MAPAS AUTOORGANIZATIVOS Y MODELOS SIMILARES

José D. Martín Guerrero, Emilio Soria, Antonio J. Serrano

PROCESADO Y ANÁLISIS DE DATOS AMBIENTALES

Curso 2009-2010

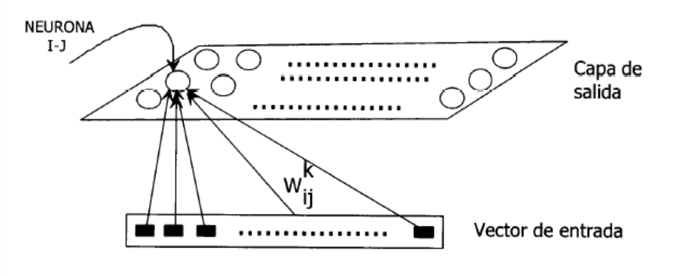
1. Esquema.

- Mapas autoorganizativos.
- Learning Vector Quantization.
- Radial Basis Function Neural Network.

2. Mapas autoorganizativos.

Arquitectura: Mapa de neuronas.

Idea: Patrones de entrada similares quedan asociados a neuronas cercanas en el mapa mediante un proceso de autoorganización (*Self-Organizing Map: SOM*) $\Rightarrow$ Preserva las relaciones topológicas con los patrones de entrada.



Actualización de los pesos de la neurona ij:

$$\mathbf{w_{ij}}(t+1) = \mathbf{w_{ij}}(t) + \alpha(t)h_t(\mathbf{w_{ij}})\left(\mathbf{x}(t) - \mathbf{w_{ij}}(t)\right)$$

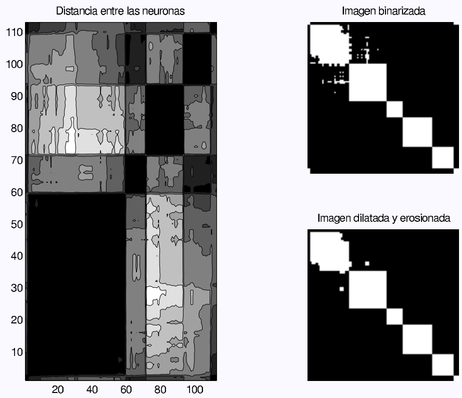
- $\mathbf{x}(t)$ es el patrón de entrada correspondiente.
- $\alpha(t)$ determina el ritmo del aprendizaje (decreciente por razones de estabilidad).
- $h_t(\mathbf{w_{ij}})$ es la función de vecindad: establece el radio de neuronas vecinas de la neurona ganadora que se actualizan y el grado en el que lo hacen.

Variantes del SOM:

- Estructuras toroidales para vencer el problema de actualización de las neuronas situadas en los bordes del mapa.
- Neural Gas.
- Algoritmo de consciencia.
- Growing Grid Algorithm.

Extracción de grupos:

- Tratamiento digital de imágenes: distancias entre neuronas, binarización, erosión y dilatación.



- Método basado en agrupamiento jerárquico: unión de neuronas próximas en el mapa + algoritmo de agrupamiento jerárquico de enlace completo.

3. Learning Vector Quantization

Arquitectura y funcionamiento: SOM
Tipo de aprendizaje: supervisado.

Algoritmo clásico $\Rightarrow$ LVQ1:

1. Se inicializan prototipos.
2. $\forall$ patrón de entrada se encuentra el prototipo más cercano:
   (a) Si las clases del vector de entrada y del vencedor son iguales, se acerca el vector ganador al de entrada:
   $$w^\star = w^\star + \alpha(x - w^\star) \ si \ clase(x) = clase(w^\star)$$
   (b) En caso contrario, se alejan:
   $$w^\star = w^\star - \alpha(x - w^\star) \ si \ clase(x) = clase(w^\star)$$
3. Existe un vector de pesos asociado a cada *cluster*.
4. Una vez entrenado puede funcionar como un clasificador.
5. Habitualmente α es inversamente proporcional al número de iteraciones, pero existe una variante (Optimum LVQ1) en el que viene definida por la siguiente expresión:
   $$\alpha_l(n) = \frac{\alpha_l(n-1)}{1 + (-1)^\beta \alpha_l(n-1)}$$
   donde $\beta = 0$ si se clasifica correctamente el patrón y 1 en caso contrario.

