Redes Neuronales Artificiales

Programación de Sistemas Adaptativos

Unidad 3: Sistemas inteligentes

Contenidos

- Caso de estudio
- Conceptos básicos
- Perceptrones
- Otros tipos de redes

Caso de estudio

- Una universidad desea utilizar un sistema para decidir cuándo otorga becas.
- Cuenta con datos históricos que incluyen
  - las características de varios candidatos a la beca
  - la decisión tomada (sí se otorgó la beca / no se otorgó).

Caso de estudio

Candidato	¿Escasos recursos?	¿Promedio > 85?	¿Puntaje admisión = sobresaliente?	Beca
C1	No	No	No	No
C2	Sí	Sí	Sí	Sí
C3	No	Sí	Sí	Sí
C4	Sí	Sí	No	No

¿Cómo puede hacer la universidad?

Podría usar una **red neuronal** para que le ayude a tomar esta decisión.

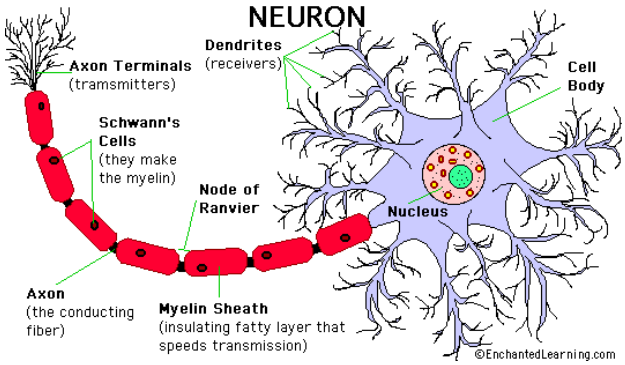
¿Qué son las redes neuronales?

- Técnica de ***aprendizaje automático supervisado***
- que sirve para hacer tareas de **predicción** y **reconocimiento de patrones**
- se basa en el funcionamiento del cerebro (específicamente, las neuronas).

Presentadas por McCulloch y Pitts en 1943.

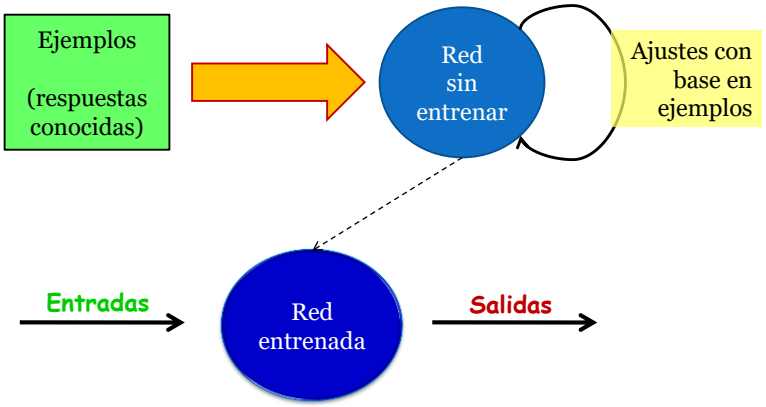
¿Cómo funcionan las redes neuronales?

Nuestro cerebro aprende a través de *modificar* los **pesos** en las *conexiones* de las neuronas.



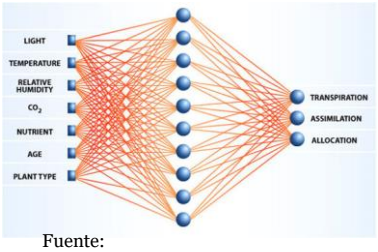
Fuente: http://www.thestudentroom.co.uk/showthread.php?t=1056916

Aprendizaje supervisado



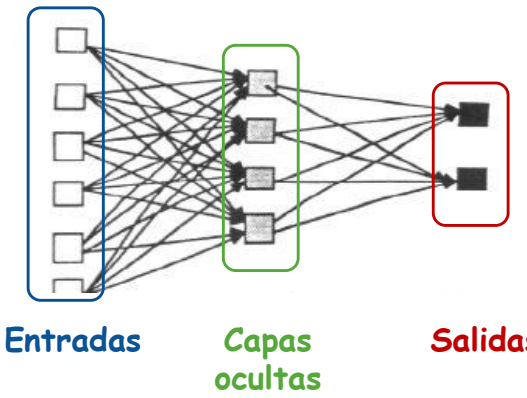
¿En qué consiste una red neuronal?

