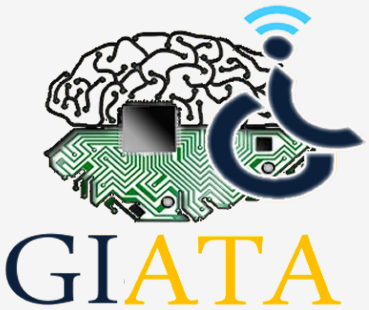
Inteligencia Artificial II: Redes Neuronales Artificiales (RNA)

(Redes Neuronales con conexiones hacia adelante)

cognitio

V. Robles-Bykbaev (vrobles@ups.edu.ec)
http://catedraunescoinclusion.org/
http://giiata.blog.ups.edu.ec/

Universidad Politécnica Salesiana,
Sede Matriz Cuenca, 2019



Contenidos

Contenidos

Introducción

Conceptos básicos y principales tipos de arquitecturas de RNA

La arquitectura de una RNA define como las múltiples neuronas que la conforman se organizan y se relacionan unas con otras. Por ello, importante considerar los siguientes aspectos [Nunes da Silva et al., 2017]:

- Pueden existir 2 RNA, la una con 17 neuronas y la otra con 34 y ambas estar organizadas bajo la misma topología.
- La organización que usa una RNA es esencial para dirigir las conexiones sinápticas de las neuronas.
- En una RNA artificial pueden existir neuronas que usen diferentes tipos de funciones de transferencia. Por ejemplo, un grupo de neuronas puede usar la función logística, mientras que otro grupo la hiperbólica.

Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

- Básicamente, una RNA se puede dividir en tres partes conocidas como **capas**:
  - *Capa de entrada:* se encarga de recibir la información (datos), señales, características o medidas del ambiente externo. Comúnmente estas entradas se normalizan de acuerdo a los límites que definen las diferentes funciones de activación o transferencia.
  - *Capa(s) oculta(s):* una RNA puede tener una o varias de estas capas. Las neuronas contenidas en estas capas se encargan de extraer los patrones asociados con el proceso o sistema analizado. Realizan la mayor parte del procesamiento interno de la red.
  - *Capa de salida:* puede contener una o varias neuronas (de acuerdo a cómo se dese representar la salida que entrega la red).

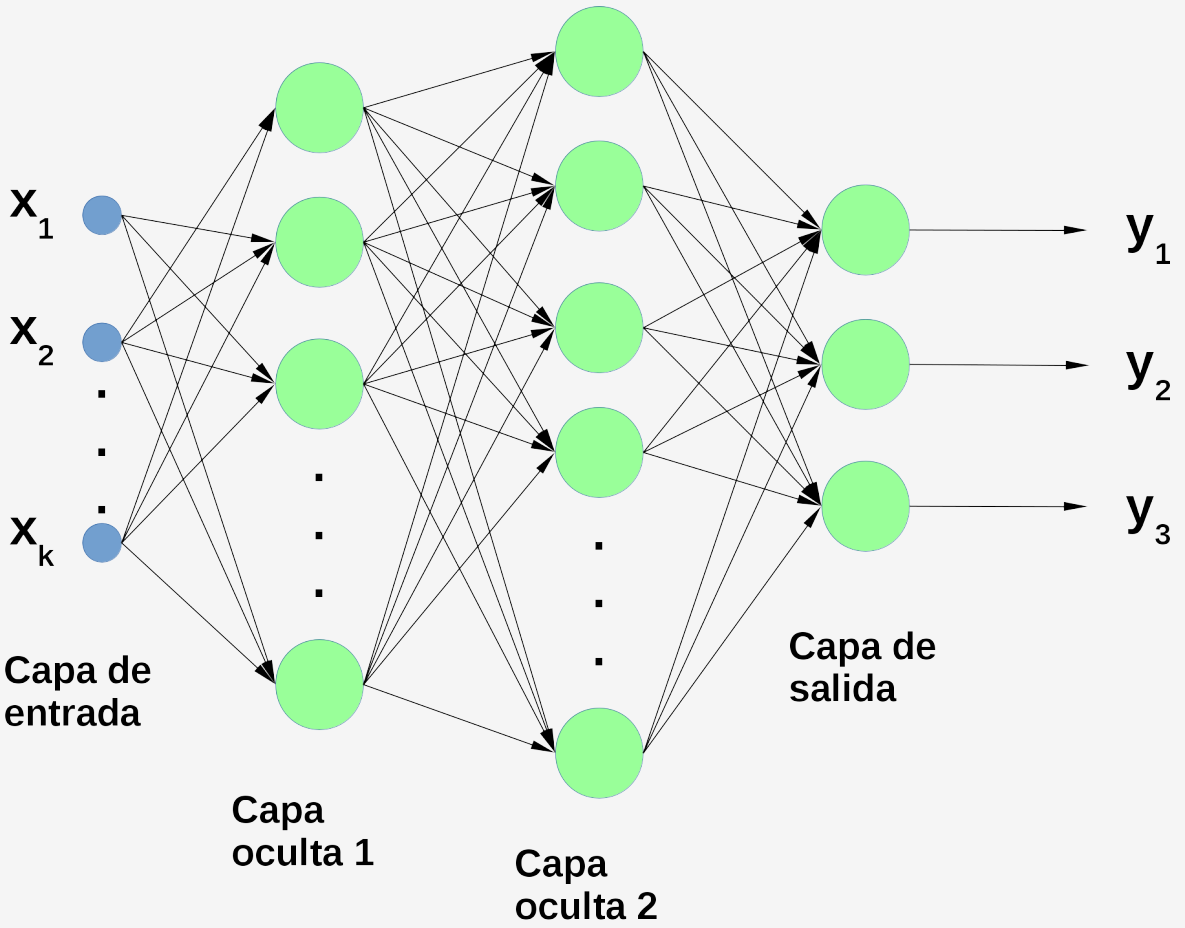
Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

