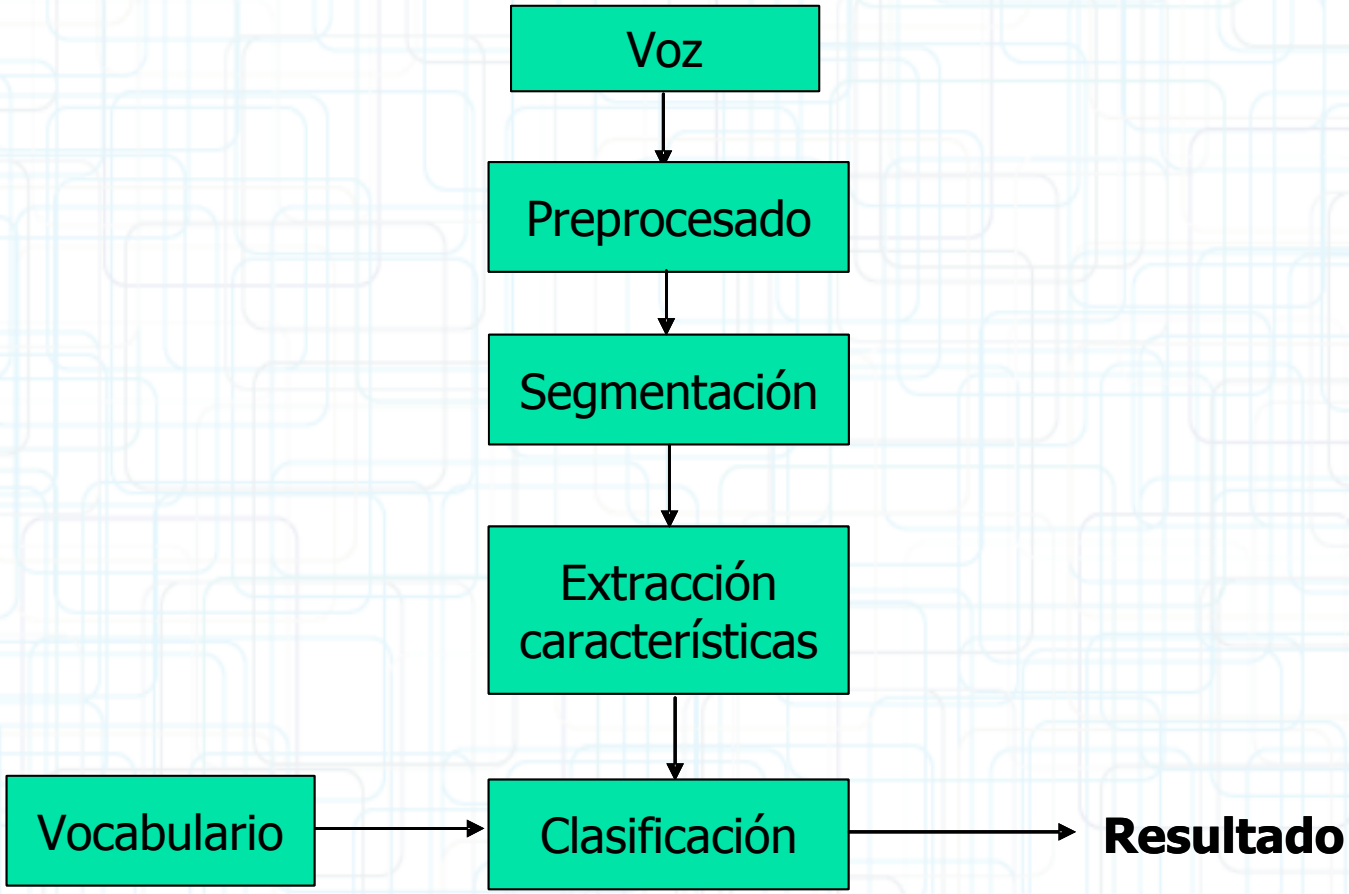
Reconocimiento automático del habla IV

Cuantificación vectorial e introducción a las técnicas de reconocimiento

Fases de un sistema de reconocimiento de voz



Composición del vector de características o de observación

- Al final del proceso de extracción de características obtenemos un vector que tendrá un conjunto de características que representan el fragmento de la señal de voz de la trama.
- Estas características pueden ser:
  - Los coeficientes LPC, Cepstrum o MFCC.
  - También suelen incorporarse:
    - Primera y segunda derivadas del cepstrum respecto al tiempo (características dinámicas del sistema)
    - Logaritmo de la energía,
    - ...

Cuantificación vectorial

- Idea:
  - Clasificar un conjunto de vectores de características y buscar los mejores representantes (*codewords*) para reducir el tamaño de la información a manejar.
- Error por distorsión:
  - Error que se produce al reemplazar los vectores de entrada por un conjunto de *codewords* (*codebook*).

Cuantificación vectorial

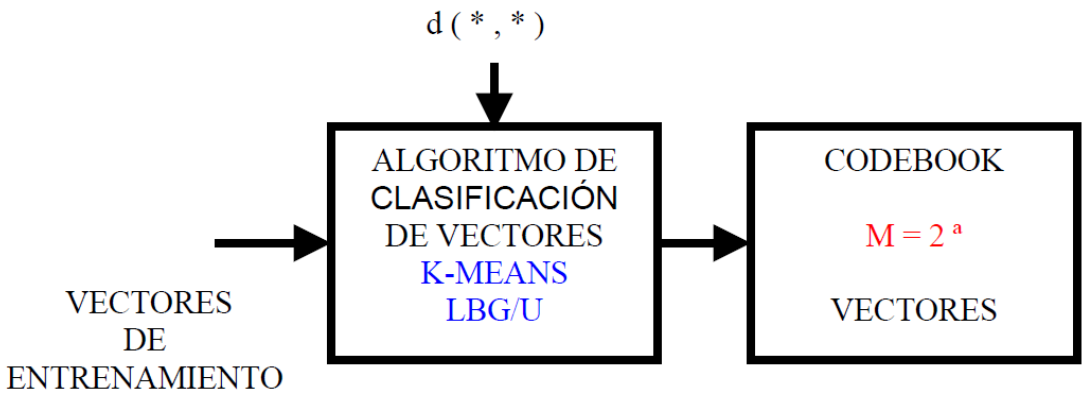
- Ventajas:
  - Reduce el almacenamiento de la información.
  - Reduce el cálculo para determinar la distancia entre vectores espectrales.
  - Representación discreta de las señales de voz.

Cuantificación vectorial

- Desventajas:
  - Distorsión en la representación del vector.
  - El almacenamiento requerido para un *codebook* que minimice el error por distorsión no es pequeño.

Cuantificación vectorial

- Cálculo del *codebook*:
  - Un conjunto de vectores de observación que se usa como conjunto de entrenamiento.
  - Una medida de distancia.
  - Un procedimiento para clasificar y calcular los centroides.



Cuantificación vectorial

- Aspectos a considerar del grupo de entrenamiento:
  - Para señales de voz: edades, acentos, sexo, velocidad, …
  - Condiciones del discurso: ambiente ruidoso o silencioso, …
  - Transductores y sistema de transmisión: ancho de banda del micrófono o del canal, …
  - Reconocimiento de palabras aisladas o de un discurso continuo.

