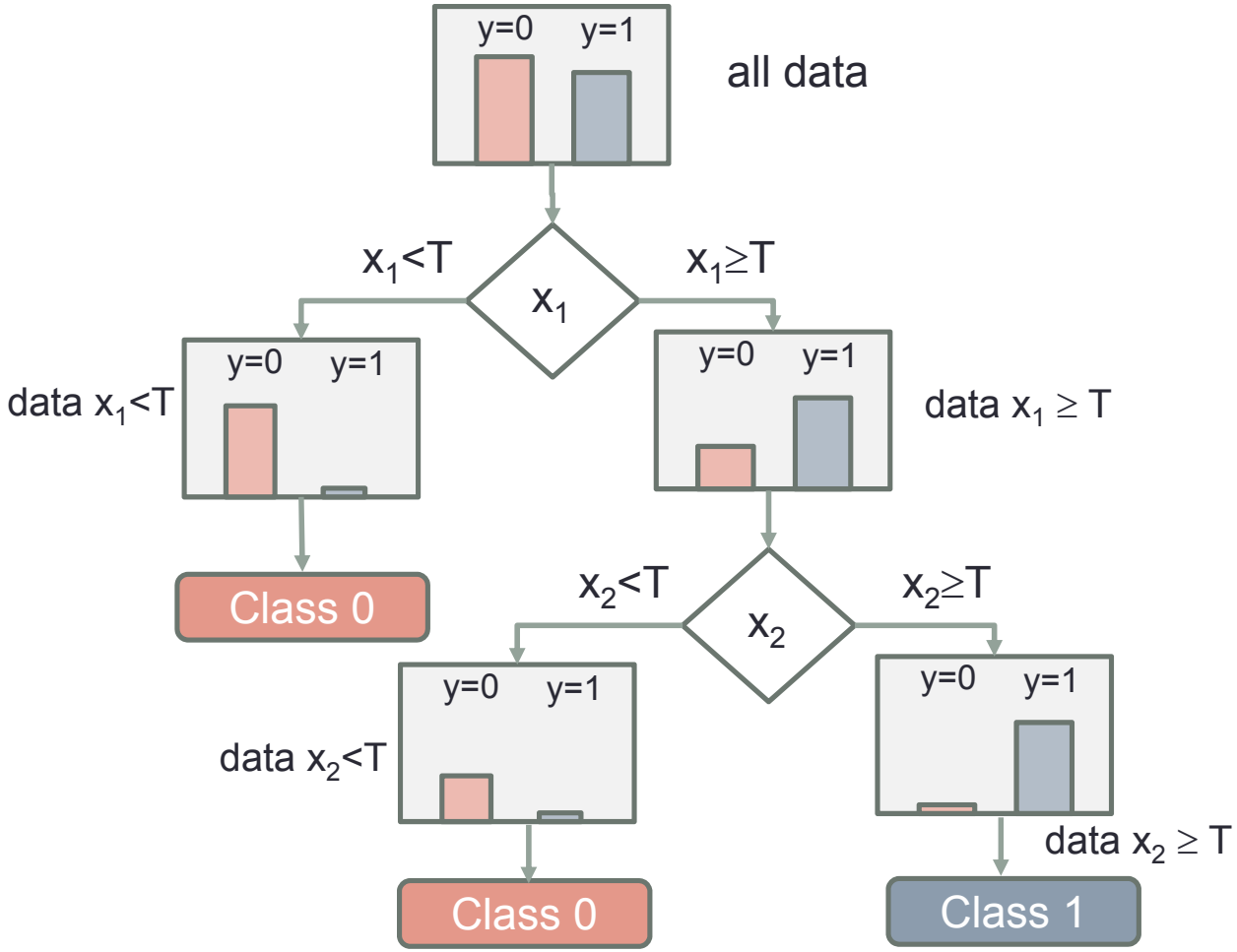
  - Objetivo en aprendizaje automático: inferir un árbol de decisión a partir de un conjunto de instancias o ejemplos

Árboles de Decisión

- Múltiples formas de inferir el árbol:
  - ***Trivial***: se crea una ruta del árbol por cada instancia de entrenamiento.
    - Árboles excesivamente grandes.
    - No funcionan bien con instancias nuevas.
  - ***Óptimo***: el árbol más pequeño posible compatible con todas las instancias.
    - Inviable computacionalmente.
  - ***Pseudo-optimo*** (*Heurístico*): selección del atributo en cada nivel del árbol en función de la calidad de la división que produce.
    - Los principales programas de generación de árboles utilizan procedimientos similares (C4.5, CART, etc).

