Java Motor de Red Neuronal

En este documento se explica de forma aproximada el funcionamiento de este motor de redes neuronales. Para más información, consultar el JavaDoc o el código.

Configuración de la red

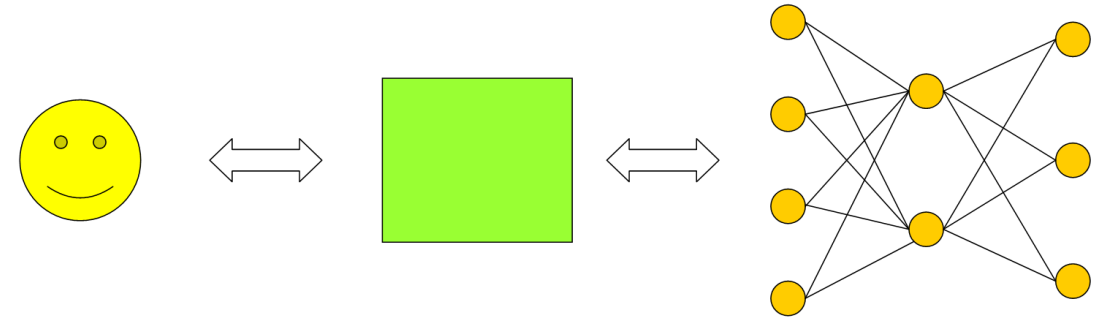
La red neuronal admite los siguientes parámetros de configuración: coeficiente de entrenamiento; coeficiente de momento; los pesos mínimos y máximos para la asignación aleatoria inicial de los pesos iniciales; el número de salidas de la red; el número de entradas; y el número de capas ocultas y las neuronas de cada capa.

Arquitectura de la red

La red se compone de una serie de objetos que implementan la interfaz INeurona interconectados entre sí mediante referencias. Durante la clasificación de una entrada, cada elemento de la red colecciona los datos de todos los objetos conectados a su entrada. Cuando tiene todas las entradas de la capa anterior, calcula la salida y los pasa a todos los objetos conectados a su salida. La red funciona de forma similar pero en sentido inverso en la retropropagación. De esta forma, el funcionamiento de la red consiste en una sucesión de llamadas de los objetos INeurona que forman la primera capa hacia los que forman la capa siguiente, y así sucesivamente.

Estos objetos son creados y administrados por otro de la clase RedNeuronal, que es la que hace de interfaz con el programador. Las funciones que este objeto proporciona a este último son:

- Cálculo: Recibe una entrada del programador, en forma de un vector de flotantes, y pasa cada elemento del vector a cada una de las entradas de la red, lo que supone el inicio de la computación de los datos por parte de la red. Cuando la computación ha acabado, el objeto RedNeuronal compila la salida de las neuronas de la última capa y las devuelve al programador en un vector de float.
- Entrenamiento: Como antes, recibe del programador un vector con la entrada, pero además, recibe otro vector con los datos esperados. La red computa una salida, y a partir de ella y los datos esperados, invoca la función de entrenamiento en los datos de salida. Esta función se retropropaga por las neuronas de la red hacia atrás, cambiando los pesos de las conexiones. Al acabar el proceso, se devuelve la el control al programador, sin devolver nada.
- Otras funciones auxiliares, como cargar y guardar los pesos de la red (para guardar los pesos de una red entrenada, por ejemplo).