Cuantificación vectorial

- Medidas de distancia:
  - Condiciones que debe cumplir una medida de distancia:
    - $d(X,Y) \geq 0$
    - $d(X,Y) = d(Y,X)$
    - $d(X,X) = 0$
    - $d(X,Y) \leq d(X,Z) + d(Z,Y)$

$X = (x_1, x_2, \ldots, x_n)$

$Y = (y_1, y_2, \ldots, y_n)$

Cuantificación vectorial

- Medidas de distancia:

Distancia euclídea: $$d(X,Y)=\sqrt{\sum_{i=1}^{n}(X_i-Y_i)^2}$$

Distancia euclídea al cuadrado: $$d(X,Y)=\sum_{i=1}^{n}(X_i-Y_i)^2$$

Distancia del coseno: $$d(X,Y)=\frac{\sum_{i=1}^{n}(X_iY_i)}{\sqrt{\sum_{i=1}^{n}X_i^2\sum_{i=1}^{n}Y_i^2}}$$

Distancia de Manhattan: $$d(X,Y)=\sum_{i=1}^{n}|X_i-Y_i|$$

Distancia de Chebyshev: $$d(X,Y)=\max|X_i-Y_i|$$

Distancia de potencias: $$d(X,Y)=\sqrt[r]{\sum_{i=1}^{n}(X_i-Y_i)^p}$$

Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Los N vectores de características de tamaño D quedan representados por M vectores (*codewords).*
  - El conjunto entero de vectores forma el *codebook.*
  - Se delimitan M regiones (regiones de *Voronoi*) con sus correspondientes *codewords* como centroides:

$$V_i = \left\{ x \in \Re^D : d(x, Cw_i) \leq d(x, Cw_j) \right\} \forall \ j \neq i \quad \text{con} \quad 1 \leq i \leq M$$

Cuantificación vectorial

- Algoritmo de creación del *codebook*:

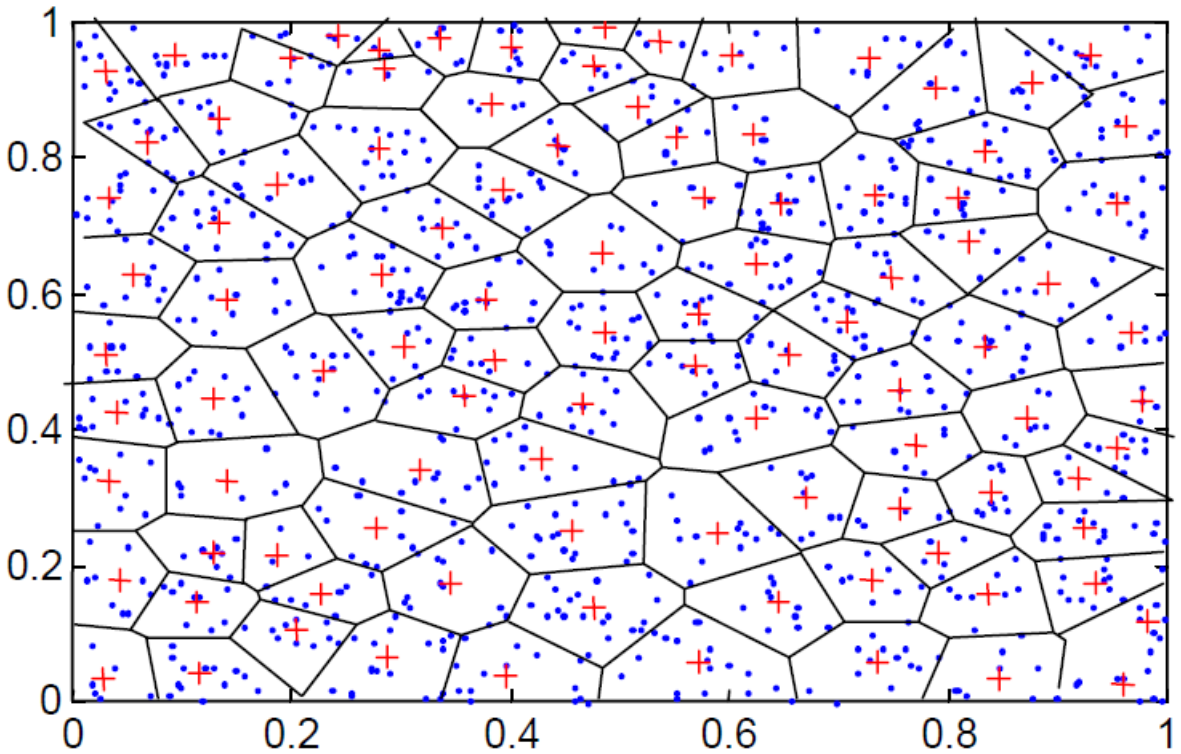


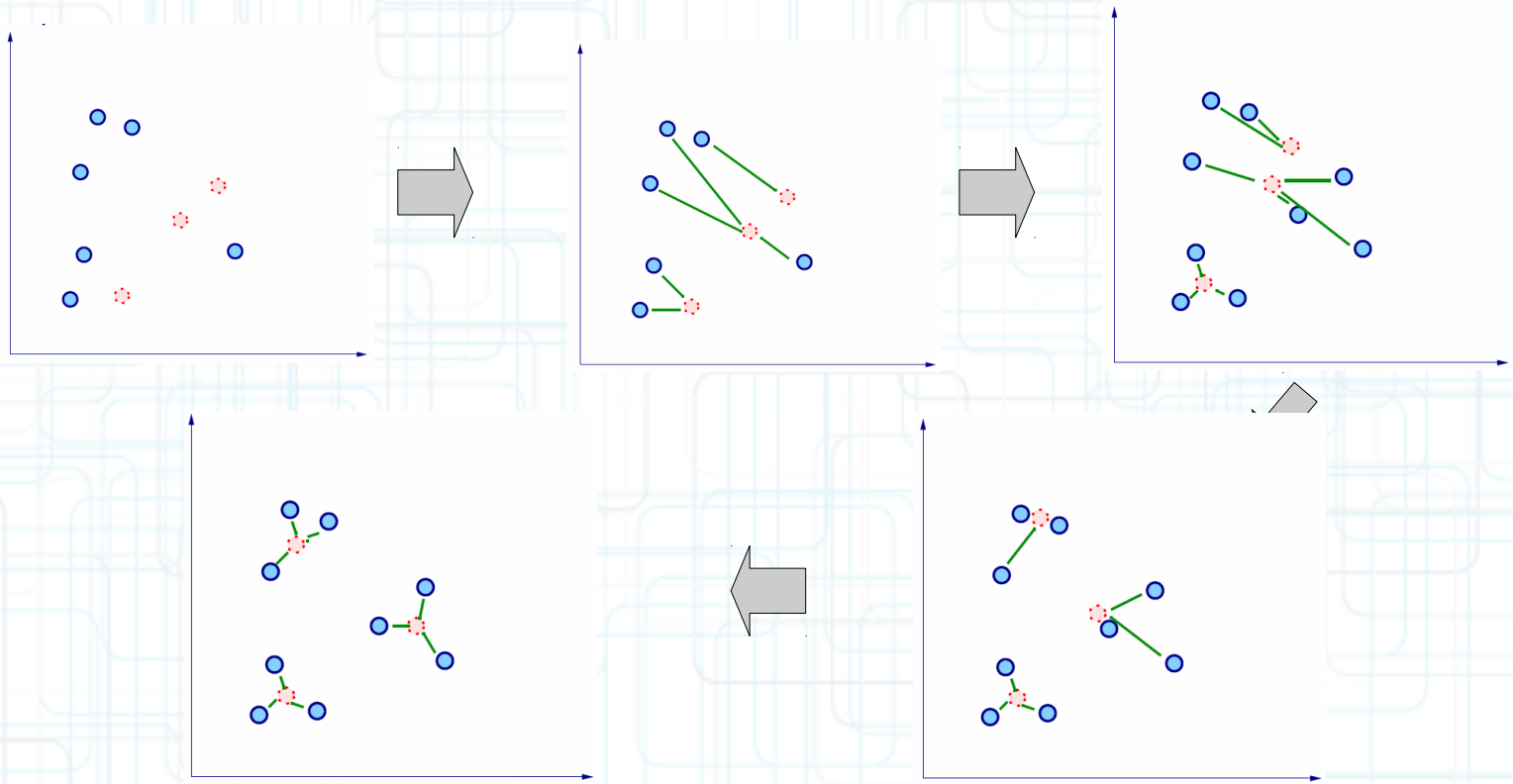
Diagrama de Voronoi

Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo K-means:
    1. Elegir M ejemplos que actúan como semillas (*M* número de clusters).
    2. Para cada ejemplo, añadir ejemplo a la clase más similar.
    3. Calcular el centroide de cada clase, que pasan a ser las nuevas semillas.
    4. Si no se llega a un criterio de convergencia (por ejemplo, dos iteraciones no cambian las clasificaciones de los ejemplos), volver a 2.