Árboles de Decisión

- Formas de inferir el árbol:
  - ***Greedy algorithm*** : Se clasifica por votación de los datos de entrenamiento. Selección del atributo en cada nivel del árbol en función de la calidad de la división que produce (error de clasificación).



1. Se comienza con un árbol vacío
2. Se selecciona una nueva característica
3. Se establece una regla de decisión (umbral) y se divide el conjunto de datos según la clasificación
4. Si el error de clasificación es pequeño finalizar la rama con una predicción
5. Si el error de clasificación es grande volver al punto 2

Lista de Reglas

- Modelo similar a un árbol de decisión
- Reglas más complejas
- Tipos de reglas:
  - Reglas de clasificación
    - Una combinación lógica de los valores de los atributos sirve para predecir la clase
  - Reglas de asociación
    - Una combinación lógica de los valores de los atributos sirve para predecir el valor de otro atributo
  - Reglas relacionales
    - Una condición puede combinar los valores de varios atributos
    - Permiten expresar relaciones más complejas

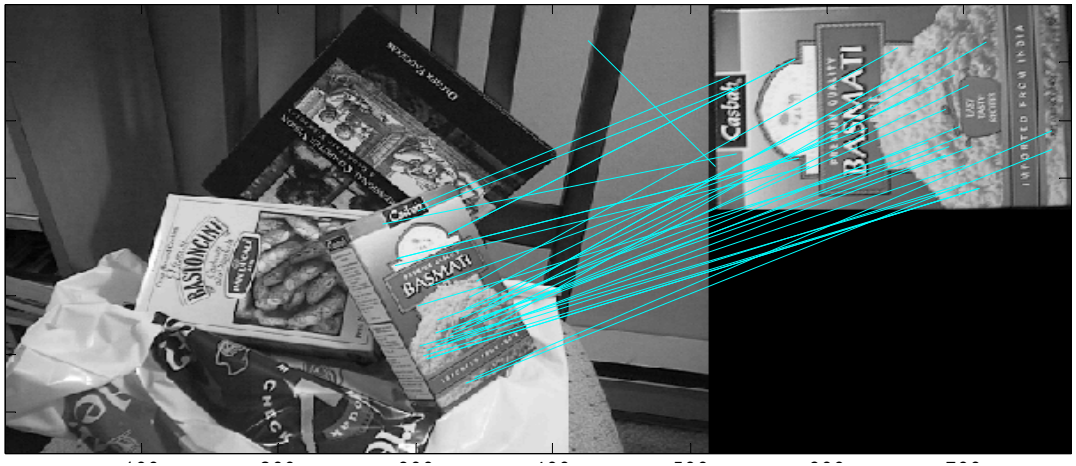
Otros Métodos de Reconocimiento

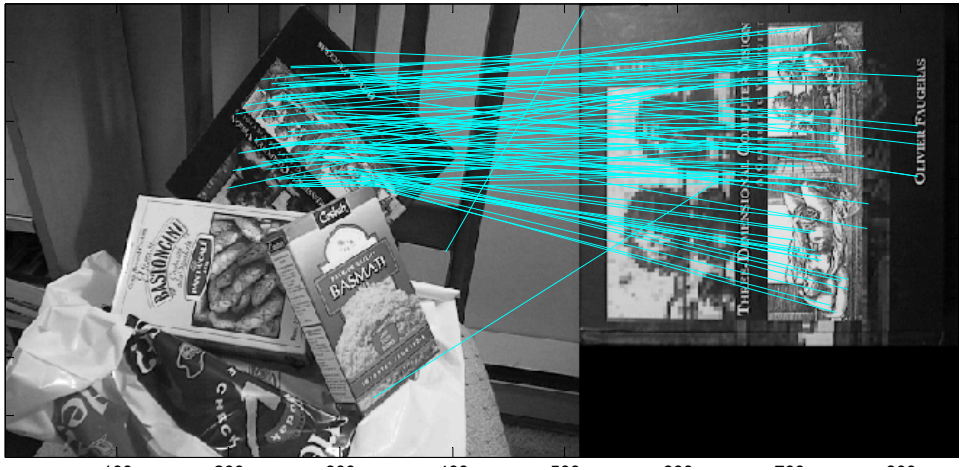
- Métodos de Aprendizaje Inductivo.
  - Adaboost
  - Vecino más cercano
  - Clustering (K-means – Mean-Shift)
  - Redes neuronales
  - Árboles de decisión
  - Listas de reglas

- Métodos Basados en Puntos Significativos (Bag of Words)