- Sistema que contiene unidades de procesamiento (**neuronas**) ligadas por **conexiones** con **pesos**.



Fuente: http://lowercolumbia.edu/students/academics/facultyPages/rhode-cary/intro-neural-net.htm

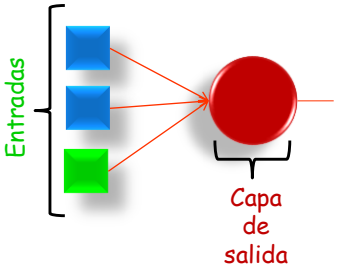
Visualizando redes neuronales



- Se tienen **entradas externas** y una serie de capas.
- Las **salidas** se consideran la última capa.
- Las capas que no son de salida se denominan *internas*.
- Las entradas *no* se toman como capa.

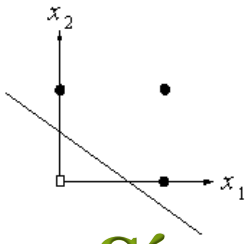
Perceptrón

- Tipo más simple de red neuronal
- Consiste de una sola capa.
- Para clasificación **binaria** de datos ***linealmente separables***.

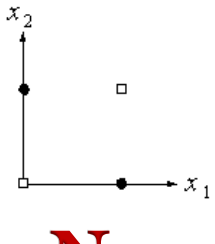


Datos linealmente separables

- Pueden dividirse por un *hiperplano*.



Sí

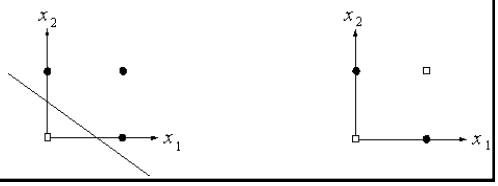


No

Fuente:
http://neuron.eng.wayne.edu/tarek/MITbook/chap1/1_1.html

Entrenamiento de un perceptrón

- El perceptrón va tratar de "dibujar" una *línea* que separe los datos en diferentes clases.
- Para aproximar esta línea, va a ser *entrenado* con un conjunto de datos. A este conjunto de datos le llamaremos **conjunto de entrenamiento**.



Entrenamiento de un perceptrón

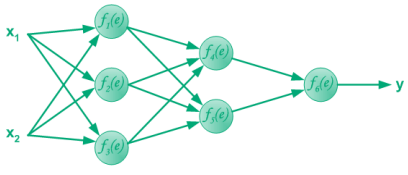
- Para el conjunto de entrenamiento, SIEMPRE debemos tener las **salidas esperadas** (respuestas correctas).
- El conjunto de entrenamiento son nuestras ***entradas*** al perceptrón.
- Cada entrada es un **vector de características** y lo denotaremos por X. A la entrada también le podemos llamar *ejemplo*.

Entrenamiento de un perceptrón

- X=<x_1...x_n>, donde x_i es una característica.
- Introduciremos una x_0 a la cual llamaremos *sesgo*.
- Cada entrada será introducida a la **función de núcleo**, donde cada característica x_i será relacionada con un peso w_i .
- La salida de la función de núcleo será introducida a la **función de activación**.

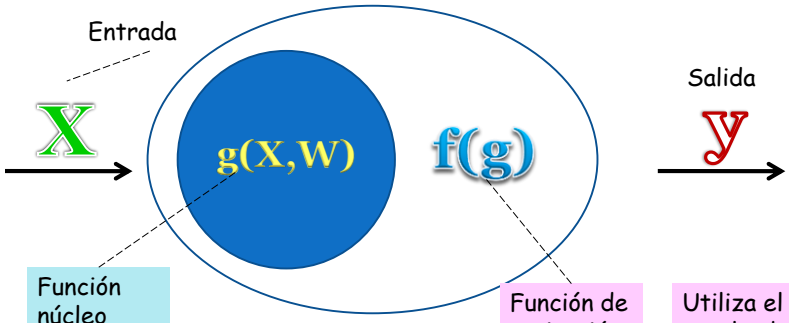
Entrenamiento del perceptrón

- Si la salida obtenida por la función de activación no coincide con la salida deseada, se deben ***corregir*** los pesos.



