Introducción a Redes de Neuronas

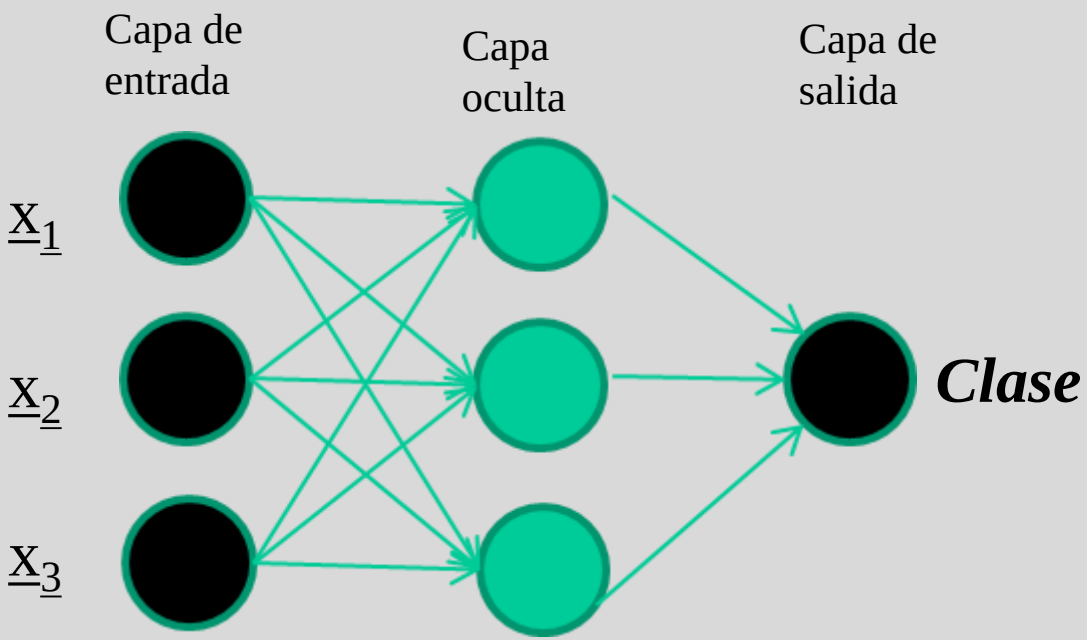
uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE
APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DATOS
GRADO EN ESTADÍSTICA Y EMPRESA
Ricardo Aler

Redes de neuronas (con una capa oculta)

Datos

Atributos			***Clase***
x_1	x_2	x_3	y
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
…			

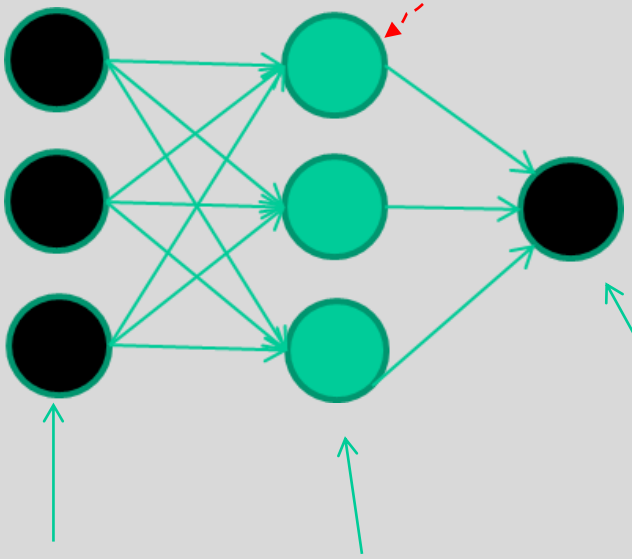


Datos

Atributos			***Clase***
x_1	x_2	x_3	
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
…			

