5.1. Redes de aprendizaje supervisado basadas en la cuantificación vectorial

Curso de doctoramiento

"Técnicas de Computación Flexíbeis"

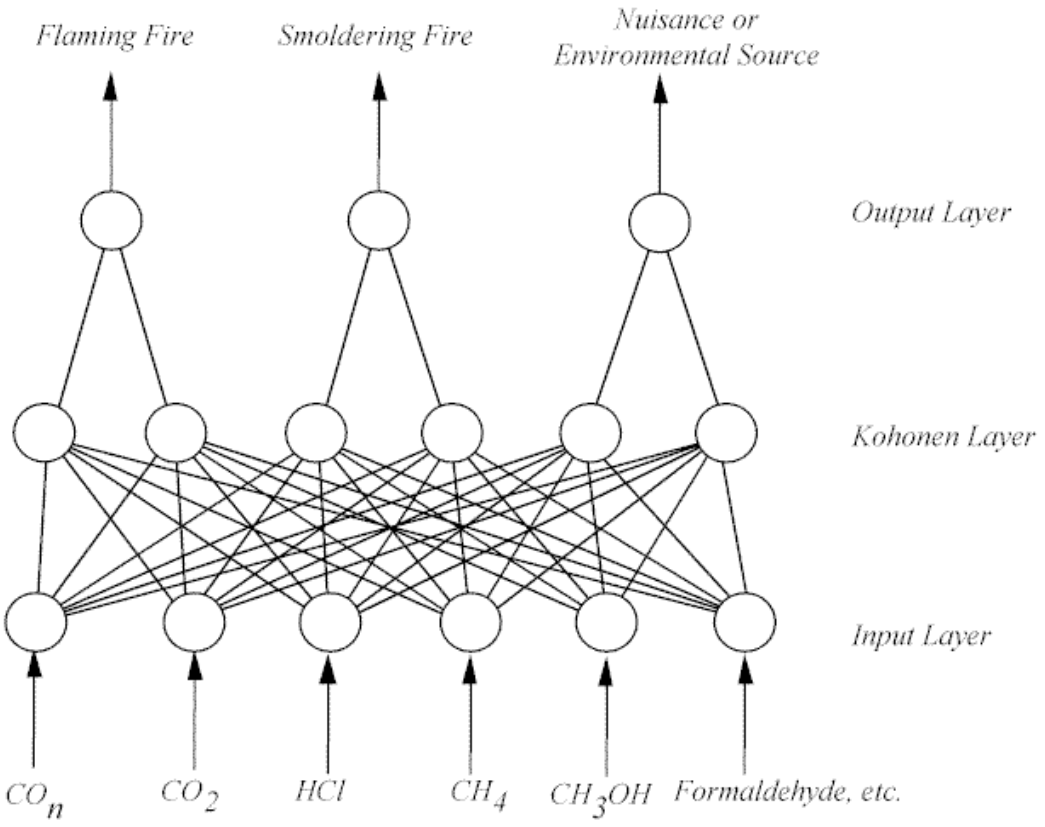
Learning Vector Quantization (LVQ)

- Versión supervisada de SOM (SOM para clasificación)
- LVQ usa todos los datos para situar todos los prototipos.
- Pista: modificar los prototipos de modo que se sitúen cerca de sus clases.
  - Seleccionar un ejemplo de entrenamiento:
  - Si el prototipo más cercano pertenece a la misma clase que el ejemplo, acerca el prototipo al ejemplo.
  - Si el prototipo más cercano pertenece a otra clase, aleja el prototipo del exemplo.

Learning Vector Quantization (LVQ)

- Red mono-capa de propagación directa
- Entrenamiento híbrido supervisado – no supervisado.
- Entrenamiento en diferido.
- Transforma patrones multi-dimensionales a una representación bi-dimensional en la que no se conserva la topología (a diferencia de SOM).