Figure: Estructura de una RNA multicapa con conexión hacia adelante y 2 capas ocultas

Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA



Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

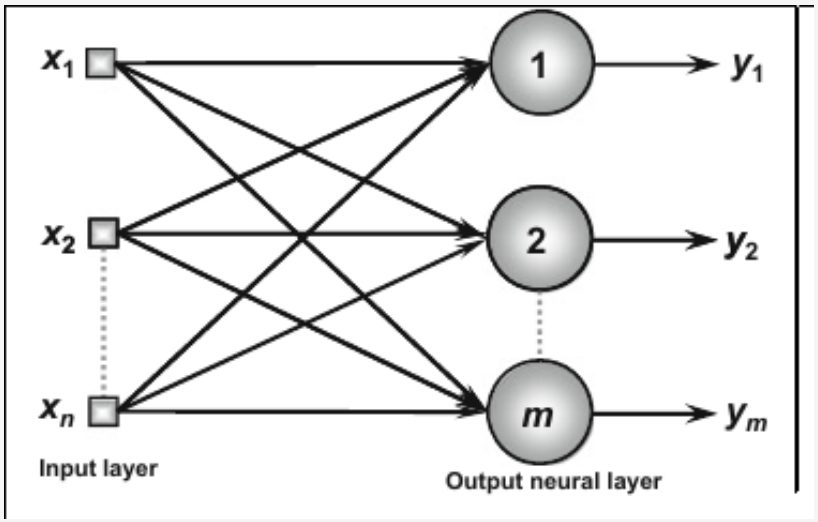
Como se puede apreciar en la Figura 1, tenemos las siguientes arquitecturas de red [Nunes da Silva et al., 2017]:

- *RNA de una sola capa con conexión hacia adelante:* posee una capa de entrada y una sola capa de neuronas que actúa también como capa de salida (Figura 2). La información siempre fluye en una dirección y el número de neuronas que posee siempre coincide con el número de entradas. Generalmente se emplea en problemas de **clasificación** y **filtrado lineal**. El **perceptrón** y la red **ADALINE (Adaptive Linear Element)** poseen este tipo de arquitectura, y para realizar el entrenamiento emplean la regla de aprendizaje de Hebb y la regla Delta.

Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

Figure: RNA con una sola capa y conexión hacia adelante [Nunes da Silva et al., 2017].



Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

- *RNA de múltiples capas con conexión hacia adelante:* poseen una o más capas ocultas (ver Figura 1) y se emplean para tareas como aproximación de funciones, clasificación de patrones, identificación de sistemas, procesos de control, optimización, robótica, entre otras varias más. Los principales tipos de redes que tienen esta arquitectura son el **Perceptrón Multicapa** y las redes de **Función de Base Radial (*Radial Basis Function*)**. Es importante observar que en este tipo de redes el número de salidas **no** necesariamente coincide con el número de entradas.