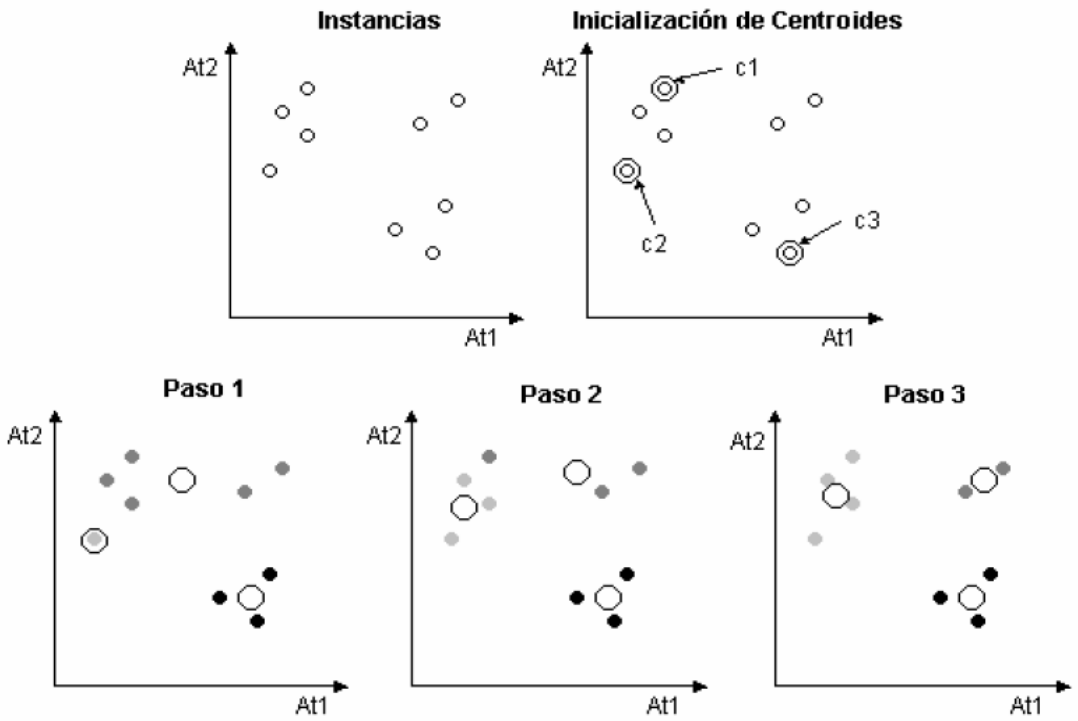
Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo K-means:



Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo K-means:

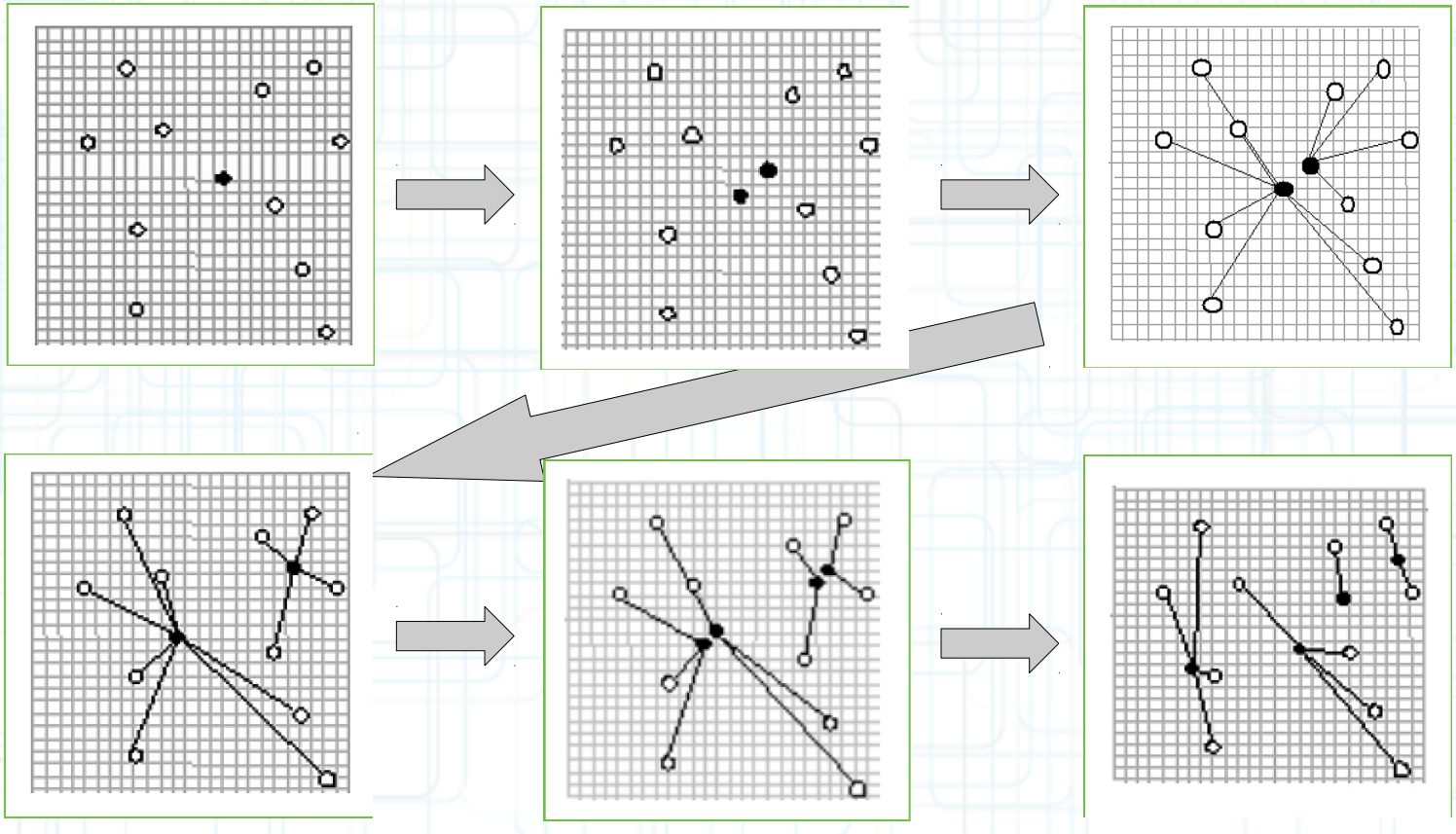


Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo LBG (Linde-Buzo-Gray):
    1. Designar un *codeword* inicial (promedio de los vectores de características del conjunto de entrenamiento).
    2. Dividir el *codeword* inicial en dos (casi juntos y en la misma posición que el original).
    3. Aplicar k-means para calcular el mejor grupo de centroides.
    4. Repetir los pasos 2 y 3 hasta llegar a un *codebook* de tamaño M.