Algoritmo LVQ2.1: Tiene en cuenta tanto el peso (prototipo) vencedor como el 2º clasificado. La adaptación de estos pesos depende de varios factores:

- Los dos pesos corresponden a clases diferentes.
- El vector de entrada pertence a la misma clase que uno de ellos.
- El vector de entrada está en una ventana definida sobre el punto medio de estos dos vectores:

  $$\min\left(\frac{d_1}{d_2}, \frac{d_2}{d_1}\right) > \frac{1-\omega}{1+\omega}$$

  siendo d_1 y d_2 las distancias de los dos vectores más próximos al vector de entrada y ω la anchura de la ventana. Actualización de los pesos:

  $$w_k = w_k + \alpha(x - w_k)$$
  $$w_j = w_j - \alpha(x - w_j)$$

  donde los pesos k y j son los más cercanos, y $x \in C_k$

Algoritmo LVQ3: Se preocupa de donde acaban los pesos al final del proceso de aprendizaje. Si los dos pesos más cercanos representan a la misma clase, ambos se actualizan:

$$w_k = w_k + \beta\alpha(x - w_k)$$

donde $\beta < 1$. Los pesos no solamente respetan las fronteras de decisión sino también la distribución de los datos de entrada.

CONSEJOS PRÁCTICOS:

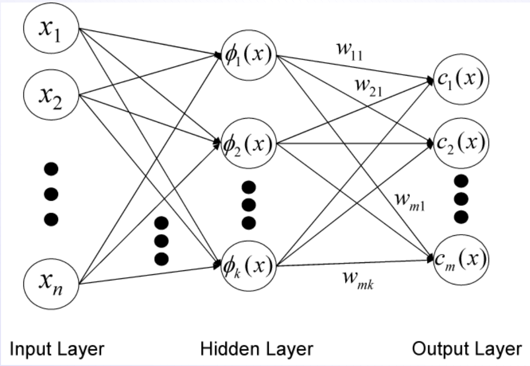
- **Utilizar OLVQ1 hasta la convergencia.**
- **Realizar ajuste fino con LVQ1 o LVQ3.**
- **No utilizar durante muchas iteraciones LVQ2.1.**

4. Radial Basis Function

Arquitectura: una capa de entrada, una capa de neuronas que realizan un procesado no lineal de las entradas y una capa de salida.
Salida i-ésima:

$$y_i = f_i(\mathbf{x}) = \sum_{k=1}^{N} w_{ik}\phi_k\left(\mathbf{x}, \mathbf{c}_k\right) = \sum_{k=1}^{N} w_{ik}\phi_k\left(\|\mathbf{x} - \mathbf{c}_k\|_2\right),\ i = 1, 2, \ldots, m$$

donde $\mathbf{x} \in \Re^{n\times 1}$ es el vector de entrada, $\phi_k(\cdot) : \Re^+ \to \Re$, w_{ik} los pesos de la capa de salida, N el número de neuronas de la capa oculta y $c_k \in \Re^{n\times 1}$ los centros RBF de los vectores de entrada.



Típicas funciones $\phi(\cdot)$:

1. Función lineal: $\phi(x) = x$
2. Aproximación cúbica: $\phi(x) = x^3$
3. Función "thin-plate-spline": $\phi(x) = x^2 \ln x$
4. Función Gaussiana: $\phi(x) = \exp\left(\frac{-x^2}{\sigma^2}\right)$
5. Función multicuadrática: $\phi(x) = \sqrt{x^2 + \sigma^2}$
6. Función multicuadrática inversa: $\phi(x) = \frac{1}{\sqrt{x^2+\sigma^2}}$

En estas funciones x se refiere a $\|x - c\|$. En la capa oculta de la RBF, se calculan las distancias respecto a los centros (entrada y centro asociado).

Aprendizaje de las RBF

- 1ª fase: Cálculo de los centros y de los parámetros de la función base de manera no supervisada (minimización de la dispersión inter-grupo).
- 2ª fase: Cálculo de los pesos w_{ik} minimizando:

$$J(n) = \frac{1}{2}\left|e(n)\right|^2 = \frac{1}{2}\left[y_d(n) - \sum_{k=1}^{N} w_k(n)\phi\left\{x(n), c_k(n)\right\}\right]^2$$

Si la RBF es gaussiana:

$$J(n) = \frac{1}{2}\left[y_d(n) - \sum_{k=1}^{N} w_k(n)\exp\left(-\frac{\|x(n) - c_k(n)\|_2^2}{\sigma_k^2(n)}\right)\right]^2$$

Ecuaciones de actualización:

$$w(n+1) = w(n) - \mu_w \frac{\partial J(n)}{\partial w}$$

Los parámetros calculados en la primera fase también se pueden calcular por descenso de gradiente.

RBF comparado con (MLP):

- Activación igual en hiperelipsoides (hiperplanos).
- Información localizada (distribuida).
- Estructura fija de dos capas diferenciadas (múltiples capas).
- Aprendizaje rápido (no lineal).
- Aprendizaje en dos etapas: supervisada y no supervisada (una etapa supervisada).

Consejos:

- Se debe usar RBF en sistemas adaptativos de aprendizaje rápido, cuando haya fuertes no linealidades presentes o cuando la función de distribución dé idea de la importancia de los datos.
- Se debe usar MLP cuando haya escasez de muestras o se necesite extrapolación de datos.