Redes Neuronales

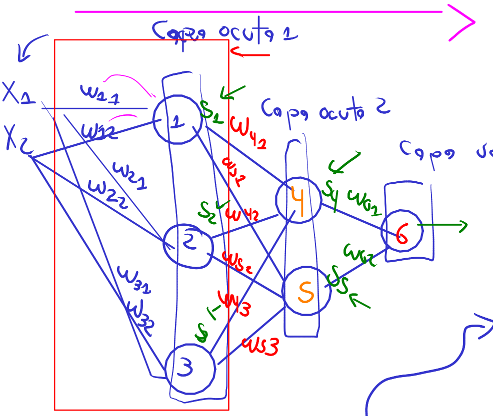
Aprendizaje supervisado I

carlos.andres.delgado@correounivalle.edu.co

Carlos Andrés Delgado S.

Universidad San Buenaventura, Cali

Febrero de 2021



$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$X_1 = w_{11}X_1 + w_{12}X_2$$

$$X_2 = w_{21}X_1 + w_{22}X_2$$

$$X_3 = w_{32}X_1 + w_{32}X_2$$

Contenido

1 Preceptrón multicapa (MLP)

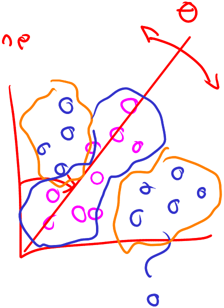
2 Algoritmo BackPropagation (BP)

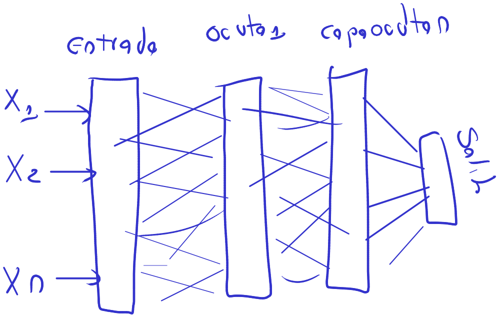
Contenido

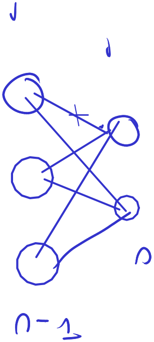
1. Preceptrón multicapa (MLP)
2. Algoritmo BackPropagation (BP)

Perceptron

Adaline




Entrada
oculta 1
capa oculta n
X_1
X_2
X_n
Salida


n
n - 1

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

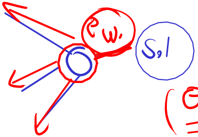
Perceptrón multicapa

Entrenamiento

$\overline{X}$

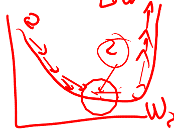
1. **Paso hacia adelante:** La señal de entrada es aplicada y se propaga capa a capa
2. **Paso hacia atrás:** Se ajustan los pesos de cada capa utilizando la regla de corrección de error.

$w_2 \;\; \alpha \frac{\Delta e}{\Delta w} \to 0$



$y - T$

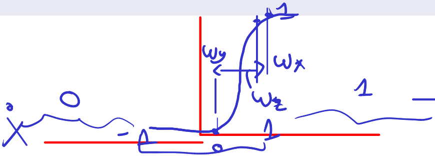
(error)