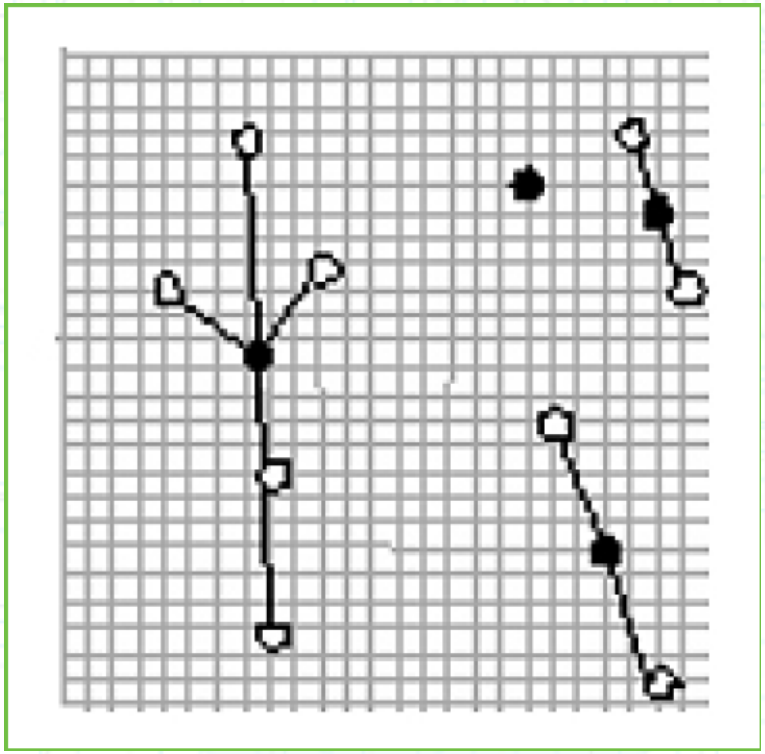
Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo LBG (Linde-Buzo-Gray):



Cuantificación vectorial

- Algoritmo de creación del *codebook*:
  - Algoritmo LBG (Linde-Buzo-Gray):



Cuantificación vectorial

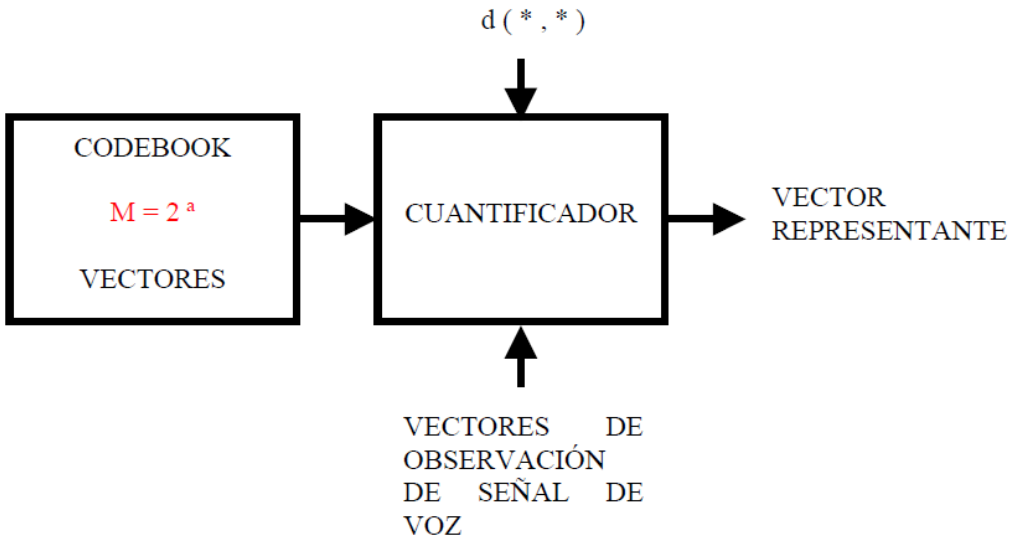
- Algoritmo de cuantificación de vectores:

$$Vm^* = arg\ min\ d(V, Cw)$$

Vector del *codebook*

Vector de características

$$1 \leq w \leq M$$



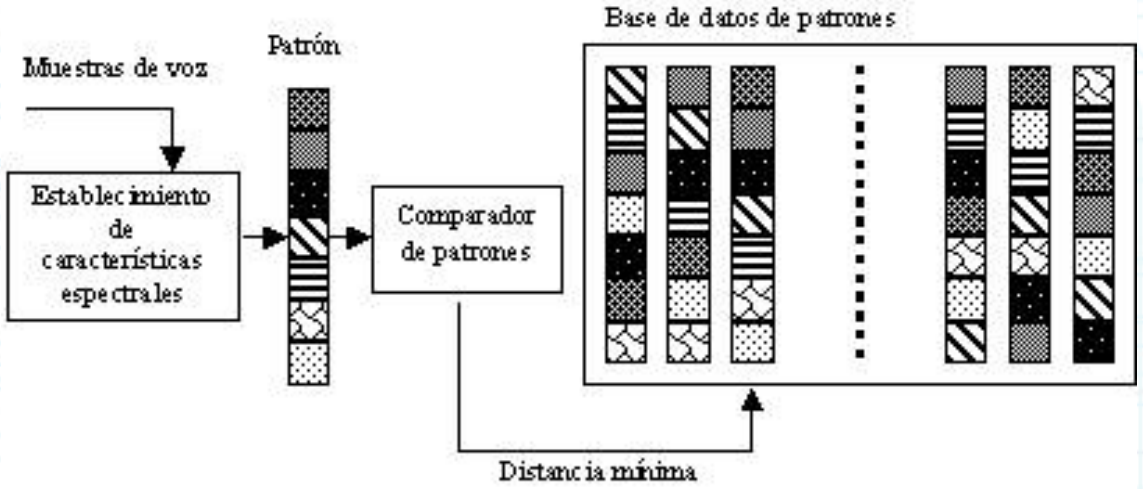
Técnicas de reconocimiento

- Comparación de patrones:
  - Alineamiento temporal.
  - DTW (*Dynamic Time Warping*).
- Modelos ocultos de Markov (HMM – *Hidden Markov Models*)
- Redes Neuronales

Alineamiento temporal

- Compara la plantilla o patrón de entrada con los patrones de la base de datos.
- Estas plantillas o patrones representan las unidades a reconocer y son un conjunto de características acústicas ordenadas en el tiempo (secuencias de vectores de características o *codewords* de un *codebook*).