Introducción (cont'd)

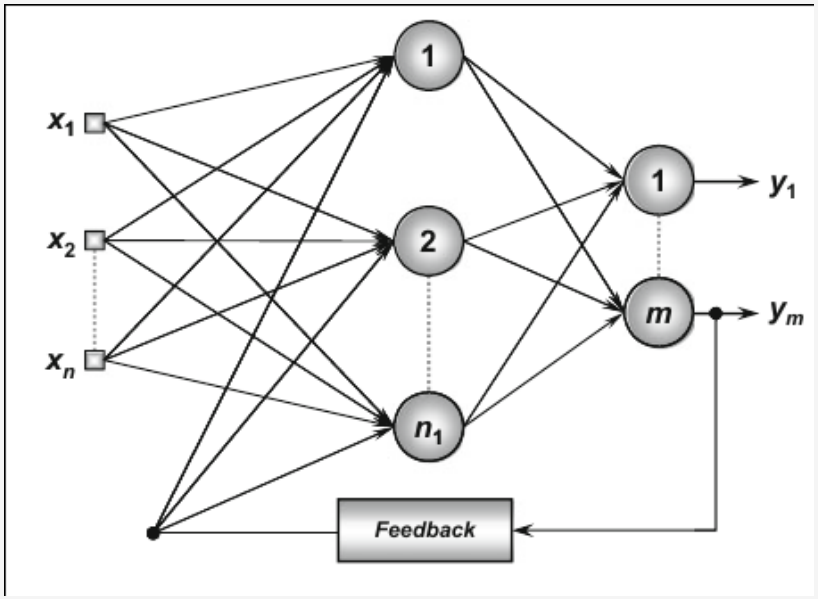
Conceptos básicos y principales tipos de arquitecturas de RNA

- *Arquitectura recurrente o con retroalimentación:* en esta arquitectura la salida de las neuronas se retroalimenta como entrada para otras neuronas (Figura 3). Esta característica permite que este tipo de redes puedan procesar información de forma dinámica, lo que implica que pueden trabajar en sistemas de tiempo variante como la predicción se series, identificación y optimización de sistemas, procesamiento, control, etc. Los principales tipos de RNA que tienen esta arquitectura son la red **Hopfield** y el **Perceptrón con retroalimentación entre neuronas de distintas capas**.

Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

Figure: Red Neuronal Artificial Recurrente [Nunes da Silva et al., 2017].



Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

Práctica 3.

Realice un proceso de investigación introductoria sobre el funcionamiento y características de la red ADALINE. Para ello deberá llevar a cabo las siguientes actividades:

- Describir la arquitectura de la red y su regla de aprendizaje (de manera similar a cómo se hizo para el caso del perceptrón).
- Preparar un ejemplo completo de cálculo.
- Emplear referencias científicas.
- Desarrollar un cuaderno de Jupyter Notebook del proceso llevado a cabo.
- Detalles de la entrega: **Boletín Prácticas 3: ADALINE**

Introducción (cont'd)

Conceptos básicos y principales tipos de arquitecturas de RNA

Práctica 3.

Los siguientes recursos pueden resultar de utilidad para llevar a cabo el trabajo planteado:

- ADALINE for Pattern Classification, Prof. de la Universidad de Nueva York (*NYU Tandon School of Engineering*).
  `http://cis.poly.edu/~mleung/CS6673/s09/ADALINE.pdf`
- Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition. B. Widrow, R. Winter. `https://web.stanford.edu/class/ee373b/NNadaptivefilteringpattrenrecognition.pdf`

Contenidos

Algoritmo backpropagation

Conceptos base

Como paso previo a estudiar el algoritmo de retropropagación de error (*backpropagation*) es importante tomar en cuenta las siguientes consideraciones dentro de una RNA Perceptrón con Múltiples Capas y Neuronas [Nunes da Silva et al., 2017]:

- $W_{ji}^{(L)}$ representa las matrices de pesos que conectan la neurona j-ésima de la capa L con la neurona i-ésima de la capa $(L-1)$.
- I_j^L son los vectores cuyos elementos representan las entradas ponderadas relacionadas con la j-ésima neurona de la capa L, y están definidos por:

$$\begin{aligned} I_j^{(1)} &= \sum_{i=0}^{n} W_{ji}^{(1)} \cdot x_i \\ &= W_{1,0}^{(1)} \cdot x_0 + W_{1,1}^{(1)} \cdot x_1 + \cdots + W_{1,n}^{(1)} \cdot x_n \end{aligned} \tag{1}$$

