Redes Neuronales

Perceptrón multicapa I

Facultad de Ingeniería. Universidad del Valle

Septiembre de 2018

Contenido

1. Preceptrón multicapa (MLP)
2. Algoritmo BackPropagation (BP)

Universidad del Valle

Contenido

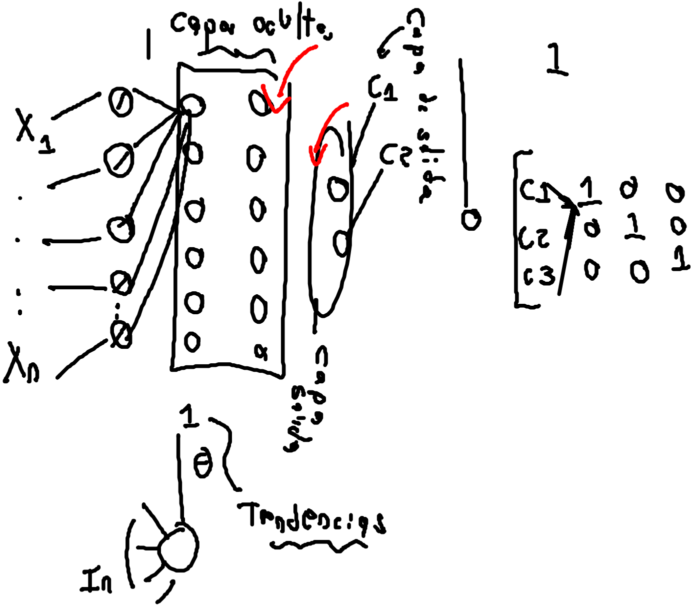
1 Preceptrón multicapa (MLP)

2 Algoritmo BackPropagation (BP)

Perceptrón multicapa

Definición

- Está compuesta por capas de entrada, capas ocultas y capas de salida
- La señal de entrada se propaga hacia adelante entre las distintas capas
- Es una generalización del perceptrón de una capa
- Pueden solucionar problemas más complejos
- El algoritmo más común de entrenamiento de el algoritmo de propagación hacia atrás (*back-propagation*) que se basa en la regla de entrenamiento de corrección del error



Capa oculta
X_1
X_n
C_1
C_2
1
1
θ
Tendencias
In
C_1 1 0 0
C_2 0 1 0
C_3 0 0 1

Perceptrón multicapa

Entrenamiento

1. **Paso hacia adelante:** La señal de entrada es aplicada y se propaga capa a capa
2. **Paso hacia atrás:** Se ajustan los pesos de cada capa utilizando la regla de corrección de error.

Perceptrón multicapa

Características

1. **Señal de activación:** Debe ser derivable, ya que en el calculo del error, debemos trabajar con la derivada de la función de activación. Las que se utilizan son función **lineal** y **sigmoide**.
2. **Capas ocultas** Pueden ser un o más capas ocultas, las cuales no están conectadas a las entradas y salidas directamente
3. **Conectividad** Está determinada por los pesos de las conexiones entre cada capa

Perceptrón multicapa

Características

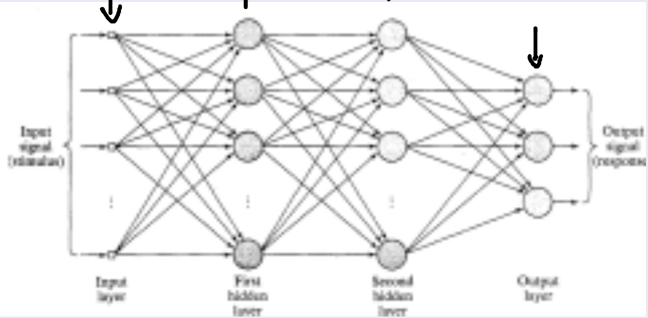


Figura: Arquitectura de MLP [Haykin, 1998]