Redes de neuronas

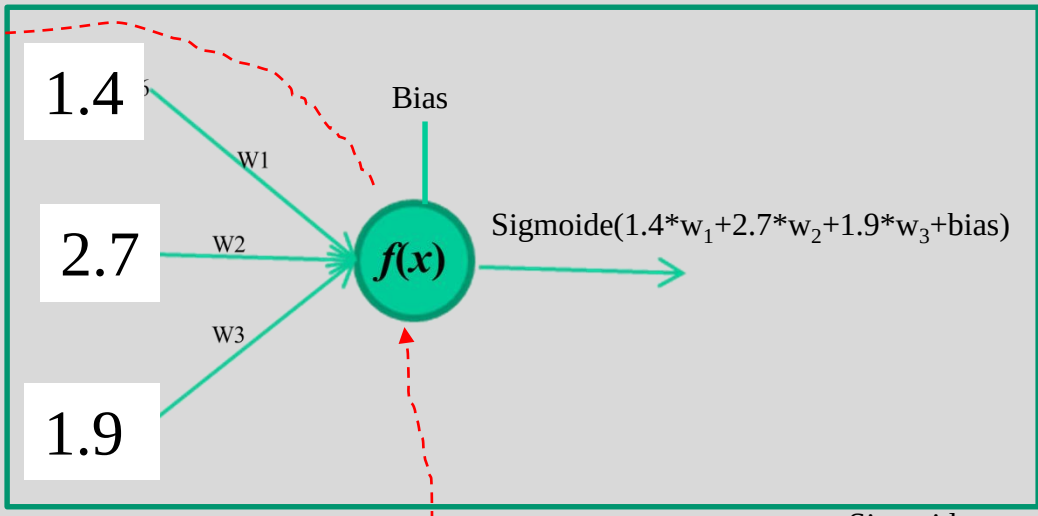
- Con una capa oculta

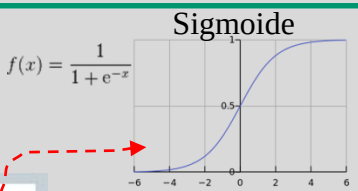


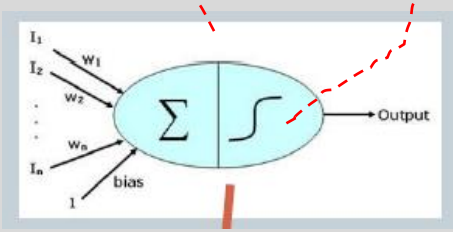
Entradas

Capa oculta

Salida

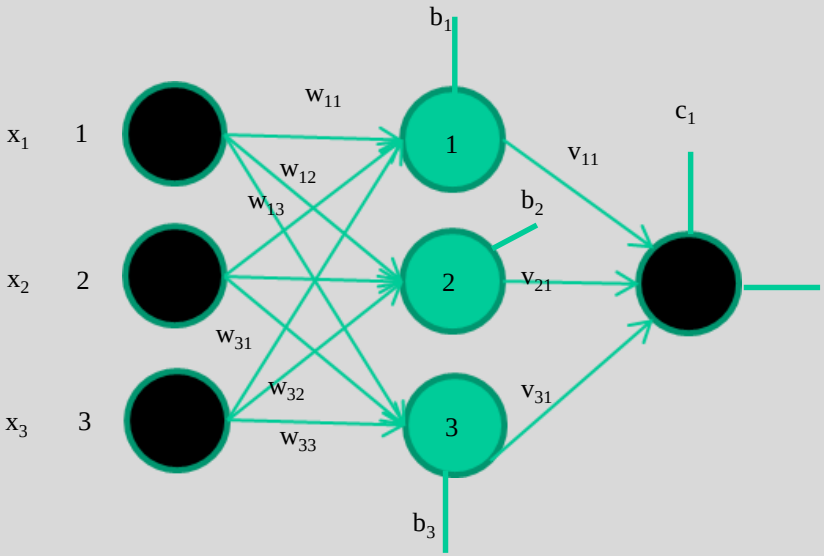




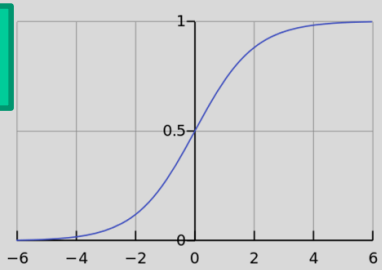


Redes de neuronas

- En último término, se trata de un modelo no lineal, con parámetros ajustables w_{ij}, b_i, v_{ij}, c_i llamados pesos.
- Las RRNN son aproximadores universales:
  - Pueden aproximar cualquier otra función
  - Siempre que haya suficientes neuronas ocultas



S(x)



