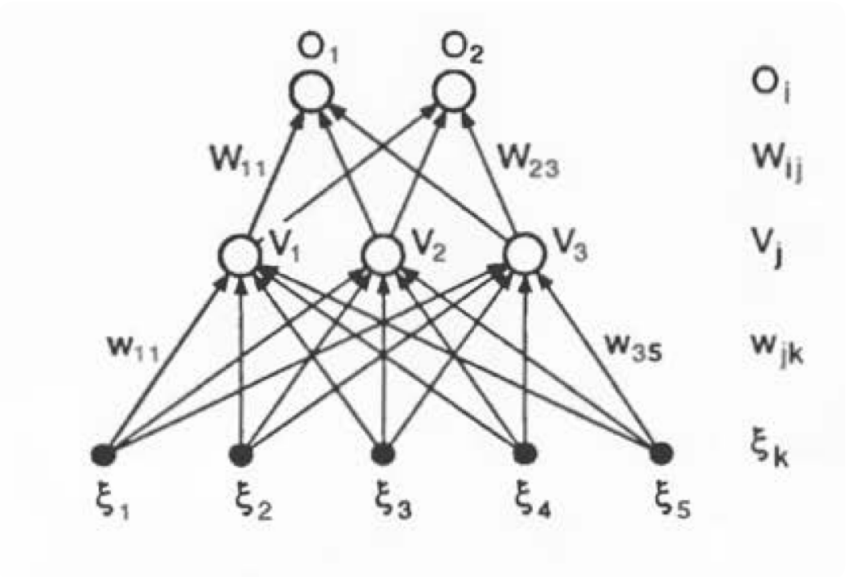
REDES MULTI CAPAS

Esta será una de las clases más importantes de este curso. Veremos como usar redes multi-capas, con $M \geq 2$. O sea, tenemos los N elementos de entrada que definen un vector en $\mathbb{R}^N$

$$\vec{\xi} = (\xi_1, \xi_2, \xi_3, \ldots, \xi_N),$$

Tenemos M neuronas de salida y al menos una capa oculta. Consideremos el caso de una capa oculta ($M=2$) y luego será fácil generalizarlo. El gráfico siguiente nos da un esquema general



Las capas se organizan de abajo hacia arriba

Una vez que la red lee los N entradas ξ_k podemos calcular los estados de las neuronas V_j de la capa oculta.

$$V_j^\mu = g(h_j^\mu) = g\left(\sum_k w_{jk}\,\xi_k^\mu\right)$$

Las variables de entrada ξ_k^μ pueden ser 0 o 1, +1 o -1, o puede tomar valores continuos, dependiendo de la forma de $g(h)$.

Ahora con los V_j^μ podemos calcular las neuronas de salida

$$\begin{aligned} O_i^\mu &= g(h_i^\mu) \\ &= g\left(\sum_j W_{ij}\,V_j^\mu\right) \\ &= g\left(\sum_j W_{ij}\,g\left(\sum_k w_{jk}\,\xi_k^\mu\right)\right) \end{aligned}$$

Noten que ahora no podemos reducir la red en subredes simples con 1 neurona de salida.

La función costo es la misma

$$E(\vec{w}, \vec{W}) = \frac{1}{2} \sum_{j\mu} \left(\zeta_i^\mu - O_i^\mu \right)^2$$

$$= \frac{1}{2} \sum_{j\mu} \left[\zeta_i^\mu - g\left(\sum_j W_{ij}\, g\left(\sum_k w_{jk} \xi_k^\mu \right) \right) \right]^2$$

Si tenemos N entradas

L neuronas ocultas en la 1° capa

M neuronas de salida en la 2° capa

tendremos

$N \times L$ elementos w_{jk}

$L \times M$ elementos W_{ij}

Para generalizar el método de descenso por el gradiente para encontrar un conjunto $(\vec{w}, \vec{W})$ tal que $E(\vec{w}, \vec{W}) = 0$ deberemos encontrar el gradiente de E, que será un campo vectorial

$$\nabla E : \mathbb{R}^{L(N+M)} \longrightarrow \mathbb{R}^{L(N+M)}$$

Como ven, es un problema al menos complicado

$$\nabla E = \left(\frac{\partial E}{\partial w_{11}}, \frac{\partial E}{\partial w_{12}}, \dots \frac{\partial E}{\partial w_{NL}}, \frac{\partial E}{\partial W_{11}}, \dots, \frac{\partial E}{\partial W_{LM}} \right)$$

$N \times L$ $\qquad$ $L \times M$

$N \times L + L \times M = L(N+M)$ sinapsis ¡!