Perceptrón multicapa

Características

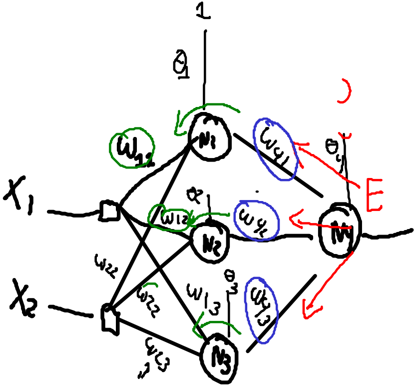
1. La computación de las entradas se puede expresar como una señal continua no lineal
2. La computación de un gradiente, es necesario para propagar el error a través de toda la red (regla de aprendizaje) y así ajustar los pesos

Universidad del Valle

Contenido

1 Preceptrón multicapa (MLP)

2 Algoritmo BackPropagation (BP)



$$\begin{cases} Neta_{N1} = X_1 w_{11} + X_2 w_{12} + \theta_1 \\ \vdots \\ Neta_{N3} = X_1 w_{13} + X_2 w_{23} + \theta_3 \end{cases}$$

$$S_{N1} = F(Neta_{N1})$$

$$S_{N3} = F(Neta_{N3})$$

$$Neta_{N4} = w_{41} S_{N1} + w_{42} S_{N2} + w_{43} S_{N3} + \theta_4$$

$$S = F(Neta_{N4})$$

$$E = \frac{(t - F(Neta_{N4}))^2}{2}$$

Algoritmo BackPropagation

Descripción

- La señal de error de una neurona j en una iteración n es definida por

$$e_j(n) = t_j(n) - y_j(n)$$

- Se toma como error de una capa c como el error cuadrático medio

$$\epsilon(n) = \frac{1}{2} \sum_{j \in c} e_j^2(n)$$

- Y el error global de toda la red, donde M es el conjunto de capas

$$\epsilon(n) = \frac{1}{2} \sum_{c \in M} \sum_{j \in c} e_j^2(n)$$

Algoritmo BackPropagation

Capa de salida

- Se busca el error mínimo, mediante el **gradiente descendiente**

$$E = \frac{1}{2}\left(t - F(Neta)\right)^2 \qquad \boxed{\frac{\partial E_j}{\partial w_{ij}}} \quad t'\ F'(\ )$$

Realizamos los cálculos respectivos y obtenemos:

$$t \cdot t \cdot x\, w_{ij}$$

$$\frac{\partial E_j}{\partial w_{ij}} = -(t-y)f'(Neta) * x_j$$

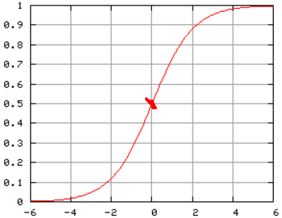
Donde f es la función sigmoide, x_i es la entrada i y *Neta* es la entrada total que recibe la neurona

Sigmoide

$$Sig = \frac{1}{1+e^{-x}} \qquad Sig = (\underbrace{1+e^{-x}})^{-1}$$

$$-(1+e^{-x})^{-2} \times (-)\, e^{-x}$$

$$(1+e^{-x})^{-2}\, e^{-x}$$



$$\frac{1}{(1+e^{-x})^{2}}\, e^{-x} \longrightarrow \overbrace{(Sig)^{2} e^{-x}}$$

Relu

$$F(x) = \begin{cases} x & si\ x > 0 \\ 0 & si\ x \leq 0 \end{cases} \quad x \geq 0 \qquad F'(x) = 1$$

$$\mathrm{Sig}' \quad \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\boxed{\mathrm{Sig}(1-\mathrm{Sig})}$$

$$\frac{1}{1+e^{-x}}\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)$$

$$\frac{1}{1+e^{-x}}\left(\frac{1+e^{-x}-1}{1+e^{-x}}\right) = \frac{1}{1+e^{-x}}\left(\frac{e^{-x}}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Algoritmo BackPropagation