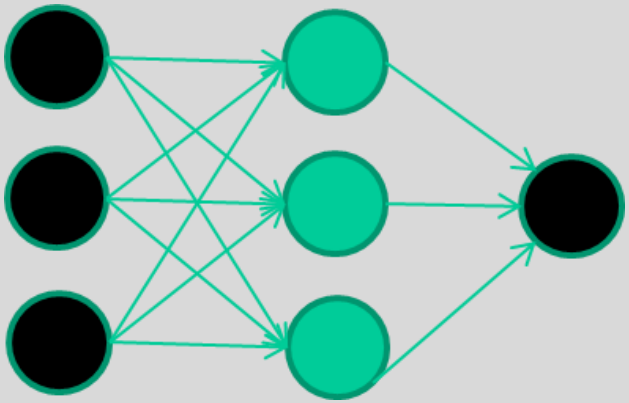
$$y = S(\; S(x_1*w_{11}+x_2*w_{21}+x_3*w_{31}+b_1)*v_{11} + \\ + S(x_1*w_{12}+x_2*w_{22}+x_3*w_{32}+b_2)*v_{21} + \\ + S(x_1*w_{13}+x_2*w_{23}+x_3*w_{33}+b_3)*v_{31} + c_1)$$

Redes de neuronas. Aprendizaje

- El aprendizaje de la red consiste en encontrar los pesos w_{ij}, b_i, v_{ij}, c_i que transformen correctamente la entrada en la salida
- Típicamente se utiliza el algoritmo de retropropagación del gradiente

$$E = \sum_j \frac{1}{2}(t_j - y_j)^2$$

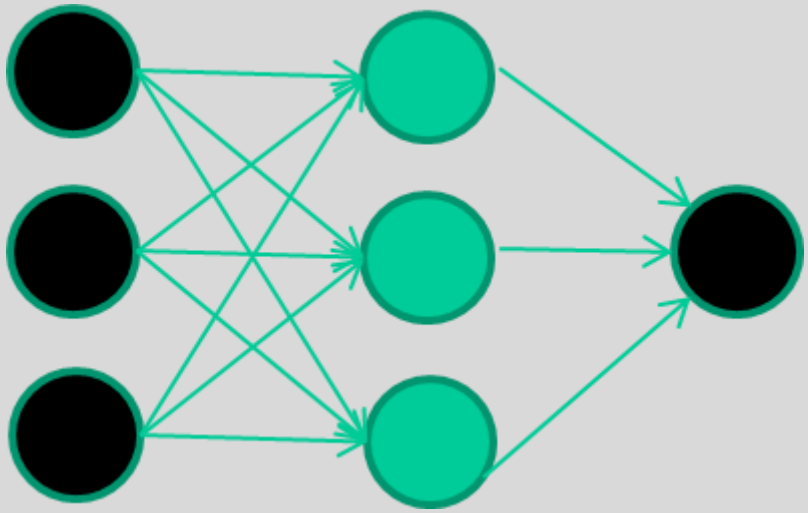
$$\Delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_i$$