A cada característica x_i le corresponde un peso w_i

La característica x_0 será el sesgo. Su valor es constante.



Función núcleo

Relaciona entradas y pesos.

Función de activación

Utiliza el resultado de la función núcleo para calcular la salida.

Algoritmo

1. Fijar **pesos iniciales** con valores aleatorios.
2. Por cada entrada:
   a. Calcular la **salida** $y=f(g(X,W))$ de la neurona.
   b. Si y ≠ resultado deseado (d)
      i. **Actualizar** los pesos con $w_{i\_nuevo} = w_i + \eta x_i e$ donde $e = d - y$
3. Ejecutar 2 hasta un valor ε de tolerancia o hasta que los pesos varíen poco.

Funciones que utilizaremos

- Núcleo
  - Suma ponderada
    - $X=(x_0 \ldots x_n)$, $W=(w_0 \ldots w_n)$
    - $g((x_0 \ldots x_n), (w_0 \ldots w_n)) = w_0x_0 + w_1x_1 + \ldots w_nx_n$
    - Ejemplo: $g(x_0, x_1, x_2) = w_0x_0 + w_1x_1 + w_2x_2$

El sesgo siempre tendrá valor $x_0 = -1$

- Activación
  - Escalonada

$$f(g(X,W)) = \begin{cases} 1 & g(X,W) \geq 0 \text{ ------ umbral} \\ 0 & \text{de otra manera} \end{cases}$$

Actualización de los pesos

Característica correspondiente al peso (depende de la entrada)

$$w_{i\_nuevo} = w_i + \eta x_i e$$

Tasa de aprendizaje $\eta \in [0,1]$ (0.1 sugerido)

$$e = d - y$$

Salida deseada (depende de la entrada)

¡Practiquemos!

Queremos entrenar un perceptrón para predecir si un personaje de videojuegos tiene buen equipamiento.

	Descripción del personaje		
	¿Sin armamento?	*¿Muere rápido?*	*¿Equipado?*
Link	No	No	Sí
Mario	No	Sí	Sí
Pac Man	Sí	Sí	No

¡Practiquemos!

Entradas (vectores de características)

	X			
Sesgo →	x_0	x_1	x_2	d
Link	-1	0	0	1
Mario	-1	0	1	1
Pac Man	-1	1	1	0

Entradas (ejemplos)

El peso del sesgo (w0) es inicialmente **0.5**. w1=**-0.5**, w2=**1**. η=0.1

1.- Asignar pesos iniciales aleatorios

w0=0.5, w1=-0.5 y w2=1

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X,W)) de la neurona

Entrada Link

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 0, 0),(0.5, -0.5, 1)) = (0.5)(-1) + (-0.5)(0) + (1)(0)
g((-1, 0, 0),(0.5, -0.5, 1)) = **-0.5**

y = f (-0.5) = **0**

b.- Si y != d:

actualizar pesos

	x_0	x_1	x_2	d
Link	-1	0	0	1

¿0=1? No. El perceptrón se equivocó.

ACTUALIZACIÓN DE PESOS

w0= w0 + ηx0(d-y)
= 0.5 + (0.1)(-1)(1-0)
= **0.4**

w1= w1 + ηx1(d-y)
= -0.5 + (0.1)(0)(1-0)
= **-0.5** (No cambia)

w2 = w2 + ηx2(d-y)
= 1 + (0.1)(0)(1-0)
= **1** (No cambia)

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X)) de la neurona

Entrada Mario

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 1, 1), (0.4, -0.5, 1)) = (**0.4**)(-1) + (-0.5)(0) + (1)(1)
g((-1, 1, 1), (0.4, -0.5, 1)) = **0.6**

y = f (0.6) = **1**

b.- Si y != d:

actualizar pesos

	x_0	x_1	x_2	d
Mario	-1	0	1	1

¿1=1? **Sí.** El perceptrón obtuvo la respuesta correcta. No es necesario actualizar los pesos.