Capa de salida

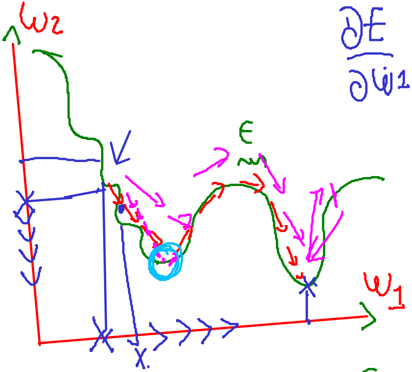
- El proceso de entrenamiento buscar modificar el peso w_{ij} de acuerdo al error calculado de la siguiente forma:

$$w_{ij}(n+1) = w_{ij}(n) + \epsilon(-\frac{\partial E_j}{\partial w_{ij}})$$

De aquí se obtiene

Regla Delta

$$w_{ij}(n+1) = w_{ij}(n) + \epsilon(t-y)f'(Neta) * x_j)$$



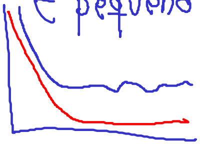
$$\frac{\partial E}{\partial w_1} = 0$$

ε

$$\frac{\partial E}{\partial w_2} = 0$$

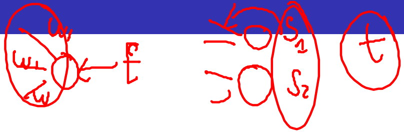
ε cuanto converge ma rapido

ε pequeño



ε grande

Algoritmo BackPropagation



Capa de salida

- Si la función de activación es lineal, se obtiene que la derivada es 1, por lo que la variación del peso será:

$$w_{ij}(n+1) = w_{ij}(n) + \epsilon(t-y) * x_j$$

- Si es la función sigmoide $s = \frac{1}{1+e^{neta}}$

Pari (S,g)

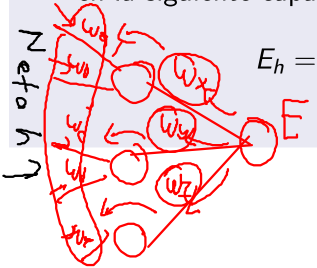
$$w_{ij}(n+1) = w_{ij}(n) + \epsilon * (t-y)s(1-s) * x_j$$

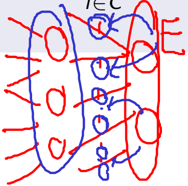
Algoritmo BackPropagation

Capa oculta

- La actualización de los pesos depende del error de las capas ocultas siguientes y de salida
- El error de la capa oculta h y se tiene el conjunto C neuronas en la siguiente capa.

$$E_h = f'(Neta_h) \sum_{i \in C} E_{(h+1)} w_{(h+1)i}$$





Universidad del Valle

Algoritmo BackPropagation

Descripción

- Se utiliza un conjunto de patrones para entrenar la red 80% 20%
- F Se aplica la entrada a la red y se calcula la salida total
- B Se calcula el error entre el valor deseado y la salida
- Se propaga el error hacia atrás, es decir que el error de la capa n se basa en el error de la capa $n+1$
- Se modifican los pesos de las capas ΔW. Este calculo depende de la capa siguiente.
- Se verifica la condición de parada ← Max. itr, Error Aceptado

Universidad del Valle

Algoritmo BackPropagation

Algoritmo

1. Se inicializan los pesos del MLP entre [-1,1]
2. Mientras la condición de parada sea falsa se repiten los pasos 3 a 12
3. Se aplica la entrada
4. Se calcula los valores netas para la capa oculta h

$W_i X_i$

$$Neta^h = \sum_{i=1}^{N} + \Theta_k$$

Se supone que la capa h tiene N neuronas

Algoritmo BackPropagation

Algoritmo

1. Se inicializan los pesos del MLP entre [-1,1]
2. Mientras la condición de parada sea falsa se repiten los pasos 3 a 12
3. Se aplica la entrada
4. Se calcula los valores de entrada netos para la capa oculta h