Propagación y retropropagación en la red

Objetos de la red neuronal

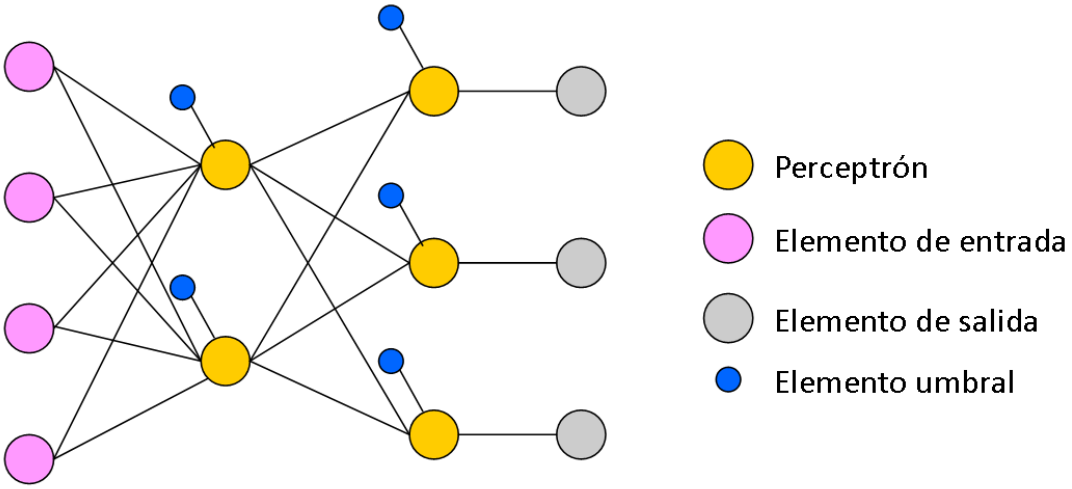
Como se dijo anteriormente, los objetos de la red neuronal están formados por objetos que implementan la interfaz INeurona. Todos estos objetos tienen cuatro funciones, que se enuncian en pseudocódigo a continuación:

- calcular (dato): clasificación
- retropropagar(parámetro de retropropagación)
- conectarEntrante (INeurona)
- conectarSaliente (INeurona)

Las dos primeras funciones son llamadas de forma recursiva de unos elementos de la red a los siguientes o precedentres, respectivamente. Las dos últimas son usadas por el objeto RedNeuronal para crear las conexiones entre los objetos INeurona.

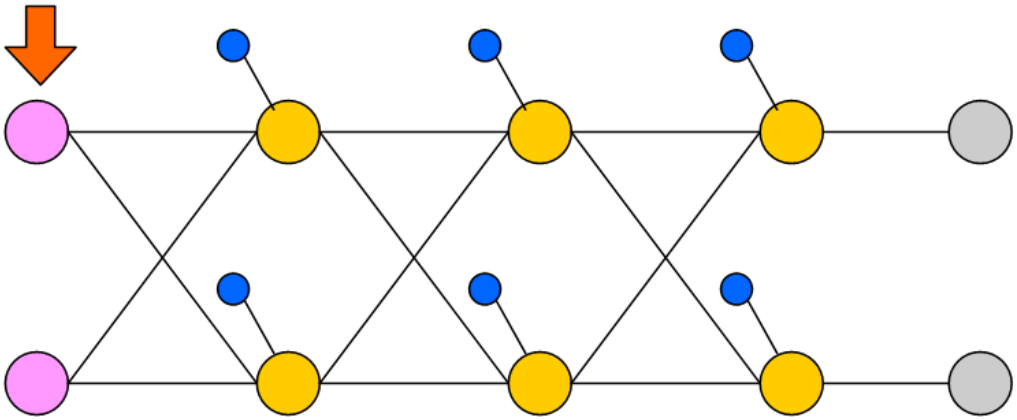
Existen cuatro tipos de objetos INeurona.