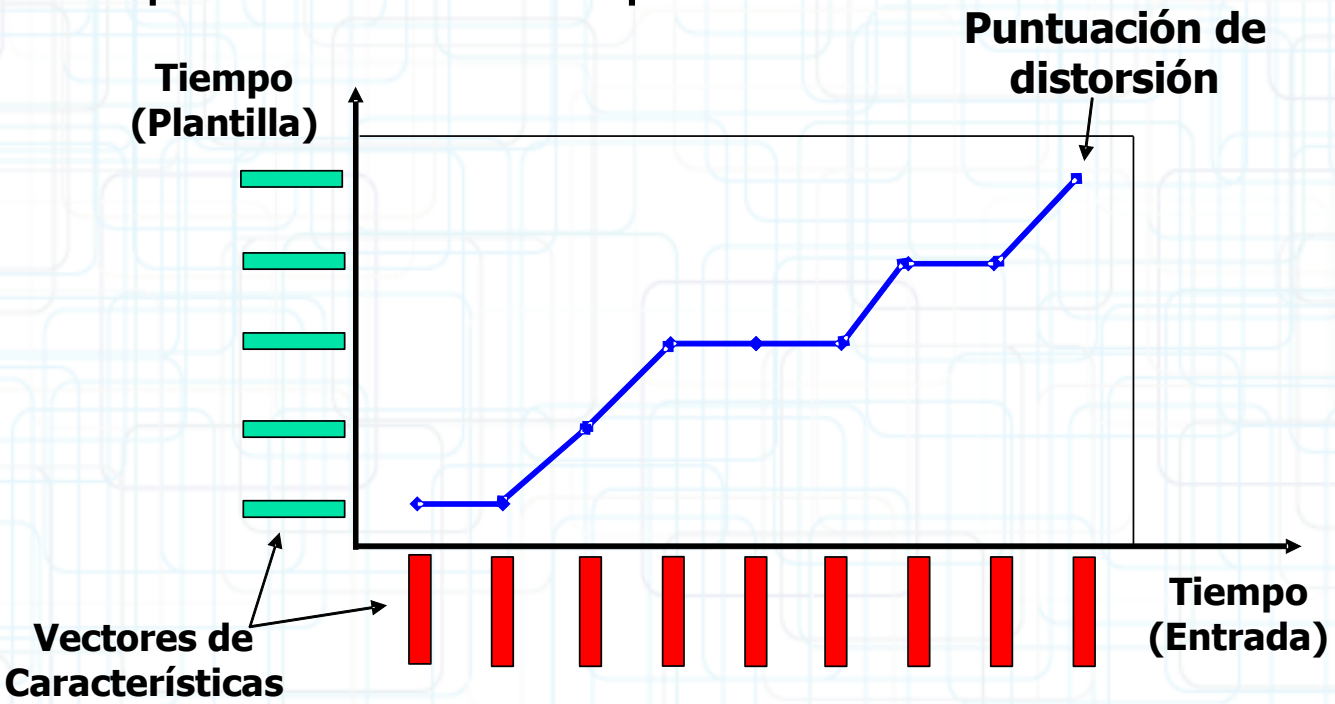
Alineamiento temporal



- El que menor distancia obtiene es la palabra elegida.
- Problema: Las palabras no tienen siempre la misma duración.

Alineamiento temporal dinámico (DTW)

- Es un algoritmo para medir la similaridad entre dos patrones que pueden variar en el tiempo.
- Intenta encontrar el mejor alineamiento entre los dos patrones a comparar.

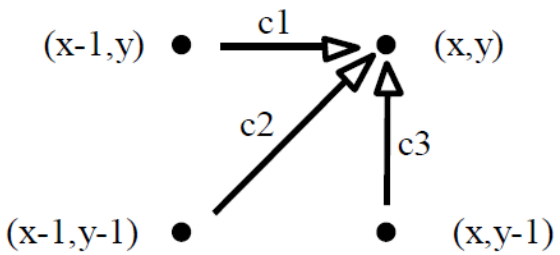


Alineamiento temporal dinámico (DTW)

- La distancia entre dos patrones será la distancia menor de todos los caminos posibles.
- La distancia de un camino es la suma de las distancias parciales a lo largo del dicho camino.

Alineamiento temporal dinámico (DTW)

- La distancia entre dos patrones será la distancia menor de todos los caminos posibles.
- La distancia de un camino es la suma de las distancias parciales a lo largo del dicho camino.



$$D(1,1) = d(T[1],R[1])$$

$$D(x,y) = \min\begin{cases} D(x-1,y) + c1 * d(T[x],R[y]), \\ D(x-1,y-1) + c2 * d(T[x],R[y]), \\ D(x,y-1) + c3 * d(T[x],R[y]) \end{cases}$$

costes

Modelos Ocultos de Markov (HMM)

Un HMM es un modelo es un modelo probabilístico en el que:

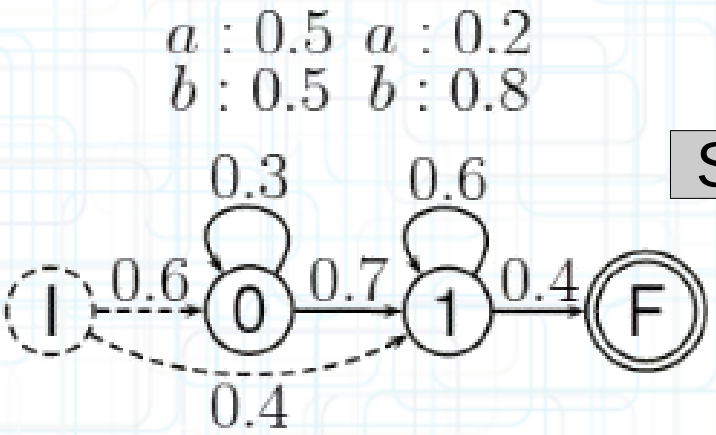
- el estado en el instante de tiempo t+1 sólo depende del estado del modelo en el instante t.
- sólo se observa una secuencia de símbolos que genera el modelo.
- la secuencia de estados visitados permanece oculta.

Modelos Ocultos de Markov (HMM)

Un HMM es un modelo formado por:

- Un conjunto finito de estados.
- Un conjunto de símbolos del alfabeto.
- Un vector de probabilidades de los estados de ser estados iniciales.
- Las transiciones entre estados vienen dadas por una matriz de probabilidades de transición.
- En cualquier estado, la observación es generada de acuerdo a una distribución de probabilidades de emisión.

Modelos Ocultos de Markov (HMM)



Estados

$$Q = \{0, 1, F\}$$

$$\Sigma = \{a, b\}$$

Símbolos

Prob. iniciales

π	0	1
	0.6	0.4

A	0	1	F
0	0.3	0.7	
1		0.6	0.4

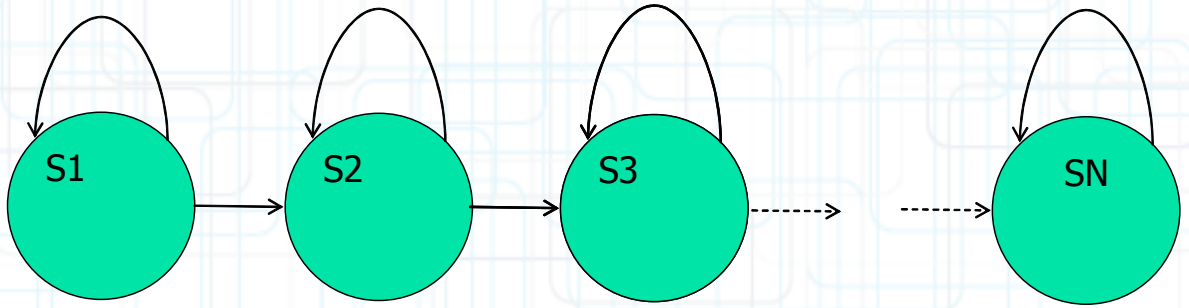
Prob. de transición

B	a	b
0	0.5	0.5
1	0.2	0.8

Prob. de emisión

Modelos Ocultos de Markov (HMM)

- En reconocimiento de voz:
  - Cada estado modela un segmento (un difono).
  - Las transiciones dentro del mismo estado representan la continuación en el tiempo de un mismo suceso acústico.
  - Cada HMM modela una palabra.