Algoritmo backpropagation (cont'd)

Conceptos base

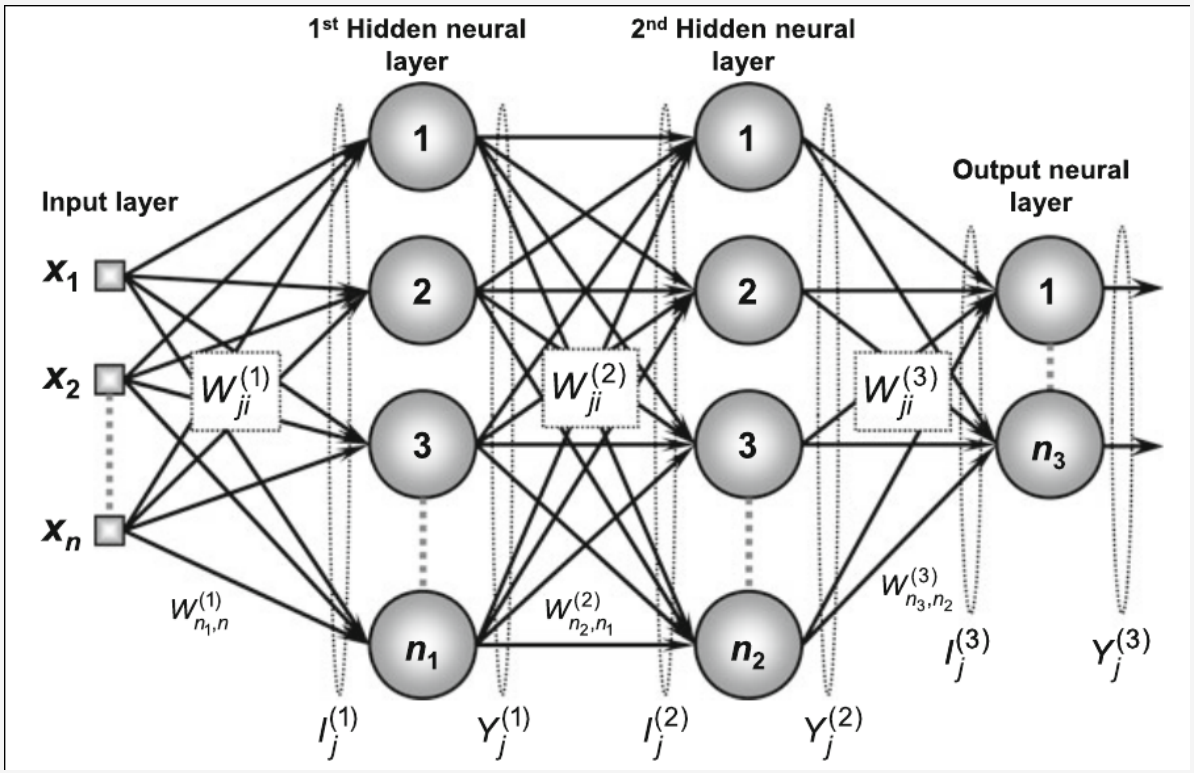
$$
\begin{aligned}
I_j^{(2)} &= \sum_{i=0}^{n_1} W_{ji}^{(2)} \cdot Y_i^{(1)} \\
&= W_{1,0}^{(2)} \cdot Y_0^{(1)} + W_{1,1}^{(2)} \cdot Y_1^{(1)} + \cdots + W_{1,n_1}^{(2)} \cdot Y_{n_1}^{(1)}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
I_j^{(3)} &= \sum_{i=0}^{n_2} W_{ji}^{(3)} \cdot Y_i^{(2)} \\
&= W_{1,0}^{(3)} \cdot Y_0^{2} + W_{1,1}^{(3)} \cdot Y_1^{(2)} + \cdots + W_{1,n_2}^{(3)} \cdot Y_{n_2}^{(2)}
\end{aligned} \tag{3}
$$

Algoritmo backpropagation (cont'd)

Conceptos base

Figure: Elementos de una RNA Perceptrón multicapa [Nunes da Silva et al., 2017]



Algoritmo backpropagation (cont'd)