Ejemplo de red LVQ



Entrenamiento en LVQ

1. Seleccionar aleatoriamente *N* ejemplos de entrenamiento como prototipos iniciales para cada clase *k*: $\mathbf{m}_{k1}$, ..., $\mathbf{m}_{kN}$
2. Seleccionar aleatoriamente un ejemplo de entrenamiento $\mathbf{x}_i$. Sea $\mathbf{m}_{kj}$ el prototipo más cercano a $\mathbf{x}_i$.
   a. Si $\mathbf{x}_i$ pertenece a la clase *k*, acercar el prototipo $\mathbf{m}_i$ al ejemplo de entrenamiento $\mathbf{x}_i$:

$$\vec{m}_{ki}(t+1) = \vec{m}_{ki}(t) + \alpha\left(\vec{x}_i - \vec{m}_{ki}(t)\right)$$

   d. Se $\mathbf{x}_i$ no pertenece a la clase *k*, alejar el prototipo $\mathbf{m}_i$ del ejemplo de entrenamiento $\mathbf{x}_i$:

$$\vec{m}_{ki}(t+1) = \vec{m}_{ki}(t) - \alpha\left(\vec{x}_i - \vec{m}_{ki}(t)\right)$$

3. Repetir el paso 2, reduciendo la velocidad de aprendizaje $\alpha(t)$ hasta 0 en cada iteración.

Entrenamiento / Procesamiento en LVQ

- Condición de terminación del entrenamiento:
  - Alcanzar un nº determinado de iteraciones.
  - O un valor bajo en la velocidad de aprendizaje.

- Procesamiento:
  - Para cada patrón, selecciona la neurona con un vector de pesos más similar, y asigna el patrón a la clase asociada a dicha neurona.

LVQ: SOM en problemas de clasificación

- Cada neurona representa una clase y define su frontera.
- Cada patrón de entrada se asigna a la clase de su neurona más cercana.
- Entrenamiento en dos etapas:
  1) Etapa no supervisada (SOM convencional).
  2) Etapa supervisada de ajuste fino de los pesos de las neuronas para mejorar la clasificación incorrecta
- La actualización de pesos es una versión incremental del aprendizaje por minimización del error (regla delta).

Aprendizaje

- **Método LVQ-1**

$$\mathbf{m}_j(t+1) \longleftarrow \mathbf{m}_j(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_j(t)] \qquad \textit{\{Premio a } \mathbf{m}_j(t)\textit{\}}$$

$$\mathbf{m}_i(t+1) \longleftarrow \mathbf{m}_i(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \qquad \textit{\{Castigo a } \mathbf{m}_i(t)\textit{\}}$$

- **Premio**: Si la clase de $\mathbf{m}_c(t)$, coincide con la de $\mathbf{x}(t)$
- **Castigo**: En otro caso, $\mathbf{m}_c(t)$ se aleja de $\mathbf{x}(t)$.

LQV-1

- Tiende a mover los prototipos hacia prototipos de aprendizaje de su misma clase y a alejarlos de los de otra clase
- Recomendable fijar un valor peqeño para $\alpha(0)$, bastante menor que 0.1 (0.02 ó 0.03).
- Número de prototipos r :

$$50\boldsymbol{N}_p \;<\; r \;<\; 200\boldsymbol{N}_p$$

  - *El valor de* r *no es muy importante si el conjunto inicial es de buena calidad (previamente editado).*

LVQ-1 Optimizado (OLVQ-1)

- Cada prototipo tiene su propia velocidad de aprendizaje:

$$\alpha_i(t) = \frac{\alpha_i(t-1)}{1 \pm \alpha_i(t-1)}$$

- Signo positivo si Clase ($\mathbf{m}_i(t)$) = Clase ($\mathbf{x}(t)$). Los prototipos situados en los centros de los agrupamientos reducen pronto su velocidad de aprendizaje, y luego no se modifican mucho.

- Negativo si **Clase** ($\mathbf{m}_i(\boldsymbol{t})$) ≠ **Clase** ($\mathbf{x}(t)$): Los prototipos de las fronteras aumentan su velocidad de aprendizaje para acercarse a los centros de los agrupamientos.