Modelos Ocultos de Markov (HMM)

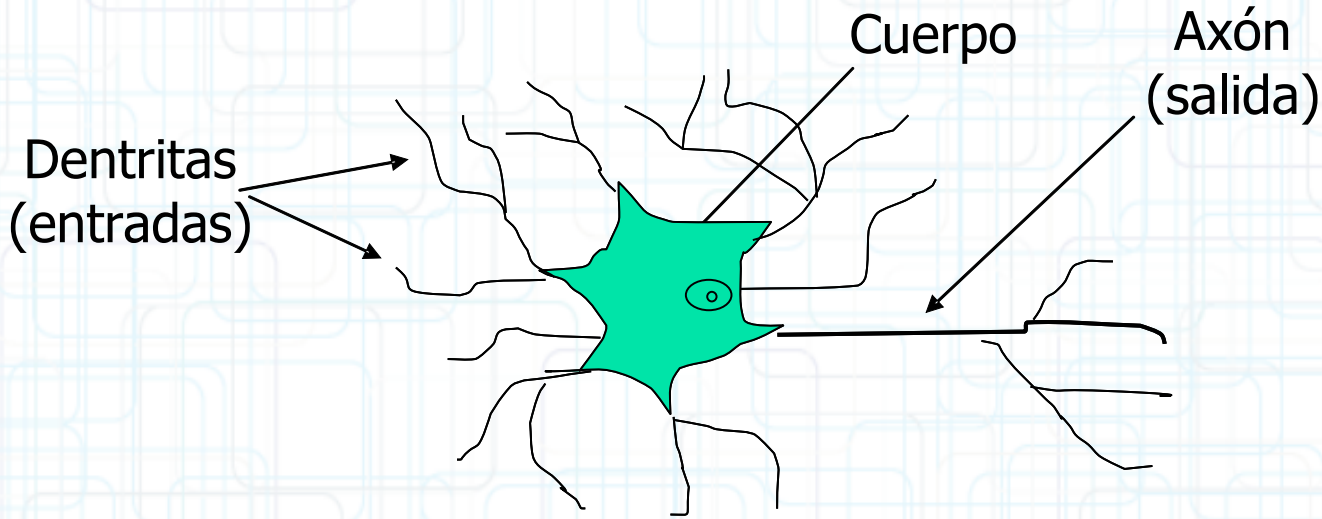
- Problemas básicos: Dada una secuencia de observaciones y un modelo ...
  - Problema de evaluación:
    - ..., ¿cual es la probabilidad de que dicho modelo haya generado esa secuencia?
  - Problema de decodificación:
    - ..., ¿cual es la secuencia óptima de estados que mejor explica esa secuencia de observaciones?
  - Problema de entrenamiento:
    - …, ¿cómo ajustar los parámetros del modelo para maximizar la probabilidad de generar esa secuencia?

Modelos Ocultos de Markov (HMM)

- Soluciones:
  - Problema de evaluación:
    - Procedimiento Forward-Backward.
  - Problema de decodificación:
    - Algoritmo de Viterbi.
  - Problema de entrenamiento:
    - Baum-Welch (máxima verosimilitud de los modelos)
    - Segmental k-means (máxima verosimilitud de la secuencia de estados)
    - Criterio MMIE (máxima información mutua)
    - Entrenamiento correctivo (mínima tasa de errores)

Redes neuronales

- Son una simulación del sistema nervioso.



Redes neuronales

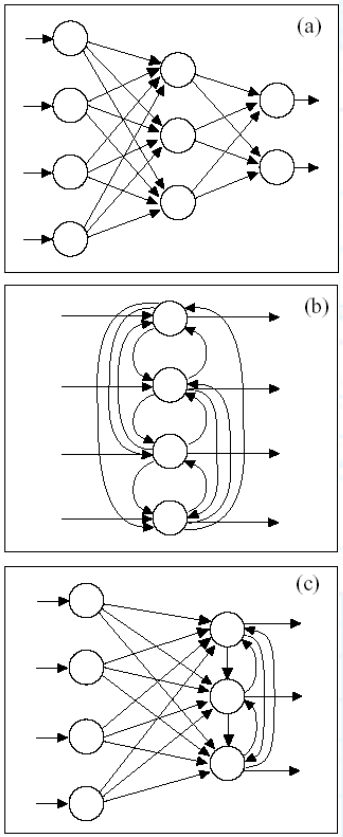
- Definición de red neuronal:
  - Un conjunto de neuronas artificiales conectadas entre sí mediante una serie de arcos llamados *conexiones*. Estas conexiones tienen valores numéricos asociados, denominados *pesos de la conexión*. Estas neuronas se suelen dividir en capas de distintos niveles con conexiones que unen las neuronas de distintas capas y/o neuronas de una misma capa.

Redes neuronales

- Definición de neurona artificial:
  - Un elemento relativamente simple que recibe un conjunto de señales de entrada (del mundo exterior o de otras neuronas) ponderadas por los pesos asociados a las conexiones por las que llegan dichas señales, procesa la información mediante una serie de operaciones simples y emite una señal de salida como respuesta a las señales de entrada.

Redes neuronales

- Clasificación:
  - Según la forma de conexión:



    - Redes con alimentación hacia delante (*feedforward*). Tienen conexiones en un solo sentido (Ej.: perceptrón multicapa y redes de funciones de base radial).
    - Redes recurrentes. Con conexiones en todas direcciones (Ej.: redes BAM, ART y de Hopfield).
    - Redes parcialmente recurrentes. Con sólo unas pocas conexiones recurrentes (Ej.: las redes de Jordan o de Elman).

Redes neuronales

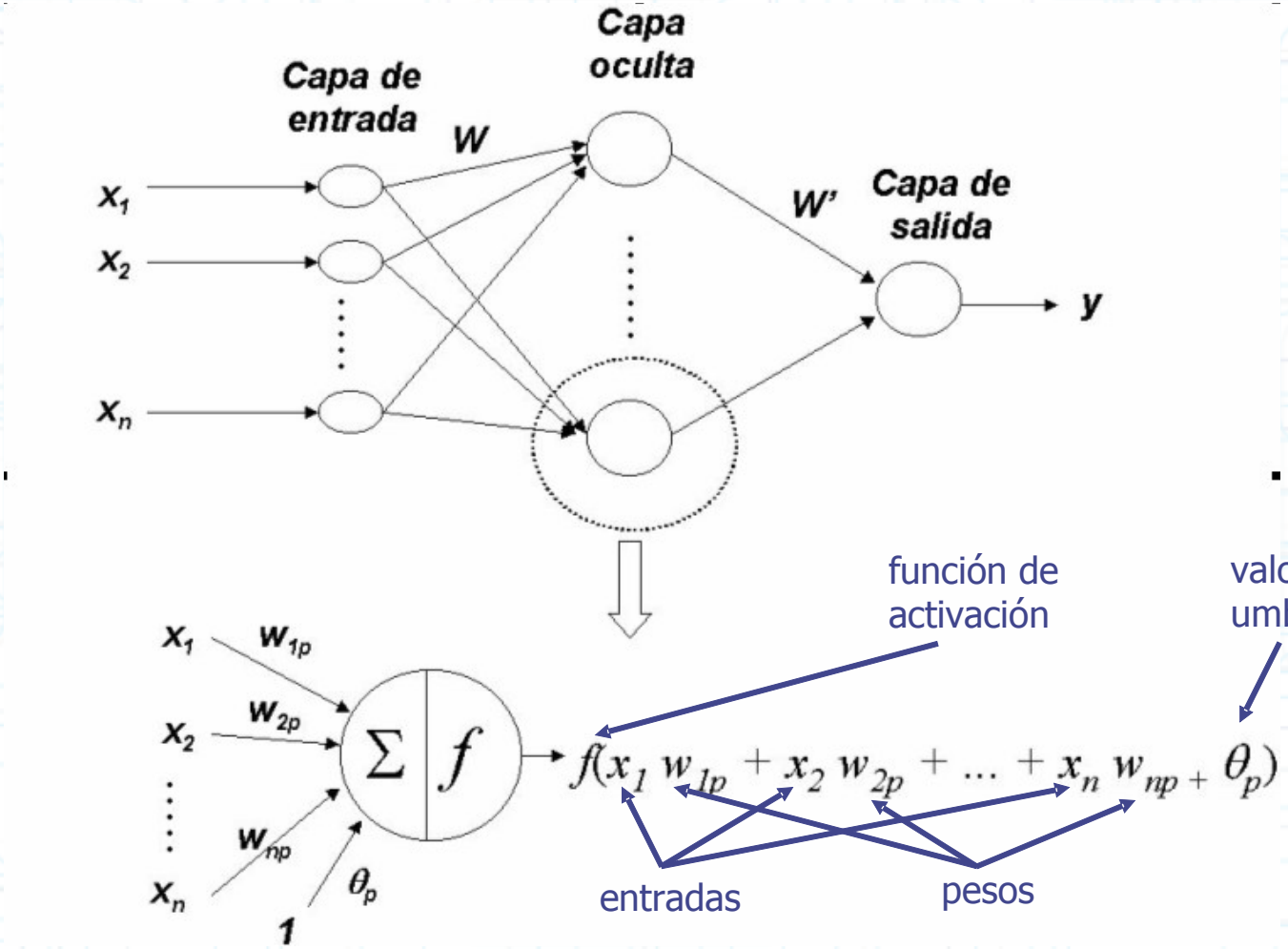
- Clasificación:
  - Según el paradigma de aprendizaje:
    - Redes supervisadas. Ejemplos de este tipo de redes son las redes con alimentación hacia delante y algunas de las redes recurrentes.
    - Redes no supervisadas. Ejemplos son parte de las redes recurrentes como la red de Kohonen y ART.