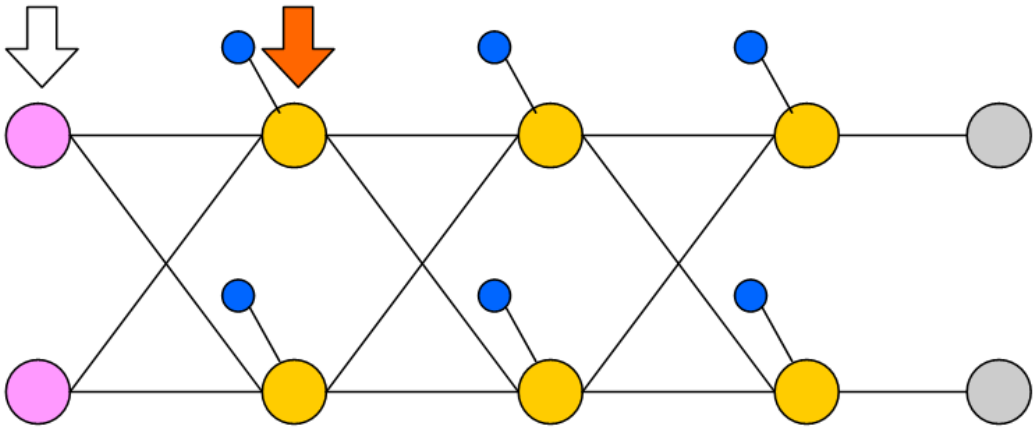
- Perceptrón: Es el único elemento de la red que realiza cálculos. Cada Perceptrón tiene una serie de elementos conectados a su entrada (de tipo Perceptrón o Elemento de entrada), que invocan su función de calcular, y una serie de elementos conectados a su salida (de tipo Perceptrón o Elemento de Salida), a los que pasar los datos calculados . Además, los elementos conectados a su salida pueden invocar su función de retropropagar.
- Elemento de entrada: Los elementos de entrada reciben un dato parcial del objeto RedNeuronal y lo propaga a todas las neuronas conectadas a él.
- Elemento de salida: Los elementos de salida están conectados a las últimas neuronas de la red. Su función es almacenar la salida hasta que el cálculo acabe en toda la red y el objeto RedNeuronal retome el control para recogerla.
- Elemento umbral: El elemento umbral es una neurona que simula la entrada con valor -1. Cada neurona de cálculo de la red (Perceptrón) tiene un elemento umbral conectado a su entrada.



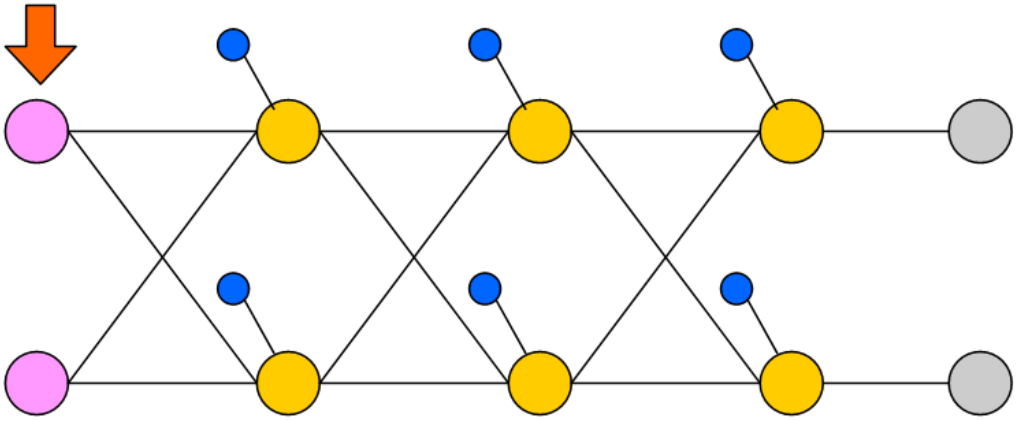
Una traza

Para ilustrar de forma gráfica el funcionamiento de la red, la siguiente tabla muestra una traza del algoritmo de cálculo para una red compuesta por dos entradas, dos capas ocultas de dos neuronas cada una, y dos salidas.