OLVQ-1

- Para evitar un crecimiento de α_c se establece un valor máximo: $(\alpha_c)_{máx} = 0.3$

- Se desestabiliza para valores altos de r :
  - Para evitarlo, se usa $30N_p < r < 50N_p$ (usualmente, $r = 40N_p$).
  - En este caso, converge rápidamente.

- El aprendizaje termina cuando no hay clasificaciones incorrectas durante un nº prefijado de iteraciones.

LVQ-2.1

- El patrón **x** modifica dos prototipos ($\mathbf{m}_i$, el más cercano de la misma clase que **x** y $\mathbf{m}_j$, el más cercano a **x** de distinta clase).
- Esta modificación sólo se realiza si **x** se encuentra en una ventana en torno al punto medio de $\mathbf{m}_i$ y $\mathbf{m}_j$. Esto ocurre si:

$$\min\left(\frac{\|m_i - x\|}{\|m_j - x\|}, \frac{\|m_j - x\|}{\|m_i - x\|}\right) > s = \frac{1-w}{1+w}$$

- Donde *w* es el ancho relativo de la ventana, usualmente *w = 0.3, s = 0.54.*

$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)(\mathbf{x} - \mathbf{m}_i(t))$ Premia a $\mathbf{m}_i(t)$

$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) - \alpha(t)(\mathbf{x} - \mathbf{m}_j(t))$ Castiga a $\mathbf{m}_j(t)$

LVQ-2.1

- Estrategia inestable para valores de *r* elevados:
  - Aleja los prototipos
  - No aproxima las funciones de densidad.
- Solución:
  - Usar $\alpha(0)$ bajo (0.02) para evitar la inestabilidad.
  - Usar *r* bajo: $30N_p < r < 200N_p$

LVQ-3

Persiste la idea de ventana.

Modifica los dos patrones más cercanos:

3. Si $\mathbf{m}_i$ y $\mathbf{m}_j$ son de la distinta clase LVQ2.1
4. Si $\mathbf{m}_i$ y $\mathbf{m}_j$ son de misma clase, se premia a ambos:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \varepsilon\alpha(t)(\mathbf{x} - \mathbf{m}_i(t))$$

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) - \varepsilon\alpha(t)(\mathbf{x} - \mathbf{m}_j(t))$$

$0.1 < \varepsilon < 0.5$ → Factor de estabilización: Tanto más bajo cuanto más pequeño sea el valor de *w* (más estrecha la ventana)

LVQ-3

Persiste la idea de ventana.

Modifica los dos patrones más cercanos:

3. Si $\mathbf{m}_i$ y $\mathbf{m}_j$ son de la distinta clase LVQ2.1
4. Si $\mathbf{m}_i$ y $\mathbf{m}_j$ son de misma clase, se premia a ambos:

$$\mathbf{m}_i(t+1) \longleftarrow \mathbf{m}_i(t) + \varepsilon\alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)]$$

$$\mathbf{m}_j(t+1) \longleftarrow m_j(t) + \varepsilon\alpha(t)[\mathbf{x}(t) - \mathbf{m}_j(t)]$$

$0.1 < \varepsilon < 0.5$ → Factor de estabilización: Tanto más bajo cuanto más pequeño sea el valor de w (más estrecha la ventana)

Comparación entre variantes de LVQ

- Las distintas estrategias (LVQ1, OLVQ1, LVQ2.1 y LVQ3) proporcionan resultados similares, con ligeras variaciones entre problemas.

- LVQ1 requiere menos parámetros, por lo tanto es la más recomendable.

Ventajas de LVQ

- Entrenamiento rápido.
- Posibilidad de actualización incremental.
- Robustez, capacidad de generalización.
- Capaz de generar conjuntos pequeños de prototipos representativos.

Inconvenientes de LVQ

- Basado en medidas de distancia
- Muy dependiente de:
  - Inicialización de los prototipos.
  - Parámetros (velocidad de aprendizaje η y nº de épocas de entrenamiento).
  - Patrones de entrenamiento seleccionados.
- El nº ideal de prototipos *r* es difícil de determinar.