Redes neuronales

- Perceptrón multicapa:
  - Red neuronal cuyas neuronas se encuentran distribuidas en capas, de modo que las salidas de todas las neuronas que constituyen una determinada capa *i* pasan a ser las entradas de las neuronas de la capa *i+1*. Las capas de neuronas que constituyen este tipo de RNA se pueden clasificar en tres tipos: de entrada, ocultas y de salida.

Redes neuronales

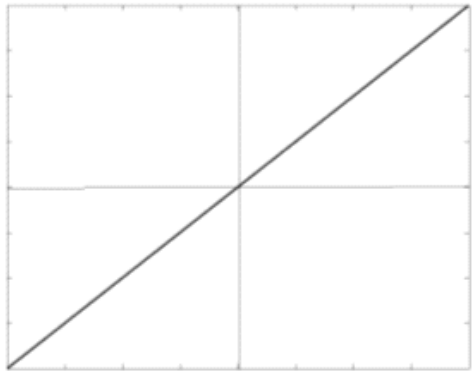
- Perceptrón multicapa:



Redes neuronales

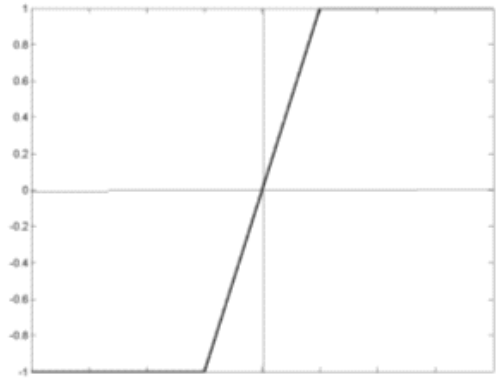
- Perceptrón multicapa (funciones de activación):

función lineal



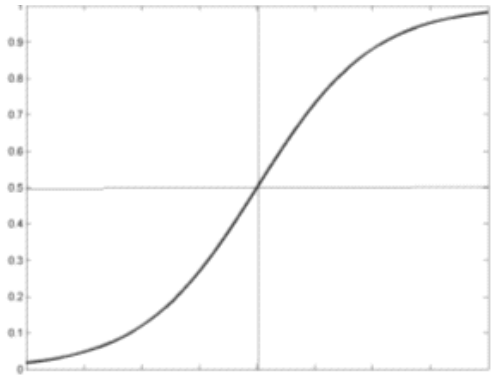
$$f(x) = x$$

función mixta



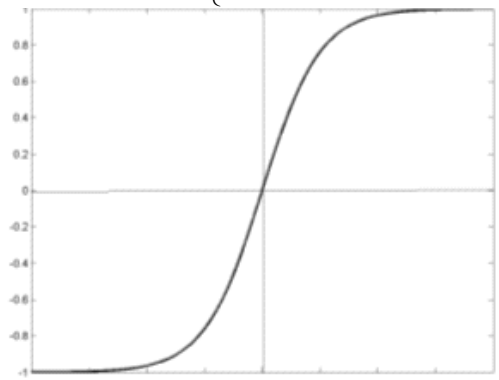
$$f(x) = \begin{cases} -1 & si \quad x < -c \\ x & si \quad -c \le x \le c \\ +1 & si \quad x > c \end{cases}$$

función sigmoidal (valores entre 0 y 1)



$$f(x) = \frac{1}{1 + e^{-x}}$$

función sigmoidal (valores entre -1 y 1)



$$f(x) = tanh(x) = \frac{senh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Redes neuronales

- Perceptrón multicapa (entrenamiento con el algoritmo BackPropagation):

1. Inicialización con valores aleatorios de los pesos y umbrales.
2. Dado un patrón del conjunto de entrenamiento $(\boldsymbol{x}, y)$, se presenta el vector $\boldsymbol{x}$ a la red y se propaga hacia delante, calculando las salidas de las neuronas de la red para dicha entrada:
   - capa de entrada:

$$a_j = x_j \text{ donde } j = 1, ..., n$$

   - capa oculta:

$$b_j = f\left(\sum_{i=1}^{n} w_{ji} a_i + \theta_j\right) \text{ donde } j = 1, ..., p$$

   - capa de salida:

$$o = f\left(\sum_{i=1}^{p} w'_i b_i + \theta'\right)$$

Redes neuronales

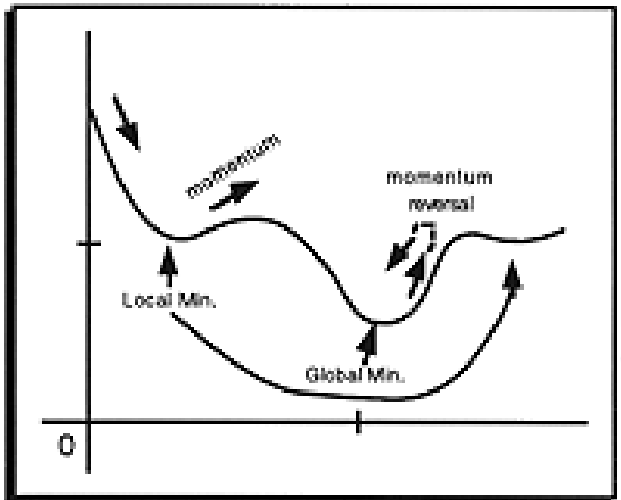
- Perceptrón multicapa (entrenamiento con el algoritmo BackPropagation):

3. Se evalúa el error cometido e por la red, es decir, el error entre la salida devuelta por la última por la última capa del perceptrón y la salida deseada y ($e = (y - o)^2$).
4. Se modifican todos los parámetros de la red utilizando las ecuaciones siguientes:
   - para la capa de salida:

$$w_i^{(t+1)} = w_i^{(t)} + \alpha^{(t)} * \frac{\partial e}{\partial w_i}$$

   - para la capa oculta:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \alpha^{(t)} * \frac{\partial e}{\partial w_{ij}}$$



Redes neuronales

- Perceptrón multicapa (entrenamiento con el algoritmo BackPropagation):

5. Se repiten los pasos 2, 3 y 4 para todos los ejemplos del conjunto de entrenamiento, completando así un ciclo de aprendizaje.
6. Se evalúa el error global cometido por la red (por ejemplo, el error cuadrático medio)
7. Se repiten todos los pasos anteriores hasta que se verifique un criterio de parada establecido: el error global sea inferior a una cota, los parámetros no sufran modificaciones o se ha realizado un número determinado de ciclos de aprendizaje.

Redes neuronales

- Perceptrón multicapa (clasificación):
  1. Se le proporciona el vector de entrada ***x*** a la capa de entrada del perceptrón multicapa.
  2. Las neuronas de entrada la envían ponderadas a la capa oculta
  3. En la capa oculta son procesadas y el resultado enviado ponderado a la capa de salida.
  4. En esta última capa son de nuevo procesadas y el resultado de las neuronas de dicha capa corresponde a la salida producida por la red.