Métodos Basados en Puntos Significativos

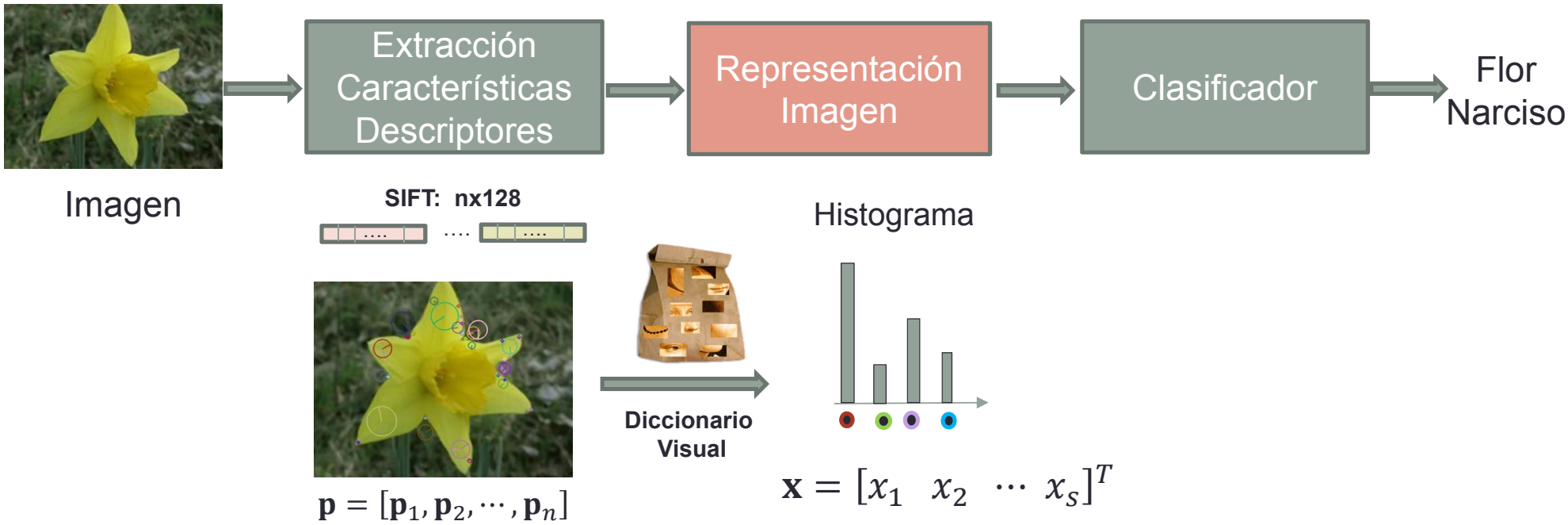
- Reconocimiento de objetos basado en puntos significativos
  - Aproximación basada en algunos puntos importantes de la imagen.
  - Cuando el objeto gira, la apariencia total cambia mucho.
  - Pero la apariencia es más estable cerca de puntos significativos. P.e. Esquinas, SIFT, SURF





Bag of Words (BoW)

- Reconocimiento de objetos basado en puntos significativos
  - Descripción basada en algunos puntos significativos de la imagen: SIFT, SURF (conjunto de puntos y descriptores)
  - Numero de puntos variables, descriptores muy grandes (concatenación de descriptores de cada punto)
  - Se introduce una nueva etapa que obtiene una representación de imagen a partir de un diccionario de palabras visuales.
  - El nuevo vector de descripción (Representación de imagen) es el histograma (frecuencia de repetición) de las palabras visuales.



Bag of Words (BoW)

- Creación del Diccionario Visual
  - Calculamos los puntos y descriptores en un conjunto (***k***) de imágenes: $\mathbf{P}_k$
  - Las representamos en el espacio de características del descriptor
  - Fijamos el tamaño del diccionario (***s***)
  - Aplicamos un algoritmo de clustering (K-means) agrupando los datos en ***s*** clases (palabras visuales) representados por sus medias