Entrenando la red

Atributos			***Clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

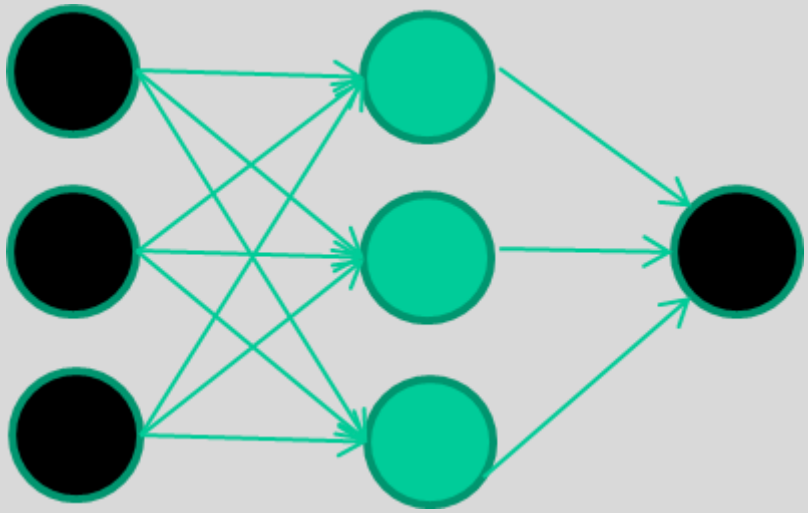


Training data

Atributos			***Clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

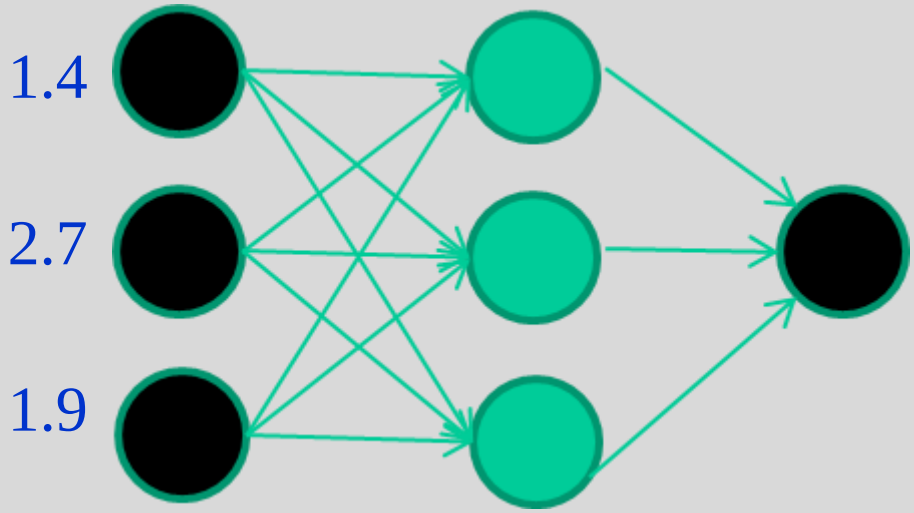