Redes neuronales

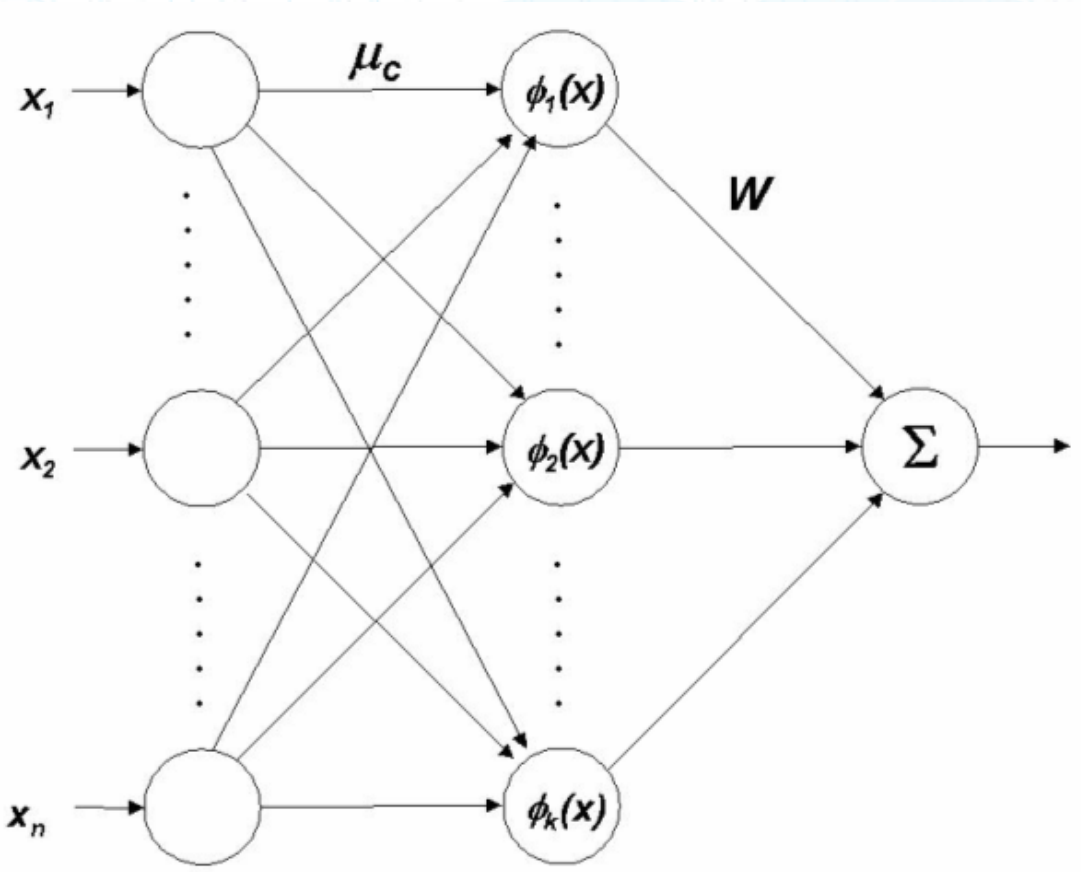
- Redes de funciones de base radial (RBFN-Radial Basis Function Networks):
  - Las funciones de base radial (*RBF - Radial Basis Functions*) son funciones cuya salida es simétrica alrededor de un valor que se suele denominar centro, centroide o núcleo (μ_c). Estas funciones se suelen ver también afectadas por otro parámetro, σ, que nos determina la anchura de la función.

$$f_k(\boldsymbol{x},\boldsymbol{w})) = w_0 + \sum_{i=1}^{k-1} w_i \, \phi\left(\|\boldsymbol{x} - \boldsymbol{\mu}_c\|\right) = \sum_{i=0}^{k-1} w_i \, \phi\left(\|\boldsymbol{x} - \boldsymbol{\mu}_c\|\right)$$

$$f_n(\boldsymbol{x},\boldsymbol{w})) = \boldsymbol{w}\boldsymbol{\phi}$$

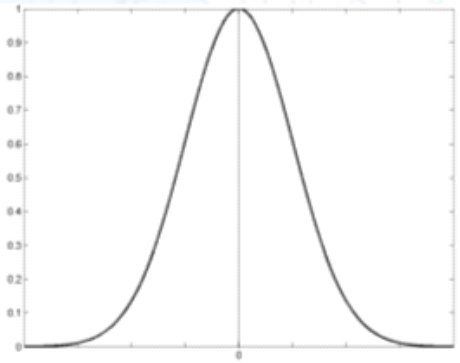
Redes neuronales

- Redes de funciones de base radial (RBFN-Radial Basis Function Networks):

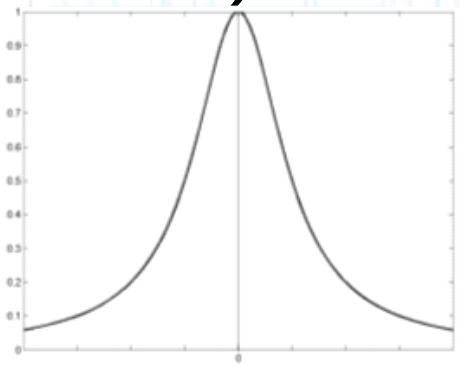


Redes neuronales

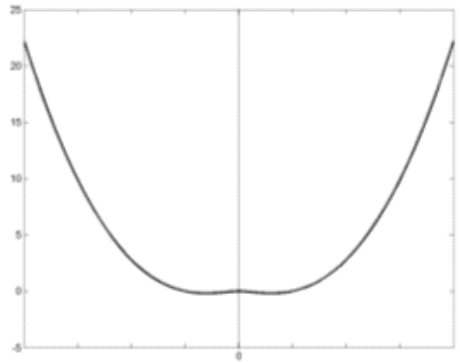
- Redes de funciones de base radial (Funciones de base radial):



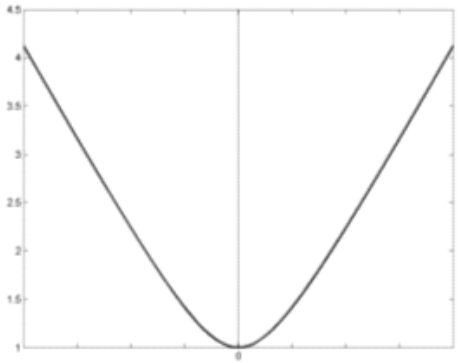
a) Gausiana: $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$



b) $\phi(x) = (x^2 + \sigma^2)^{-\alpha}$ tal que $\alpha > 0$



c) $\phi(x) = \begin{cases} x^2 + ln(|x|) & si \quad x \neq 0 \\ 0 & si \quad x = 0 \end{cases}$



d)) $\phi(x) = (x^2 + \sigma^2)^{\beta}$ tal que $0 < \beta < 1$

Redes neuronales

- Redes de funciones de base radial (Entrenamiento):
  1. Determinar los valores de centros (μ_i) y anchura (σ_i) para cada neurona de la capa oculta.
  2. Calcular los pesos de las unidades de salida.

Redes neuronales

- Redes de funciones de base radial (Estimación de centros y anchura):
  - Centros: Se pueden seleccionar de forma aleatoria entre el conj. de ejemplos o aplicando agrupación natural (*clustering*).
  - Anchuras: Puede ser ...
    - el mismo para todas las unidades (un multiplo de la distancia media entre los centros), o
    - cada unidad tome un valor diferente dependiendo de la parte del espacio de entrada que representa (un multiplo de la distancia media a los *k* vecinos más cercanos).

Redes neuronales

- Redes de funciones de base radial (Estimación de los pesos):
  - Aplicamos el método de la pseudoinversa:

$$\boldsymbol{w} = (\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{Y}$$