Vamos a introducir ahora un algoritmo muy famoso e importante que acompaña a todas las redes feed-forward, llamado BACK-PROPAGATION

BACK-PROPAGATION

El algoritmo nos brinda un método para cambiar los $L(N+M)$ pesos sinápticos en cualquier red feed-forward multi capas. La base es el método de descenso por el gradiente que vimos para el perceptron simple con salida lineal o no-lineal (continuos).

Bryson y Ho		1969
Werbos		1974
Parker		1985
Hinton, Rumelhart y Williams,		1985 ← nuestro enfoque
Le Cun		1985

Tenemos P relaciones input-output etiquetadas

$$\vec{\xi}^{\mu} \longrightarrow \vec{\zeta}^{\mu}$$

donde $\vec{O}^{\mu} = \vec{\zeta}^{\mu}$ es la salida esperada cuando el input es $\vec{\xi}^{\mu}$.

Elegimos todas las sinapsis $\{W_{jk}\}$ y $\{W_{ij}\}$ de forma aleatoria (como queramos).

Le presentamos a la red los P patrones del conjunto de entrenamiento $\vec{\xi}^{\mu}$ uno tras otro, en forma secuencial o aleatoria y acumulamos el costo. Luego usamos descenso por el gradiente para corregir los W_{ij}

$$W_{ij}^{\text{nuevo}} = W_{ij}^{\text{viejo}} + \Delta W_{ij}$$

con

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}}$$

$$= +\eta \sum_{\mu} [\zeta_i^{\mu} - O_i^{\mu}] g'(h_i^{\mu}) V_j^{\mu}$$

$$\Delta W_{ij} = \eta \sum_{\mu} \delta_i^{\mu} V_j^{\mu}$$

$$\text{con} \quad \delta_i^{\mu} = g'(h_i^{\mu})(\xi_i^{\mu} - O_i^{\mu})$$

Una vez que hemos actualizado los W_{ij} pasamos a la primera capa de sinapsis.

$$w_{jk}^{\text{nuevo}} = w_{jk}^{\text{viejo}} + \Delta w_{jk}$$

con

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

$$= -\eta \sum_{\mu} \frac{\partial E}{\partial V_j^{\mu}} \cdot \frac{\partial V_j^{\mu}}{\partial w_{jk}}$$

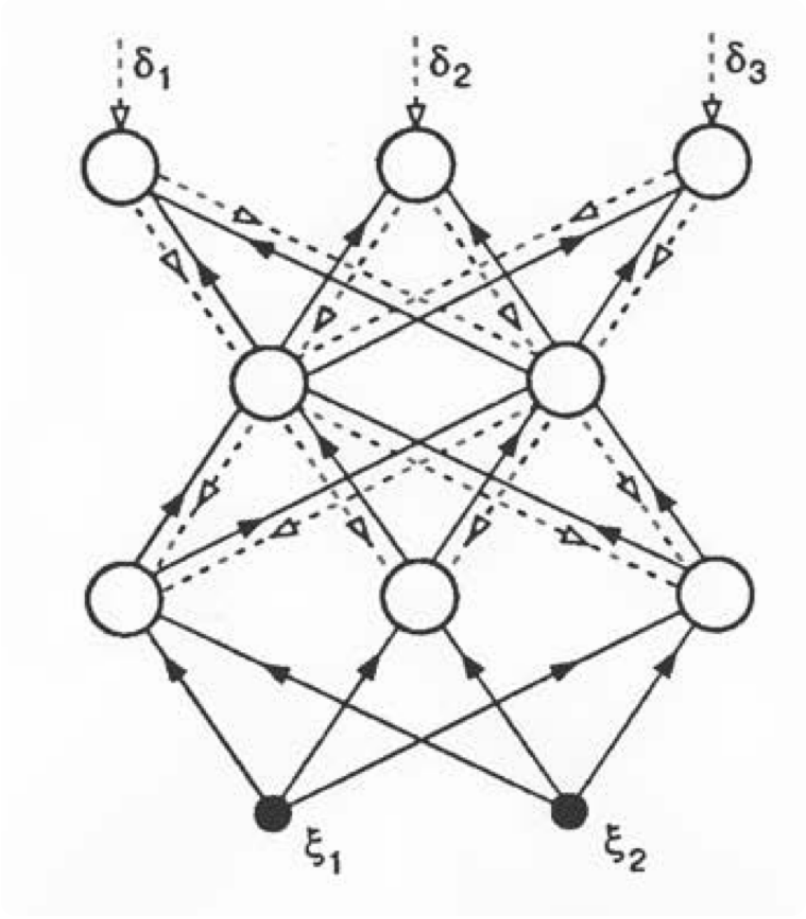
$$= \eta \sum_{\mu i} [\xi_i^{\mu} - O_i^{\mu}] g'(h_i^{\mu}) W_{ij} g'(h_j^{\mu}) \xi_k^{\mu}$$

$$= \eta \sum_{\mu i} \delta_i^{\mu} W_{ij} g'(h_j^{\mu}) \xi_k^{\mu}$$

$$\Delta w_{jk} = \eta \sum_{\mu} \delta_j^{\mu} \xi_k^{\mu}$$

con $$\delta_j^{\mu} = g'(h_j^{\mu}) \sum_i W_{ij} \delta_i^{\mu}$$

Se llama back-propagation porque la información viaja hacia adelante y las correcciones hacia atrás.