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X)) de la neurona

Entrada Pac Man

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 1, 1), (0.4, -0.5, 1)) = (0.4)(-1) + (-0.5)(1) + (1)(1)
g((-1, 1, 1), (0.4, -0.5, 1)) = **0.1**

y = f (0.1) = **1**

b.- Si y != d:

actualizar pesos

	x_0	x_1	x_2	d
Pac Man	-1	1	1	0

¿1=0? No. El perceptrón se equivocó.

ACTUALIZACIÓN DE PESOS

w0= w0 + ηx0(d-y)
= 0.4 + (0.1)(-1)(0-1)
= **0.5**

w1= w1 + ηx1(d-y)
= -0.5 + (0.1)(1)(0-1)
= **-0.6**

w2 = w2 + ηx2(d-y)
= 1 + (0.1)(1)(0-1)
= **0.9**

3.- Ejecutar paso 2 hasta un valor ε de tolerancia o hasta que los pesos varíen poco.

Por tanto, volvemos al paso 2

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X)) de la neurona

Entrada Link

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 0, 0), (0.5, -0.6, 0.9)) = (0.5)(-1) + (-0.6)(0) + (0.9)(0)
g((-1, 0, 0), (0.5, -0.6, 0.9)) = **-0.5**

y = f (-0.5) = **0**

b.- Si y != d:
actualizar pesos

	x_0	x_1	x_2	d
Link	-1	0	0	1

¿0=1? No. El perceptrón se equivocó.

ACTUALIZACIÓN DE PESOS

w0= w0 + ηx0(d-y)
= 0.5 + (0.1)(-1)(1-0)
= **0.4**

w1= w1 + ηx1(d-y)
= -0.6 + (0.1)(0)(1-0)
= **-0.6** (No cambia)

w2 = w2 + ηx2(d-y)
= 0.9 + (0.1)(0)(1-0)
= **0.9** (No cambia)

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X)) de la neurona

Entrada Mario

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 0, 1),(0.4, -0.6, 0.9)) = (**0.4**)(-1) + (-0.6)(0) + (0.9)(1)
g((-1, 0, 1),(0.4, -0.6, 0.9)) = **0.5**

y = f (0.5) = **1**

b.- Si y != d:

actualizar pesos

	x_0	x_1	x_2	d
Mario	-1	0	1	1

¿1=1? **Sí.** El perceptrón obtuvo la respuesta correcta. No es necesario actualizar los pesos.

2.- Por cada entrada:

a.- Calcular la salida y=f(g(X)) de la neurona

Entrada Pac Man

y=f (g(X,W))
=f (g((x0, x1, x2),(w0, w1, w2))

g((x0, x1, x2),(w0, w1, w2)) = w0x0 + w1x1 + w2x2
g((-1, 1, 1),(0.4,-0.6,0.9)) = (0.4)(-1) + (-0.6)(1) + (0.9)(1)
g((-1, 1, 1),(0.4,-0.6,0.9)) = **-0.1**

y = f (-0.1) = **0**

b.- Si y != d:

actualizar pesos

	x_0	x_1	x_2	d
Pac Man	-1	1	1	0

¿0=0? Sí. El perceptrón obtuvo la respuesta correcta. No es necesario actualizar los pesos.

Ahora éste…

	x			
	x_0	x_1	x_2	d
Entradas 1	-1	0	0	0
Entradas 2	-1	0	1	1
Entradas 3	-1	1	0	1
Entradas 4	-1	1	1	1

El peso del sesgo es inicialmente **1**.Todos los demás pesos son **0**. η=0.1

Otros temas interesantes

- Máquinas de soporte vectorial (*support vector machines*)
- Aprendizaje profundo (*deep learning*)
- Máquinas de aprendizaje extremo (*extreme learning machines*)

Resumen

- Las redes neuronales representan una **técnica de aprendizaje supervisado**.
  - Entrenamiento--Prueba
- Consisten en **capas** de **neuronas interconectadas**.
- Las conexiones tienen **pesos**.

Resumen

- Los pesos se **ajustan** para que la red **aprenda** a reconocer un patrón.
- El **perceptrón** es la red neuronal más sencilla.
- Cuando la red tiene más de una capa, se puede entrenar con **retro-propagación**.

Resumen

- Existen varios tipos de redes neuronales.
  - Recurrentes, mapas auto-organizados (SOM), redes profundas
- Existen otras técnicas para hacer reconocimiento de patrones.