Inicializar con pesos aleatorios



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

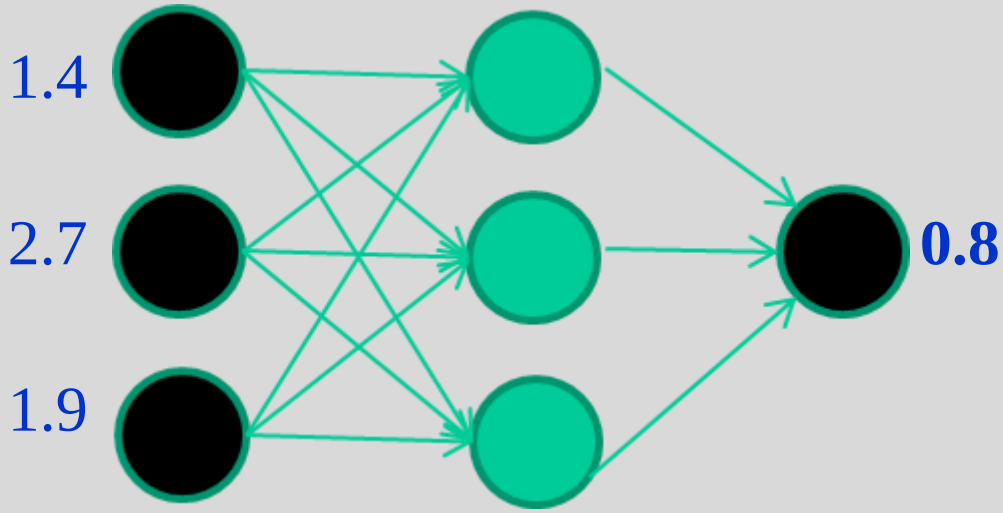
Presentar un ejemplo



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

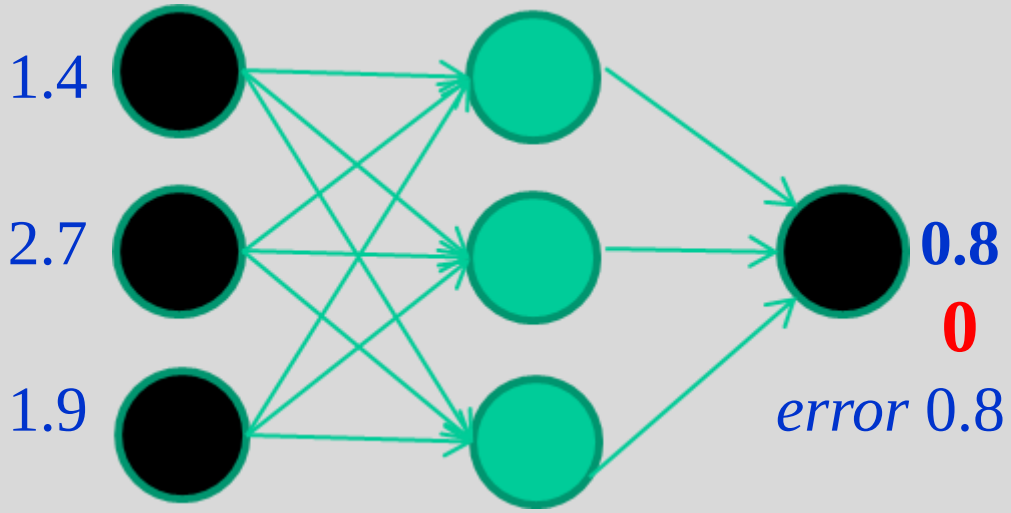
Obtener una salida