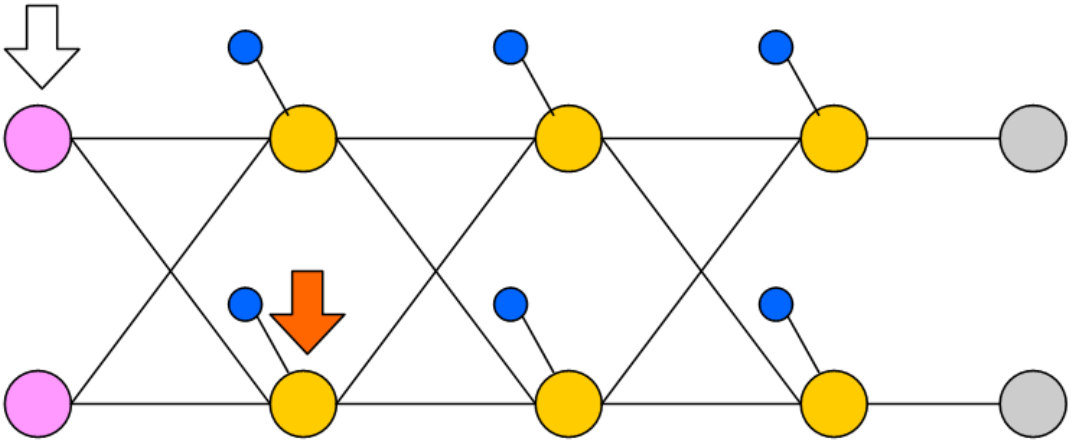
En el primer paso, el objeto RedNeuronal comienza a recorrer los elementos de entrada pasando a cada uno un valor de la entrada: 
Al tener el valor de entrada, el elemento de entrada comienza a recorrer todas las neuronas conectadas al mismo, pasándole el mismo valor que le ha pasado el objeto RedNeuronal.



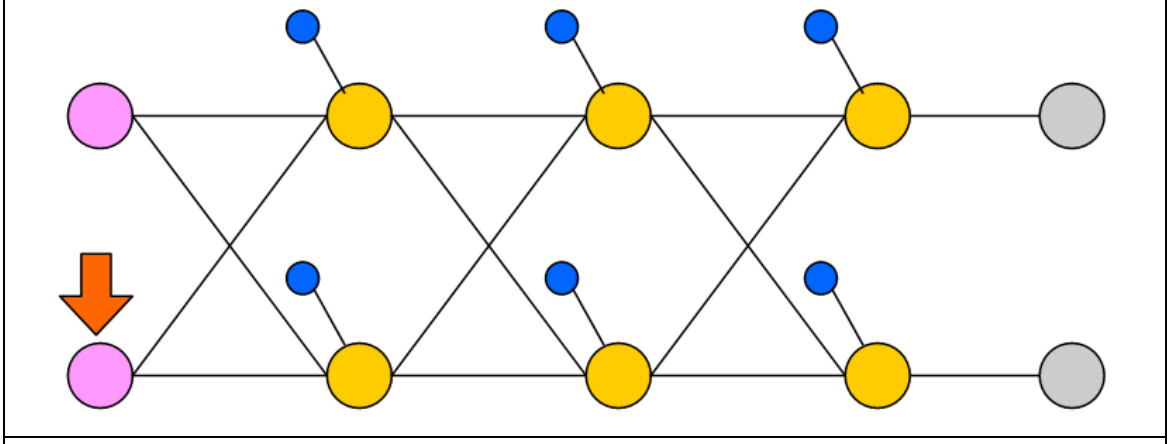
El primer perceptrón de la primera capa recibe este objeto, pero como no tiene todos los valores de sus elementos entrantes, devuelve el control al elemento de entrada:



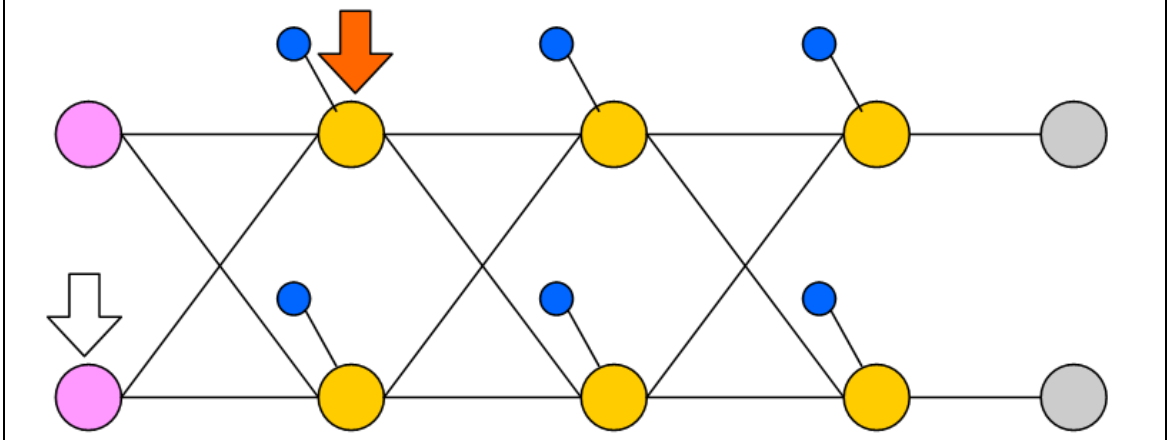
El elemento de entrada continúa con su recorrido de sus elementos salientes. En este caso, para el dato de entrada al segundo perceptrón de la primera capa:



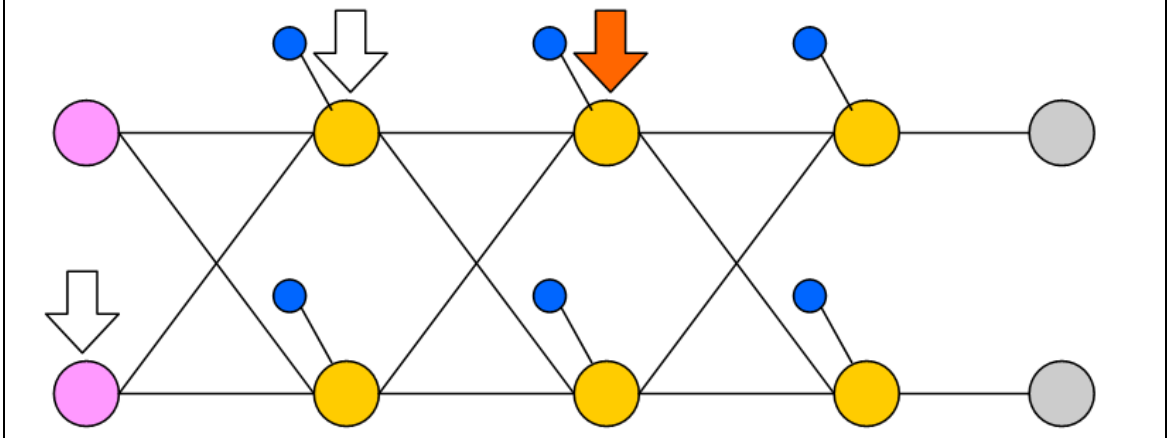
Al igual que antes, este perceptrón no puede realizar los cálculos sin todas sus entradas, así que devuelve el control al elemento entrante. Como este no tiene más conexiones salientes a los que pasar su valor, devuelve el control al objeto RedNeuronal, que le pasa el siguiente valor al siguiente objeto de entrada:



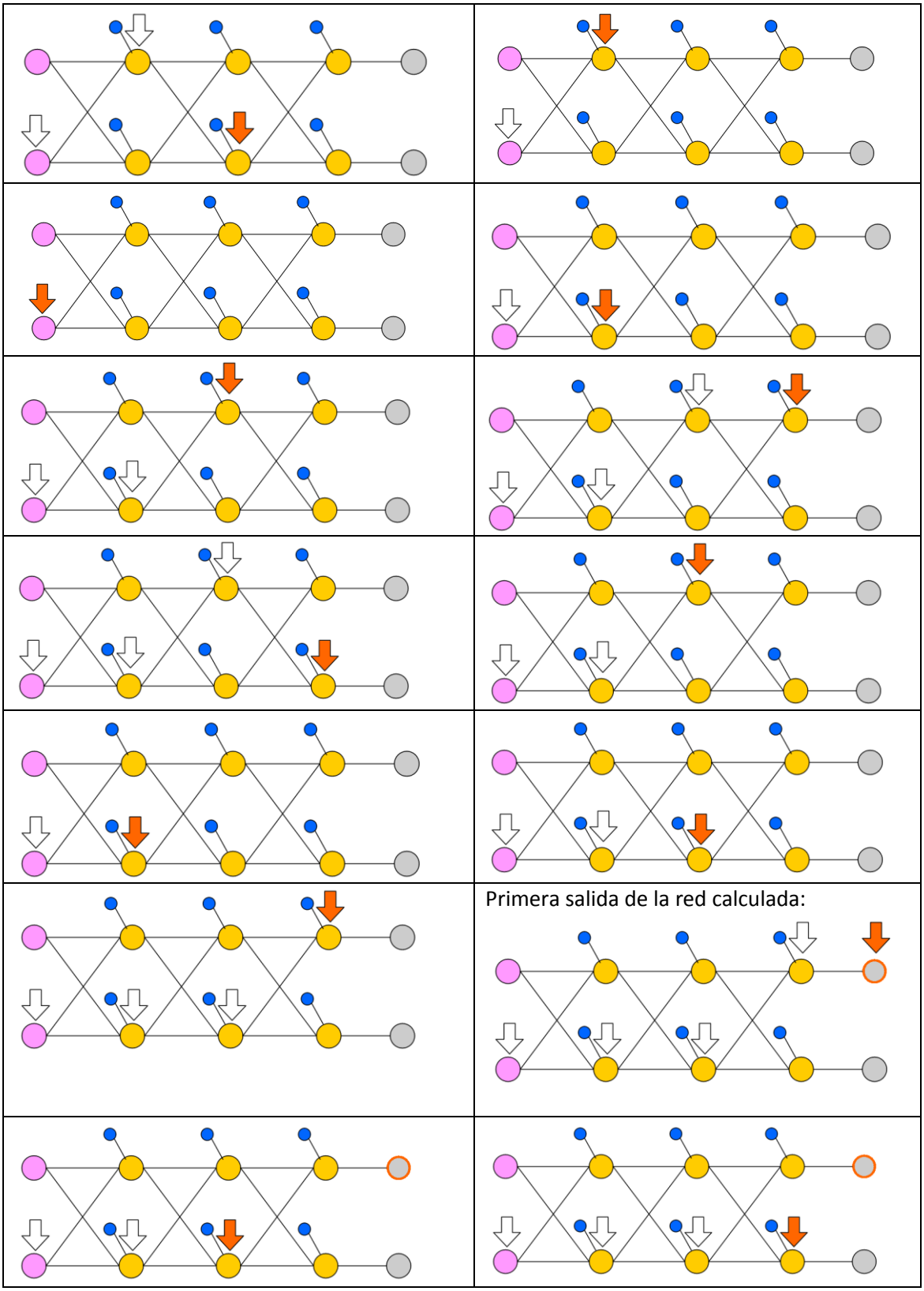
Al igual que antes, se pasa el dato al primer perceptrón:

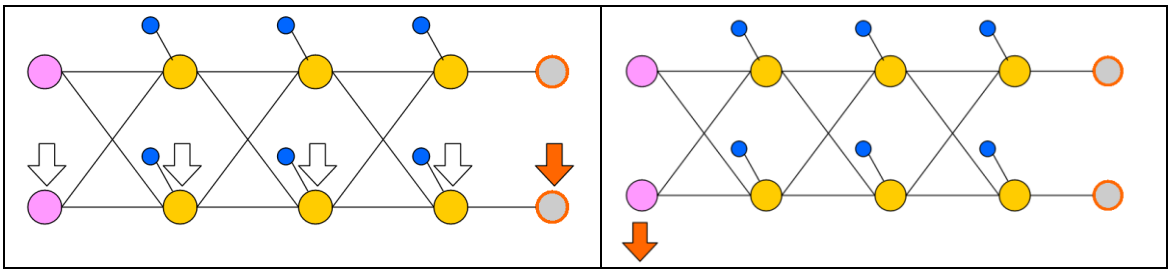


En esta ocasión, este perceptrón tiene ya todos los datos de todas las neuronas entrantes, así que puede calcular su valor de salida, y pasársela a todos los perceptrones conectados a él:



El resto de la traza se ilustra sin texto en una tabla. Léase esta de izquierda a derecha y de arriba abajo.