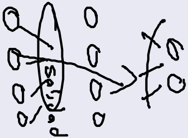
$$Neta^h = \sum_{i=1}^{N} + \Theta_k$$

Se supone que la capa h tiene N neuronas

Algoritmo BackPropagation

Algoritmo

5. Se calcula la salida de la capa oculta



$$y_h = f_h(Neta_h)$$

6. Calculamos los valores netos de entrada para la capa de salida

$$Neta = \sum_{j=1}^{L} w_{kj} y_h + \Theta_j$$

7. Calculamos la salida de la red

Universidad del Valle

Algoritmo BackPropagation

Algoritmo

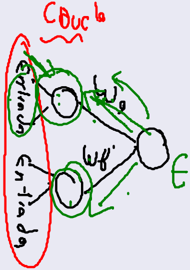
8. Calculamos la salida de la red
9. Calculamos los términos de error para la capa de salida

$$E^o = (t_u - y_u) f'(Neta_o)$$

10. Estimamos el error para las capas ocultas

$$E_{mh} = F'(Entrada_1) E w_o$$

$$E^h = f'(Neta_h) \sum_{k=1}^{M} E_i^o w_{kj}$$



Como se puede observar, el error de la capa oculta depende de la siguiente capa

Algoritmo

10. Actualizamos los pesos en la capa de salida

$$w^o(n+1) = \epsilon E^o x_i + w^o(n)$$

11. Actualizamos los pesos en la capa(s) oculta(s)

$$w^h(n+1) = \epsilon E^h x_i + w^h(n)$$

12. Verificamos si el error global cumple la condición de finalizar (un error mínimo) o un número de iteraciones

$$E_p = \frac{1}{2P} \sum_{p=1}^{P} \sum_{k=1}^{M} (t-y)^2$$

m neurona de salida

Ejemplo: Función XOR

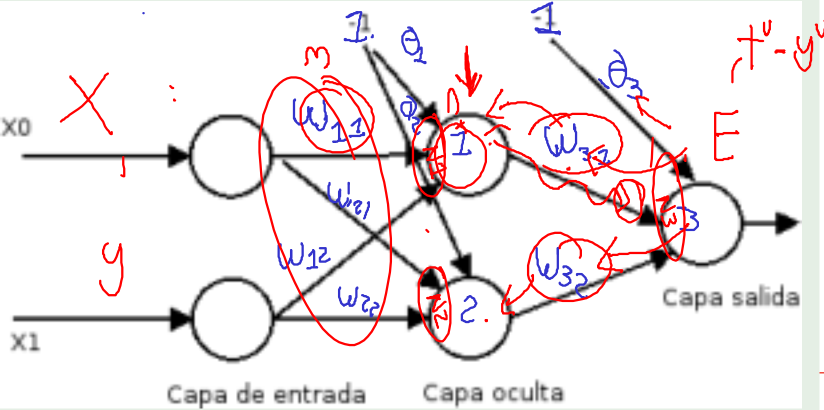
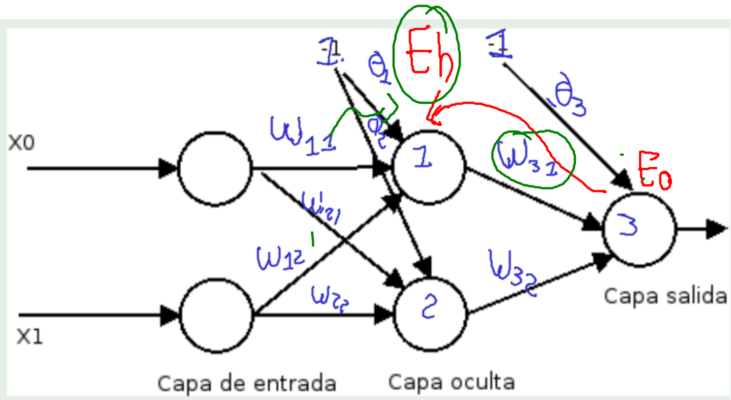