Notar que hay una forma general

$$\Delta w_{pq} = \eta \sum_{\mu} \delta_{\text{salida}} \times V_{\text{input}}$$

entre dos capas subsiguientes p y q.

Podemos actualizar

* después de cada patrón μ : on line
* después de p patrones : batch.

Podemos mezclar diferentes tipos de funciones de activación.

Podemos tener muchas capas.

El conjunto de entrenamiento puede tener diferentes tamaños

Al a lago de la actualización, cada vez que terminamos de actualizar 1 vez cada uno de los p elementos de conjunto de entrenamiento (una época) monitoreamos la función la evolución de la función costo del entrenamiento

Si nos hemos guardado un conjunto de q elementos etiquetados correctamente podemos ver como evoluciona la función costo de testeo

Resumamos: Consideramos que tenemos muchas capas $m = 0, 1, 2, \dots M$. 0 corresponde a la entrada que no es una capa.

1. Elegimos todos los pesos sinápticos entre todas las capas y entre la entrada y la 1ª capa. Los escogemos aleatorios y pequeños.

2. Tomamos un patrón ξ^μ_k y lo ponemos en la entrada

$$V^{(0)}_k = \xi^\mu_k \qquad \forall\, k$$

3. Propagamos la señal hacia adelante

$$V^{(m)}_i = g\left(h^{(m)}_i\right) = g\left(\sum_j \omega^{(m)}_{ij}\, V^{(m-1)}_j\right)$$

para cada i de la capa i-ésima y hasta llegar a la última capa $V^{(\bar{m})}_i = O_i$

4. Calculamos los deltas de la salida

$$\delta_i^{(\bar{M})} = g'\left(h_i^{(\bar{M})}\right)\left[\xi_i^{\mu} - V_i^{(\bar{M})}\right]$$

comparando lo obtenido con lo deseado.

5. Calculamos los δ de las capas anteriores, en orden decreciente

$$\delta_i^{(m-1)} = g'\left(h_i^{(m-1)}\right) \sum_j w_{ji}^{(m)} \delta_j^{(m)}$$

para $m = \bar{M}-1, \bar{M}-2, \ldots 2$ para c/neurona de cada capa.

6. Aplicamos la regla

$$w_{ij}^{(m)\,\text{nuevo}} = w_{ij}^{(m)\,\text{viejo}} + \Delta w_{ij}^{(m)}$$

con $\Delta w_{ij}^{(m)} = \eta\, \delta_i^{(m)} V_j^{(m-1)}$

7. Volvemos al punto (2)

MOMENTO

$$\Delta w_{ij}^{(m)}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}^{(m)}} + \alpha \Delta w_{ij}^{(m)}(t)$$

Cuando estamos en una zona muy plana

$$\Delta w_{ij}^{(m)}(t+1) \approx \Delta w_{ij}^{(m)}(t)$$

$$\Delta w_{ij}^{(m)}(t+1)(1-\alpha) \approx -\eta \frac{\partial E}{\partial w_{ij}^{(m)}}$$

$$\Delta w_{ij}^{(m)} \approx -\underbrace{\frac{\eta}{(1-\alpha)}}_{\eta \text{ efectivo}} \frac{\partial E}{\partial w_{ij}^{(m)}}$$

PARAMETRO ADAPTATIVO

k pesos

$$\Delta\eta = \begin{cases} +\alpha & \text{si } \Delta E < 0 \text{ consistentemente} \\ -b\eta & \text{si } \Delta E > 0 \\ 0 & \text{sino} \end{cases}$$

OTROS MÉTODOS DE APROXIMACIÓN

no lo doy pero esta en la sección 6.2 del capítulo 6 del libro de Hertz et al.

APLICACIONES

XOR

PARITY FUNCTION

⋮

TEOREMA UNIVERSAL DE LA APROXIMACIÓN

Una red feed-forward con una capa oculta ($M=2$) que contiene un número finito de neuronas puede aproximar funciones continuas sobre un subconjunto compacto de $\mathbb{R}^n$, para funciones de activación con buenas propiedades.

1989 George Cybenko (sigmoides)

En 2017 Lu et al probaron esto para redes profundas. $(4+n)$ con ReLU. EXPLICAR