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

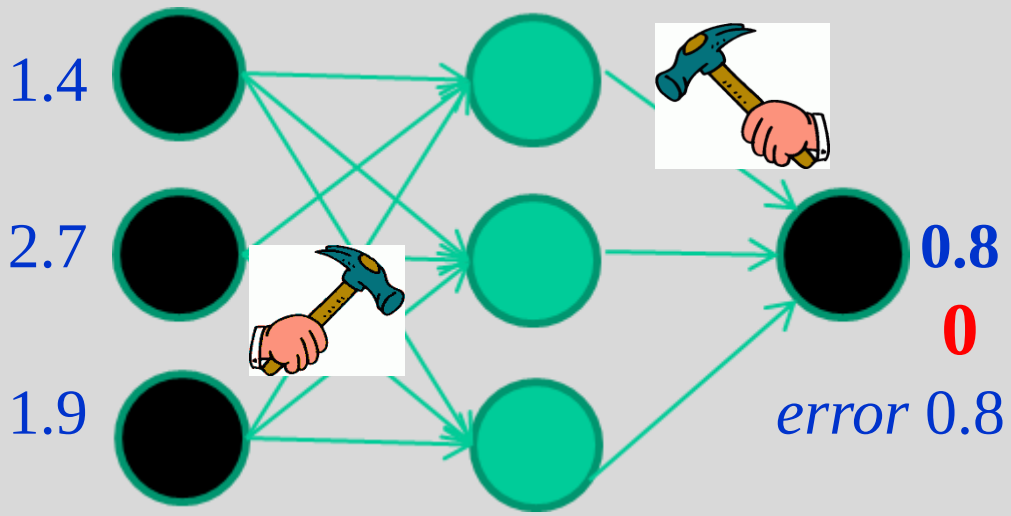
Comparar con la salida real



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

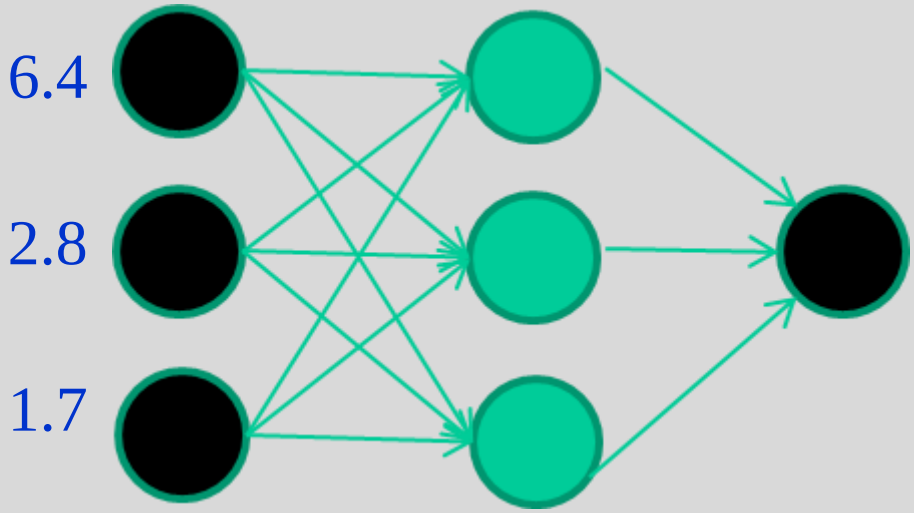
Ajustar los pesos w_i propagando el error hacia atrás