Conceptos base

- $Y_j^{(L)}$ son vectores cuyos elementos representan la salida de la j-ésima neurona relacionada con la capa L. Se definen como:

$$\begin{aligned} Y_j^{(1)} &= f(I_j^{(1)}) \\ Y_j^{(2)} &= f(I_j^{(2)}) \\ Y_j^{(3)} &= f(I_j^{(3)}) \end{aligned} \tag{4}$$

- Una de las funciones que se emplean con mayor frecuencia para representar el error de la red (salidas deseas frente a las obtenidas), es el **Error Cuadrático Medio (ECM)** calculado sobre la muestra k-ésima de entrenamiento:

$$E(k) = \frac{1}{2}\sum_{j=1}^{n_3}(d_j(k) - Y_j^3(k))^2 \tag{5}$$

Algoritmo backpropagation (cont'd)

Conceptos base

Donde:

- Y_j^3 representa el valor producido por la j-ésima neurona de salida de la red
- $d_j(k)$ es el valor deseado en la salida

Con ello, si consideramos que nuestro corpus de datos contiene p muestras de entrenamiento, el rendimiento general de la red se calcula con el error promedio, definido como:

$$E_M = \frac{1}{p} \sum_{k=1}^{p} E(k) \tag{6}$$

Ajuste de los pesos sinápticos de la capa de salida

Descenso por gradiente

El objetivo del proceso de entrenamiento es ajustar la matriz de pesos W_{ji}^3 a fin de minimizar el error entre las salidas producidas por la red, y las salidas deseadas. Para ello, se emplea la regla de descenso por gradiente [Burges et al., 2005, Nunes da Silva et al., 2017]:

$$\nabla(E^3) = \frac{\partial E}{\partial W_{ji}^{(3)}} = \frac{\partial E}{\partial Y_j^{(3)}} \cdot \frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} \cdot \frac{\partial I_j^{(3)}}{\partial W_{ji}^{(3)}} \tag{7}$$

En base a definiciones previas, se puede establecer que:

$$\frac{\partial I_j^3}{\partial W_{ji}^{(3)}} = Y_i^{(2)} \tag{8}$$

Ajuste de los pesos sinápticos de la capa de salida (cont'd)

Descenso por gradiente

$$\frac{\partial Y_j^3}{\partial I_j^{(3)}} = f'(I_j^{(3)}) \tag{9}$$

Donde $f'()$ representa la primera derivada de la función de activación o transferencia usada.

$$\frac{\partial E}{\partial Y_j^{(3)}} = -\left(d_j - Y_j^{(3)}\right) \tag{10}$$

Con ello, la ecuación nos quedaría:

$$\frac{\partial E}{\partial W_{ji}^{(3)}} = -\left(d_j - Y_j^{(3)}\right) \cdot f'(I_j^{(3)}) \cdot Y_i^{(2)} \tag{11}$$

Ajuste de los pesos sinápticos de la capa de salida (cont'd)

Descenso por gradiente

El ajuste de la matriz de pesos $W_{ji}^{(3)}$ debe hacerse en la dirección opuesta (a fin de minimizar el error):

$$\Delta W_{ji}^{(3)} = -\alpha \cdot \frac{\partial E}{\partial W_{ji}^{(3)}} \iff \Delta W_{ji}^{(3)} = \alpha \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \tag{12}$$

Donde $\delta_j^{(3)}$ es el gradiente local relacionado con la j-ésima neurona de la capa de salida y se define como sigue:

$$\delta_j^{(3)} = \left(d_j - Y_j^{(3)}\right) \cdot f'(I_j^{(3)}) \tag{13}$$

Finalmente, la Ecuación 12 se puede reescribir como sigue:

Ajuste de los pesos sinápticos de la capa de salida (cont'd)

Descenso por gradiente

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \alpha \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \tag{14}$$

Que en notación algorítmica sería:

$$W_{ji}^{(3)} \leftarrow W_{ji}^{(3)} + \alpha \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \tag{15}$$

Aplicación algoritmo Backpropagation

Ejemplo de cálculo

Ejemplo 1.