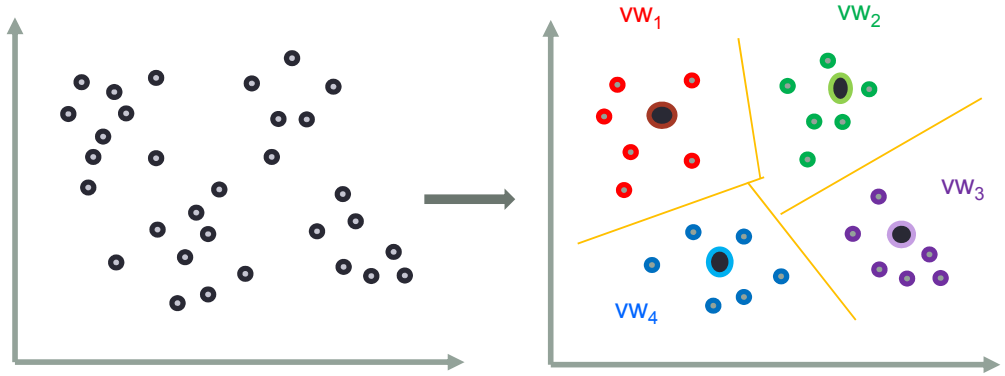
Imagen

SIFT: nx128

$$\mathbf{P}_k = [\mathbf{p}_1,\ \mathbf{p}_2, \cdots\ \mathbf{p}_{n_k}]$$

Diccionario Visual

Clustering: K-means



Tamaño diccionario (s)

Bag of Words (BoW)

- Representación de una imagen
  - Calculamos los puntos y descriptores de la imagen $\mathbf{P} = [\mathbf{p}_1,\ \mathbf{p}_2, \cdots\ \mathbf{p}_n]$
  - Asignamos cada vector de características a cada palabra visual (vecino más cercano)
  - Calculamos el histograma (número de veces que aparece cada palabra visual en la imagen). Cada palabra se pondera de forma diferente (aquellas palabras poco usuales tienen un mayor peso)
  - Se normaliza el histograma para evitar la dependencia del número de palabras en la imagen.
  - Obtenemos un vector de representación de la imagen $\mathbf{x}$ de tamaño $\boldsymbol{s}$.

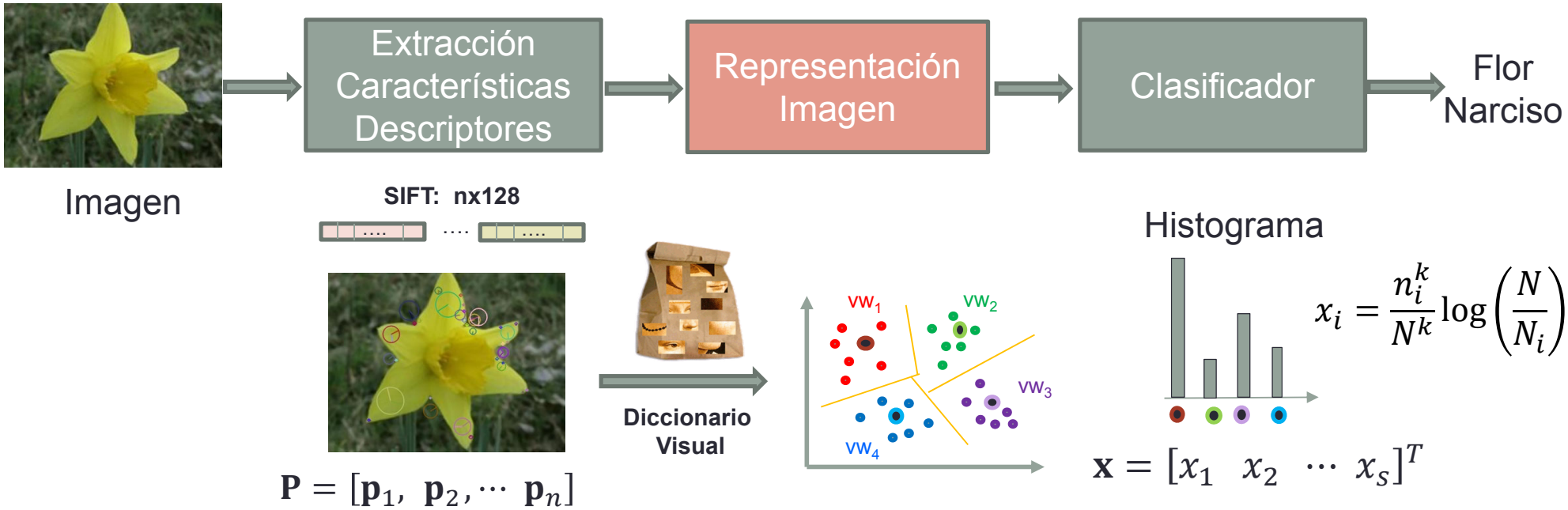


Tabla de Contenidos

- Introducción
- Esquema general de un sistema de reconocimiento
- Teoría Bayesiana de decisión
- Funciones de decisión
- Selección de características
- Otros métodos de reconocimiento

....Fin