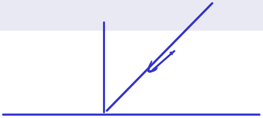


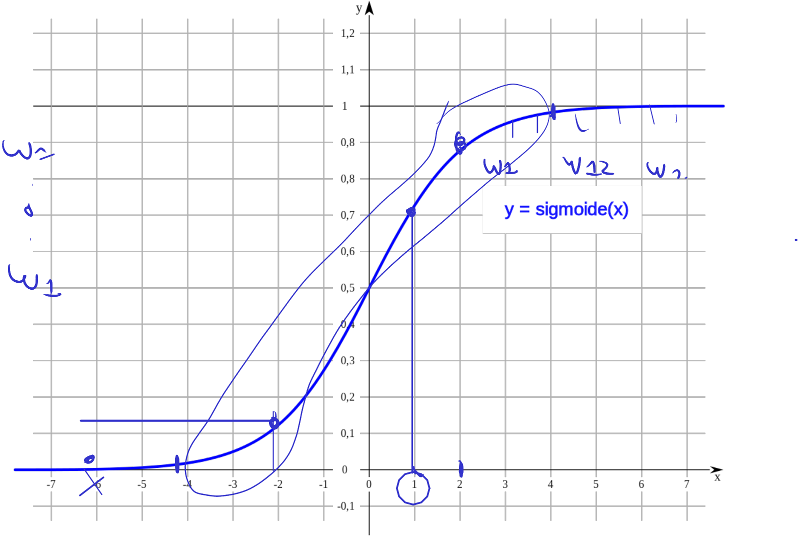
Perceptrón multicapa

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






y = sigmoide(x)
y
x
1,2
1,1
1
0,8
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
-0,1
-7
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
7

Perceptrón multicapa

Características

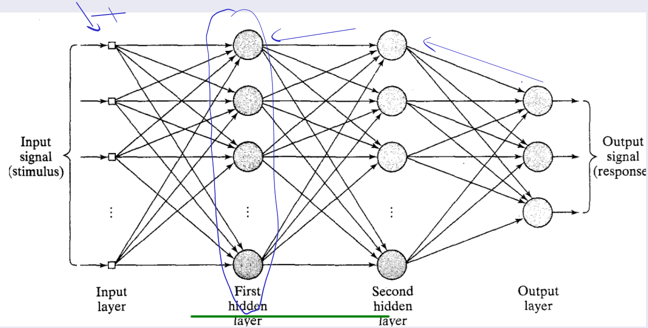


Figura: Arquitectura de MLP [Haykin, 1998]

Entrada neta

Perceptrón multicapa

Características

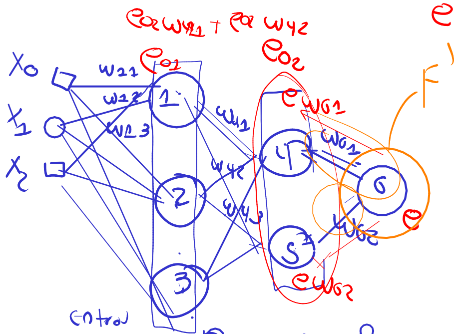
1. La computación de las entradas se puede expresar como una señal continua no lineal
2. La computación de un gradiente, es necesario para propagar el error a través de toda la red (regla de aprendizaje) y así ajustar los pesos

Contenido

1 Preceptrón multicapa (MLP)

2 Algoritmo BackPropagation (BP)

$e W_{01} + e W_{02}$



$\boxed{F(W_{01} S_{04} + W_{02} S_{ns})}$

Pesos C01 | Pesos C02 | Pesos CS

$$\begin{array}{c} \text{neuron} \\ \\ \\ \end{array} \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix} \qquad \begin{bmatrix} W_{41} & W_{42} & W_{43} \\ W_{51} & W_{52} & W_{53} \end{bmatrix} \qquad \begin{bmatrix} W_{01} & W_{02} \end{bmatrix}$$

$$\underset{3\times 3}{\begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix}} \underset{3\times 1}{\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}} = \underset{3\times 1}{\begin{bmatrix} S_{N1} \\ S_{N2} \\ S_{N3} \end{bmatrix}}$$