Primera salida de la red calculada:



Tras el último paso el control vuelve al objeto RedNeuronal, que recoge las salida de los elementos de salida, la compone en un vector, y lo devuelve al usuario.

La traza del algoritmo de retropropagación es muy similar, pero en sentido inverso.

Algoritmos

Aunque el código está comentado, en esta sección vamos a describir el algoritmo de cálculo y retropropagación de los objetos Perceptrón, para facilitar su comprensión.

Cálculo

1. Entrada: Un valor float de una neurona entrante y una referencia a ésta.
2. Se guarda este valor en un diccionario de valores entrantes, cuya clave es una referencia a dicha neurona. Se guarda también en otro diccionario similar, para el algoritmo de retropropagación.
3. Si todas las neuronas entrantes han pasado ya su valor:
    a. Sumar , por cada neurona entrante **n**, el valor que esta pasó por parámetros multiplicado por el peso de la conexión de este objeto con **n**.
    b. Usar la función sigmoide para calcular el valor de la suma anterior. Esto es el valor de salida, que guardamos en un atributo, para el algoritmo de retropropagación.
    c. Recorrer todas las neuronas de salida, invocando su función calcular, y pasando como parámetro el valor de salida y una referencia a este objeto.
    d. Hacer una copia del diccionario de valores entrantes, para conservar los valores de entrada para el algoritmo de retroprogramación.
    e. Vaciar de valores el diccionario de valores entrantes (ponerlos a null), y reiniciar el contador de neuronas entrantes que nos han pasado un valor de entrada.
4. Se acaba la función y se devuelve el control a la neurona que la invocó.

Retropropagación

1. Entrada: Un valor float de una neurona saliente, y una referencia a esta.
2. Se guarda este valor en un diccionario de valores procedentes de neuronas salientes, cuya clave es una referencia a dicha neurona.
3. Si todas las neuronas salientes han pasado ya su valor:
    a. Sumar , por cada neurona conectada a la salida, el valor que esta pasó por parámetros.

b. Hacer deltaMinúscula igual al valor de salida de este objeto, v, calculado durante el algoritmo de cálculo anterior, multiplicado por (1-v), multiplicado por el sumatorio del paso anterior.
c. Recorrer todas las neuronas entrantes, invocando su función retropropagar, y pasándole *el resultado de multiplicar deltaMinúscula* ***por el peso de la conexión*** *entre este objeto y la neurona entrante*, además de una referencia a este objeto. *Nótese cómo la multiplicación por los pesos lo hacemos en este paso, de manera que esta multiplicación no hay que hacerla en el paso 3.a.* Como es una llamada síncrona, el algoritmo espera a que la última neurona entrante acabe su función de retropropagación para seguir.
d. Por cada neurona entrante, **ni**:
    i. Hacer **diferencial** igual a la suma de:
        1. El factor de aprendizaje, multiplicado por deltaMinúscula, multiplicado por la última entrada procedente de ni (esta entrada se guardó en el paso 3.d. del algoritmo de cálculo).
        2. El factor de momento, por el diferencial obtenido en una retropropagación anterior.
    ii. Guardar este diferencial, asociado a la neurona ni, para calcular el momento en una posible retroprogación posterior.
    iii. Sumar diferencial al peso de la conexión entre ni y este objeto.
e. Vaciar de valores el diccionario de valores procedentes de neuronas salientes (ponerlos a null), y reiniciar el contador de neuronas salientes que nos han pasado un valor de retropropagación.

4. Se acaba la función y se devuelve el control a la neurona que la invocó.