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Presentar otro ejemplo

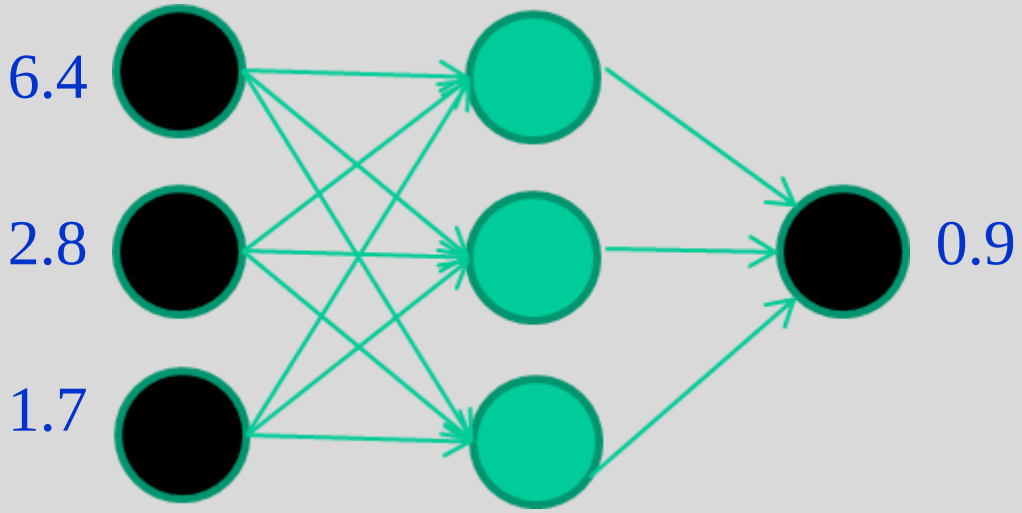


Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

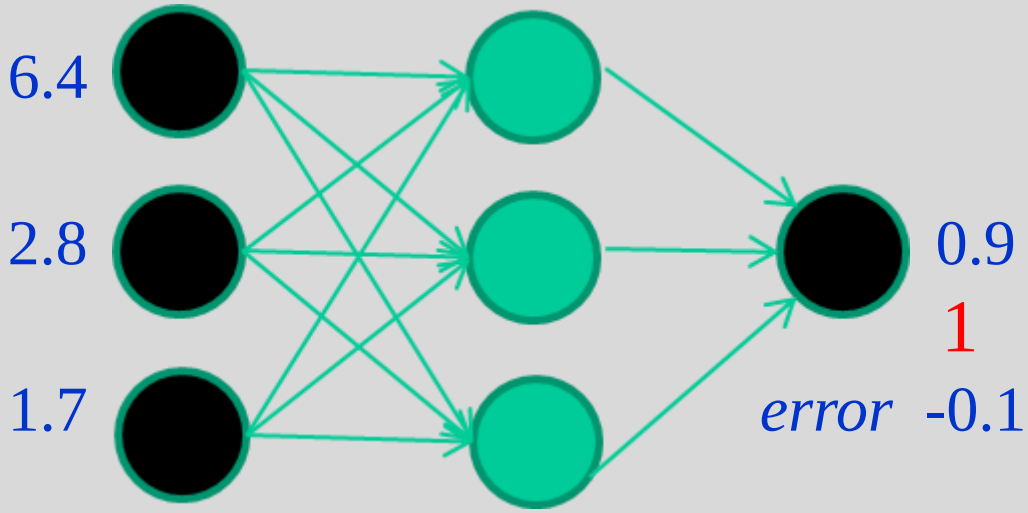
Calcular salida



Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Comparar con salida real

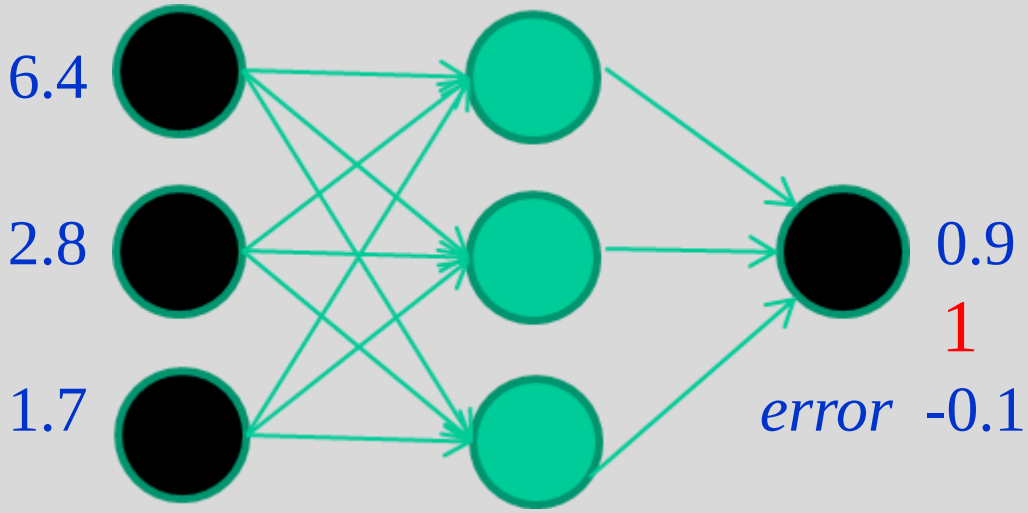


Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Ajustar pesos propagando el error hacia atrás