Salida Neta C01

$$\underset{2\times 3}{\begin{bmatrix} W_{41} & W_{42} & W_{43} \\ W_{51} & W_{52} & W_{53} \end{bmatrix}} \underset{3\times 1}{\begin{bmatrix} S_{N1} \\ S_{N2} \\ S_{N3} \end{bmatrix}} = \underset{2\times 1}{\begin{bmatrix} S_{NS} \\ S_{N6} \end{bmatrix}}$$

colum Fila

$$\underset{1\times 2}{\begin{bmatrix} W_{01} & W_{02} \end{bmatrix}} \underset{2\times 1}{\begin{bmatrix} S_{NS} \\ S_{N6} \end{bmatrix}} = \begin{bmatrix} S_S \end{bmatrix} = \text{Salida total}$$

Algoritmo BackPropagation

Descripción

- La señal de error de una neurona j en una iteración n es definida por
$$e_j(n) = \underline{t_j(n)} - \underline{y_j(n)}$$
- 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**

$$\frac{\partial E_j}{\partial w_{ij}} \quad \frac{1}{2}(t - y)^2 =$$

$X[0]w_{2x} + X[1]w_{?x}$

Realizamos los cálculos respectivos y obtenemos:

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

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

Algoritmo BackPropagation

Capa de salida

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

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

Factor ent...

De aquí se obtiene

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

ϵ

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 $s = \frac{1}{1+e^{neta}}$

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

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.

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

Algoritmo BackPropagation

Descripción

- Se utiliza un conjunto de patrones para entrenar la red
- Se aplica la entrada a la red y se calcula la salida total
- 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

num it

error

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

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

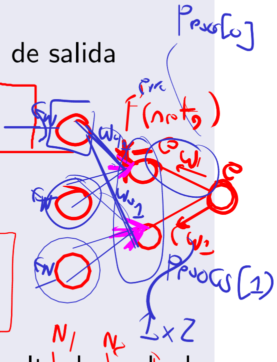
7. Calculamos la salida de la red

Algoritmo BackPropagation

Algoritmo

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

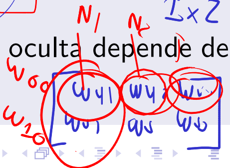
$$E^o = (t_u - y_u) f'(Neta)$$



10. Estimamos el error para las capas ocultas

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

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



$e_o W_t$

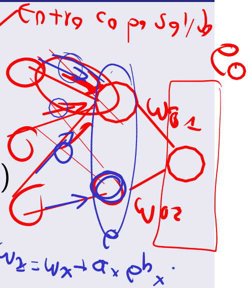
$e_{N2} = w_{41} e_c \cdot f'(Neto_2)$

Algoritmo BackPropagation

Algoritmo

10 Actualizamos los pesos en la capa de salida

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

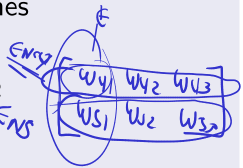


11 Actualizamos los pesos en la capa(s) oculta(s)

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

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

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



Algoritmo BackPropagation

Ejemplo: Función XOR

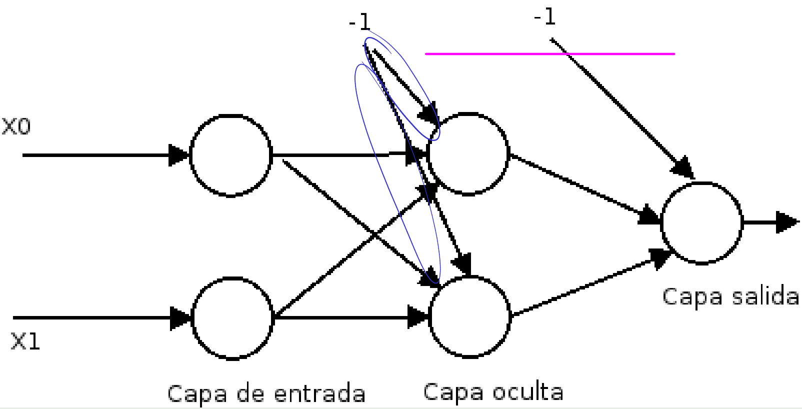


Figura: Arquitectura de ejemplo

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


0
1
1
0

E