Dada la RNA Perceptrón multicapa especificada en la Figura 5, realice las siguientes tareas con la misma:

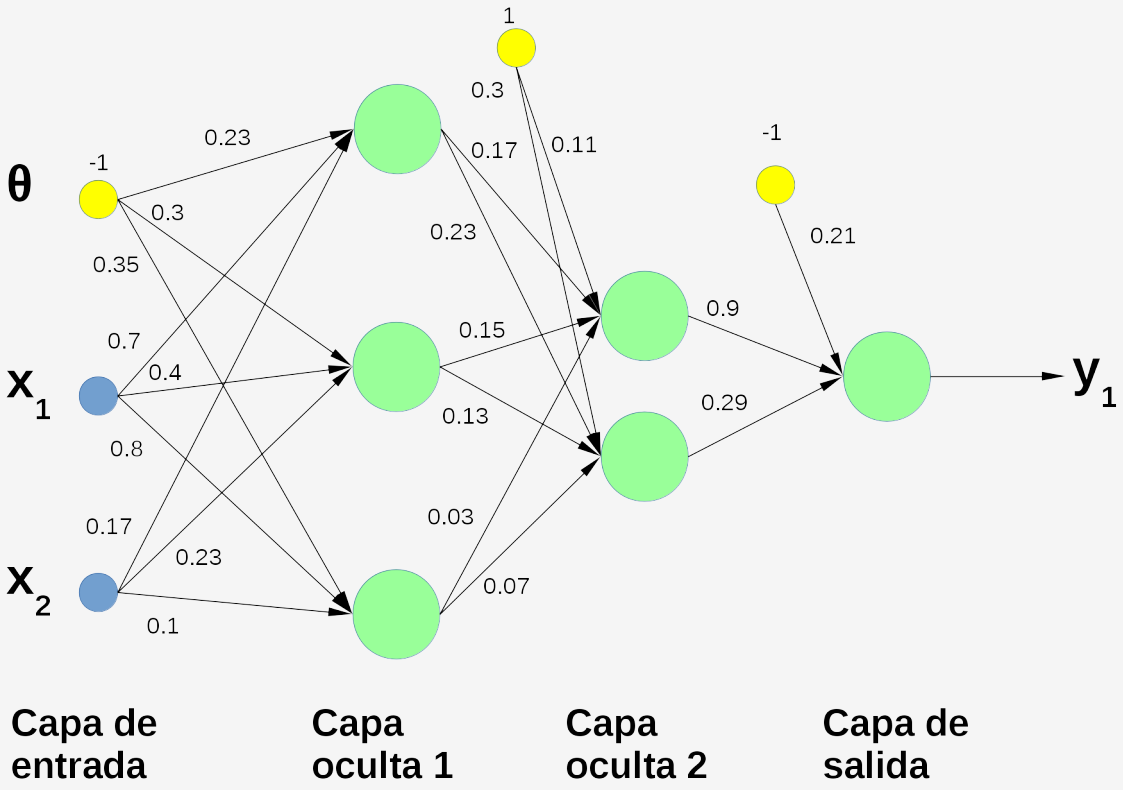
1. Representar a manera de matriz los pesos y diferentes elementos de la red.
2. Calcular la salida empleando las conexiones hacia adelante que define la red.

Asuma que la RNA emplea tangentes hiperbólicas como funciones de transferencia o activación en las neuronas.

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Figure: RNA Perceptrón multicapa de ejemplo



Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (1)

Como primer paso, representamos las matrices de pesos de la RNA. Observe que cada columna representa los pesos de las conexiones de cada neurona, partiendo del bías:

$$W_{ji}^{(1)} = \begin{bmatrix} 0.23 & 0.7 & 0.17 \\ 0.3 & 0.4 & 0.23 \\ 0.35 & 0.8 & 0.1 \end{bmatrix}$$

$$W_{ji}^{(2)} = \begin{bmatrix} 0.11 & 0.17 & 0.15 & 0.03 \\ 0.3 & 0.23 & 0.13 & 0.07 \end{bmatrix}$$

$$W_{ji}^{(3)} = \begin{bmatrix} 0.21 & 0.9 & 0.29 \end{bmatrix}$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (2)