Figura: Arquitectura de ejemplo

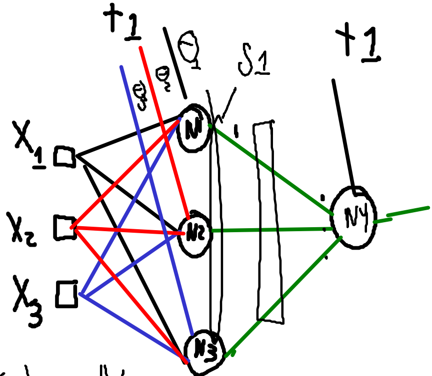
Sig



$W_{11} = 0.4$ $\quad W_{12} = 0.6$ $\quad W_{21} = -0.2$ $\quad W_{22} = 0.1$

$\theta_1 = 0.5$ $\quad \theta_2 = 0.8$

$W_{31} = 0.6$ $\quad W_{32} = 0.8$ $\quad \theta_3 = -0.1$



$$
\begin{array}{cc}
 & \begin{array}{ccc} X_1 & X_2 & X_3 \end{array} \\
P_1 = \begin{array}{c} N_1 \\ N_2 \\ N_3 \end{array} & \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}
\end{array}
$$

$$P_{1x} \cdot X^T + \Theta_i(1) = N_1$$

$$
\begin{array}{l}
E_{N_1} \leftarrow \\
E_{N_2} \leftarrow \\
E_{N_3} \leftarrow
\end{array}
\begin{bmatrix}
w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + \Theta_1 \\
w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + \Theta_2 \\
w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + \Theta_3
\end{bmatrix}
$$

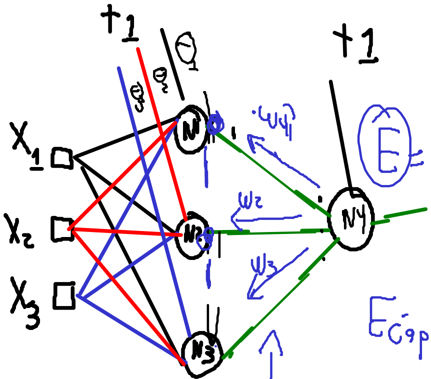
$$S_1 = F(N_1)$$

$$S_1 = \begin{bmatrix} F(E_{N_1}) \\ F(E_{N_2}) \\ F(E_{N_3}) \end{bmatrix}_{3\times 1}$$

$$P_2 = \begin{bmatrix} w_{41} & w_{42} & w_{43} \end{bmatrix}_{1\times 3}$$

$$P_{2x} S_1 := F(E_{N_1}) \times W_{41} + F(E_{N_2}) \times W_{42} + F(E_{N_3}) W_{43} \Rightarrow E_{N_4 \, 1\times 1}$$

$$S_{4|1,1} = F(E_{N_{4\times 1}} + \Theta_4)$$



$$y^u = F(.wX \quad)$$

ECM

$$E = t^u - y^u$$

$$\frac{1}{2}(t^u - y^u)^2$$

$$(t^u - y^u)$$

$$E_{capa\,S} = E$$

$$E_{oculta} = \begin{bmatrix} w_{41} & w_{42} & w_{43} \\ & & \end{bmatrix} E \times f'(S_1)$$

$$S_1 = \begin{bmatrix} F(EN_1) \\ F(EN_2) \\ F(EN_3) \end{bmatrix}$$

$$W(t+1) = W(t) + \eta \times E_c \times X$$

Actualización

Capa Salida

$$P_2 = P_2 + \eta E \times E_{N4 \times 1}$$

$$\underset{3\times3}{P_1} = P_1 + \eta \underbrace{(E_{\text{oculta}})^T_{3\times1} \times X_{1\times3}}_{3\times3}$$

$$A_{pq} \, B_{qr} = C_{pr}$$

$$AB \neq BA$$

Referencias I

- Eduardo, C. and Jesus Alfonso, L. (2009).
  Una aproximación práctica a las redes neuronales artificiales.
  Colección Libros de Texto. Programa Editorial Universidad del Valle.
- Haykin, S. (1998).
  Neural Networks: A Comprehensive Foundation (2nd Edition).
  Prentice Hall.
- Widrow, B. and Winter, R. (1988).
  Neural nets for adaptive filtering and adaptive pattern recognition.
  Computer, 21(3):25–39.

¿Preguntas?

Próximo tema:
Perceptrón multicapa II