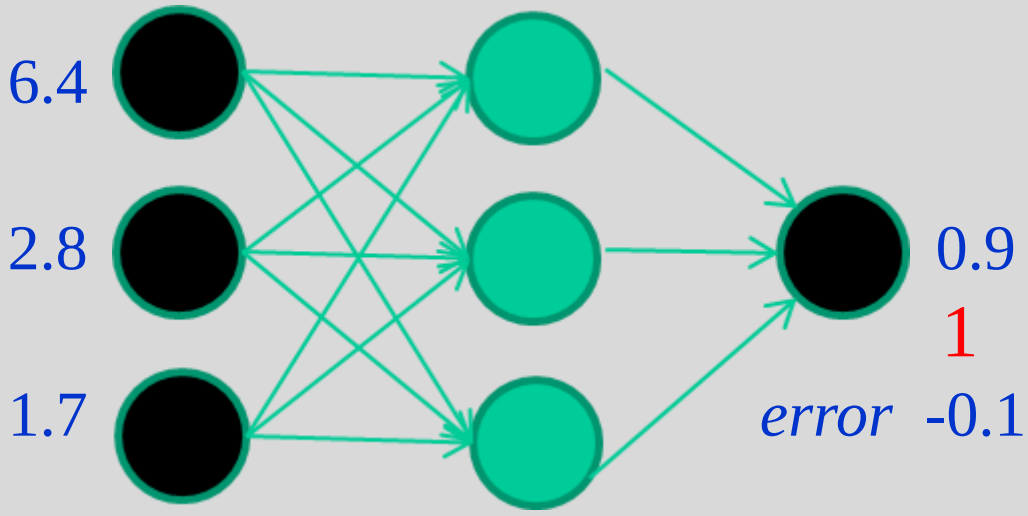


Training data

Atributos			***clase***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Etc ….



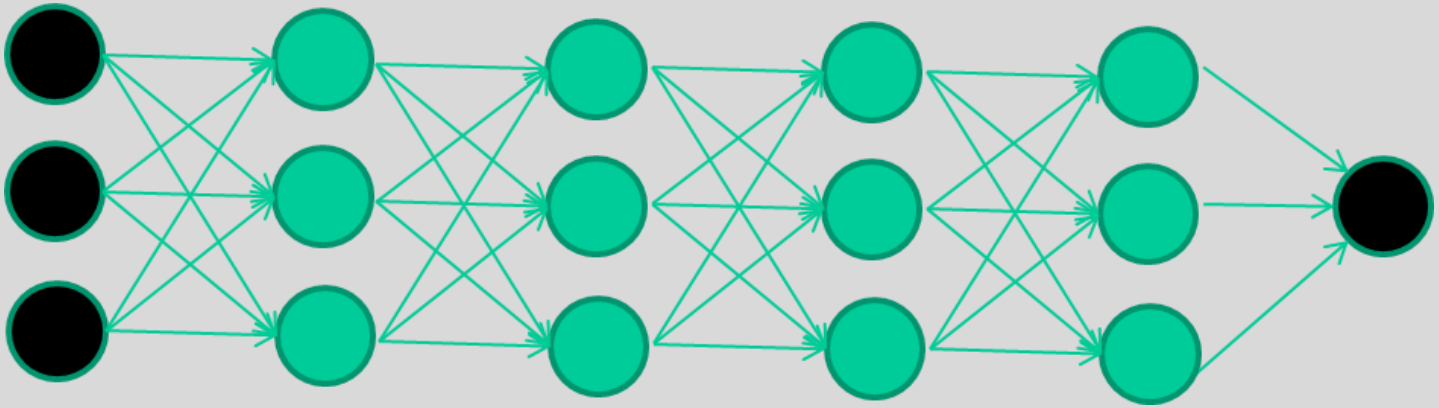
Repetir el proceso miles de veces

Hiper-parámetros de las RRNN

- Hay otros, pero típicamente son:
  - Número de neuronas en la capa oculta
  - Número de iteraciones en el algoritmo de retropropagación del gradiente (es decir, cuantas veces se ajustan los pesos. Si se ajustan muchas veces, la red puede sobreadaptar).

¿Qué es Deep Learning?

- En general, deep learning = deep neural networks. Redes de neuronas con muchas capas ocultas ocultas (típicamente sólo tienen una).



https://playground.tensorflow.org