Supongamos que $x_1 = 0.23$ y $x_2 = 0.73$, los valores de I_j^1 vienen dados por:

$$I_j^{(1)} = \begin{bmatrix} I_1^{(1)} \\ I_2^{(1)} \\ I_3^{(1)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(1)} \cdot x_0 + W_{1,1}^{(1)} \cdot x_1 + W_{1,2}^{(1)} \cdot x_2 \\ W_{2,0}^{(1)} \cdot x_0 + W_{2,1}^{(1)} \cdot x_1 + W_{2,2}^{(1)} \cdot x_2 \\ W_{3,0}^{(1)} \cdot x_0 + W_{3,1}^{(1)} \cdot x_1 + W_{3,2}^{(1)} \cdot x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.23 \cdot (-1) + 0.7 \cdot 0.23 + 0.17 \cdot 0.73 \\ 0.3 \cdot (-1) + 0.4 \cdot 0.23 + 0.23 \cdot 0.73 \\ 0.35 \cdot (-1) + 0.8 \cdot 0.23 + 0.1 \cdot 0.73 \end{bmatrix} = \begin{bmatrix} 0.055 \\ -0.040 \\ -0.093 \end{bmatrix}$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (2)

Por otra parte, los valores de Y_j^1 vienen dados por:

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} f(I_1^{(1)}) \\ f(I_2^{(1)}) \\ f(I_3^{(1)}) \end{bmatrix} = \begin{bmatrix} tanh(0.055) \\ tanh(-0.040) \\ tanh(-0.093) \end{bmatrix}$$

$$= \begin{bmatrix} 0.0549 \\ -0.0399 \\ -0.092 \end{bmatrix} \xrightarrow{Y_0^{(1)}=1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.0549 \\ -0.0399 \\ -0.092 \end{bmatrix}$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (3)

El vector I_j^2 de la capa 2 se determina de la siguiente forma:

$$
\begin{aligned}
I_j^{(2)} &= \begin{bmatrix} I_1^{(2)} \\ I_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(2)} \cdot Y_0^{(1)} + W_{1,1}^{(2)} \cdot Y_1^{(1)} + W_{1,2}^2 \cdot Y_3^{(1)} \\ W_{2,0}^{(2)} \cdot Y_0^{(1)} + W_{2,1}^{(2)} \cdot Y_1^{(1)} + W_{3,2}^2 \cdot Y_3^{(1)} \end{bmatrix} \\
&= \begin{bmatrix} 0.11 \cdot 1 + 0.17 \cdot 0.0549 + 0.15 \cdot (-0.0399) + 0.03 \cdot (-0.092) \\ 0.3 \cdot 1 + 0.23 \cdot 0.0549 + 0.13 \cdot (-0.0399) + 0.07 \cdot (-0.092) \end{bmatrix} \\
&= \begin{bmatrix} 0.11 \\ 0.301 \end{bmatrix}
\end{aligned}
$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (4)

Por otra parte, los valores de Y_j^2 vienen dados por:

$$Y_j^1 = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} f(I_1^{(2)}) \\ f(I_2^{(2)}) \end{bmatrix}$$

$$= \begin{bmatrix} tanh(0.11) \\ tanh(0.297) \end{bmatrix} = \begin{bmatrix} 0.1095 \\ 0.288 \end{bmatrix} \xrightarrow{Y_0^{(2)}=1} Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0.1095 \\ 0.288 \end{bmatrix}$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Ejemplo 1: Solución (5)

Finalmente los vectores de la capa 3 I_j^3 e Y_j^3 se determinan de la siguiente forma:

$$
\begin{aligned}
I_j^{(3)} &= \left[I_1^{(3)}\right] = \left[W_{1,0}^{(3)} \cdot Y_0^{(2)} + W_{1,1}^{(3)} \cdot Y_1^{(2)} + W_{1,2}^{(3)} \cdot Y_2^{(3)}\right] \\
&= \left[0.21 \cdot (-1) + 0.9 \cdot 0.1095 + 0.29 \cdot 0.288\right] \\
&= \left[-0.02793\right] \\
Y_j^{(3)} &= \left[Y_1^{(3)}\right] = \left[f(I_1^{(3)})\right] \\
&= \left[tanh(-0.02793)\right] = \left[-0.02792\right]
\end{aligned}
$$

Aplicación algoritmo Backpropagation (cont'd)

Ejemplo de cálculo

Práctica en clase

Determine el valor de salida de la red neuronal presentada en el ejemplo anterior para el caso en el que los pesos tomen un valor aleatorio en el rango $[0, 1]$, los mismos valores de bías (θ) y se trabaje con la función sigmodal.

Referencias

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005).
Learning to rank using gradient descent.
In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Nunes da Silva, I., Hernane Spatti, D., Andrade Flauzino, R., Bartocci Liboni, L. H., and dos Reis Alves, S. F. (2017).
Artificial Neural Networks : A Practical Course.
Springer International Publishing, 1 edition.