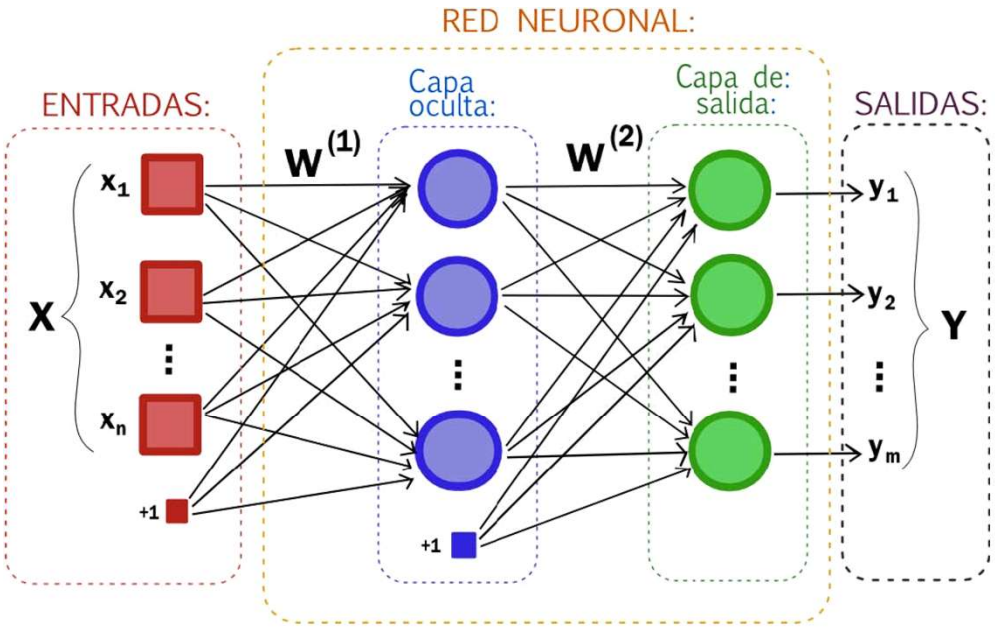
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS
ALMACENES Y MINERÍA DE DATOS

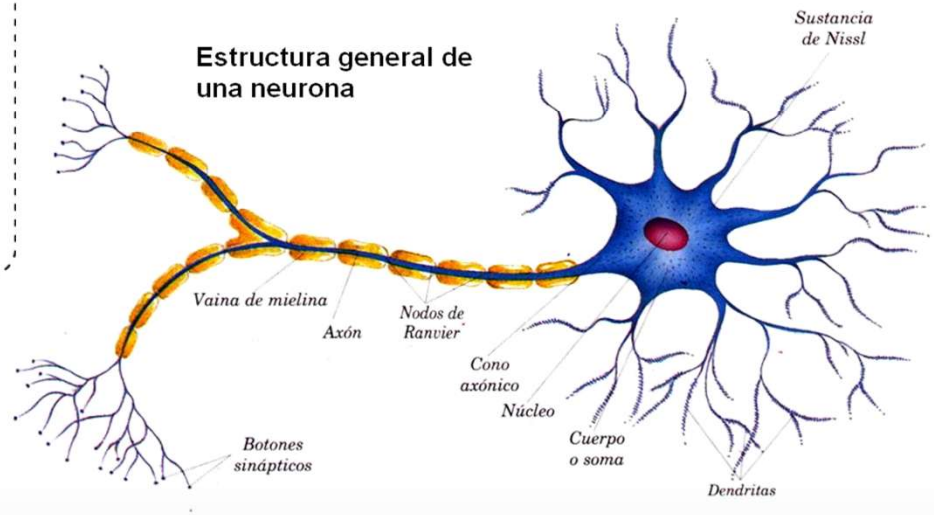
Clasificación: Redes neuronales

Gerardo Avilés Rosas
gar@ciencias.unam.mx

Introducción

- Las **Redes Neuronales** son una **colección de nodos** conectados, con **entradas, salidas** y **procesamiento** en cada uno de ellos.
- El campo de las redes neuronales fue originalmente impulsado por **psicólogos** y **neurobiólogos** quienes buscaban desarrollar y probar análogos de las **neuronas** en computadoras:



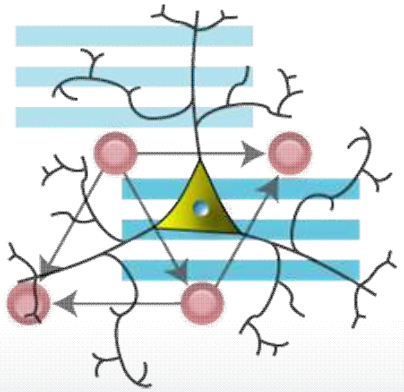


Introducción

- A los nodos de una **red neuronal** se les conoce como unidades y existen de dos tipos: ***de entrada y de salida***. Estas unidades están ***conectadas*** y cada conexión ***tiene un peso asociado***.
- Entre la **capa de entrada** y la **de salida** de la red puede existir un número de **capas ocultas** para ***procesamiento***.
- Las redes neuronales **deben ser entrenadas** con un conjunto de **patrones de aprendizaje** (*aprendizaje supervisado*). Una vez entrenada es utilizada para hacer **predicciones**.
- El aprendizaje de la red neuronal también se conoce como **aprendizaje conexionista** debido a las conexiones que existen entre las unidades.
- Suelen utilizarse para problemas de **clasificación** y para **reconocimiento de patrones**.
- En algunos casos, las redes neuronales proveen una **mejor inducción** que el resto de los algoritmos y **generalizan** de mejor forma.

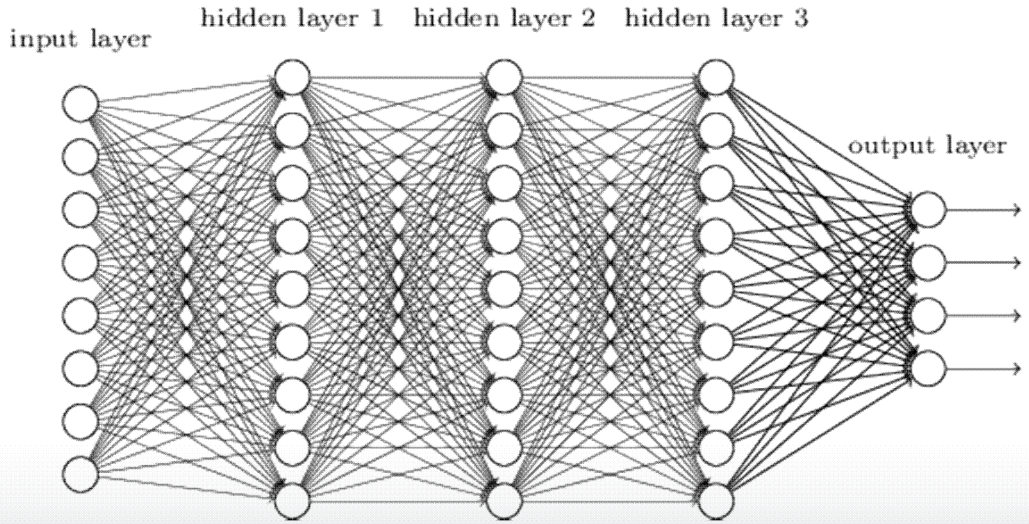
...Introducción

- Durante la **fase de aprendizaje**, la red ***aprende*** mediante el ***ajuste de los pesos*** con el fin de ser capaces de **predecir** la etiqueta de clase correcta de las tuplas de entrada.
- Pueden ser utilizadas en datos en los que se puede tener **poco conocimiento** de las relaciones entre los atributos y clases.
- Están bien adaptados para entradas y salidas con **valores continuos** y son inherentemente paralelas (*técnicas de paralelización se pueden utilizar para acelerar el proceso de cálculo*).
- Sus ventajas incluyen su **alta tolerancia** de los **datos con ruido**, así como su capacidad para **clasificar patrones** sobre los que **no han sido entrenadas.**



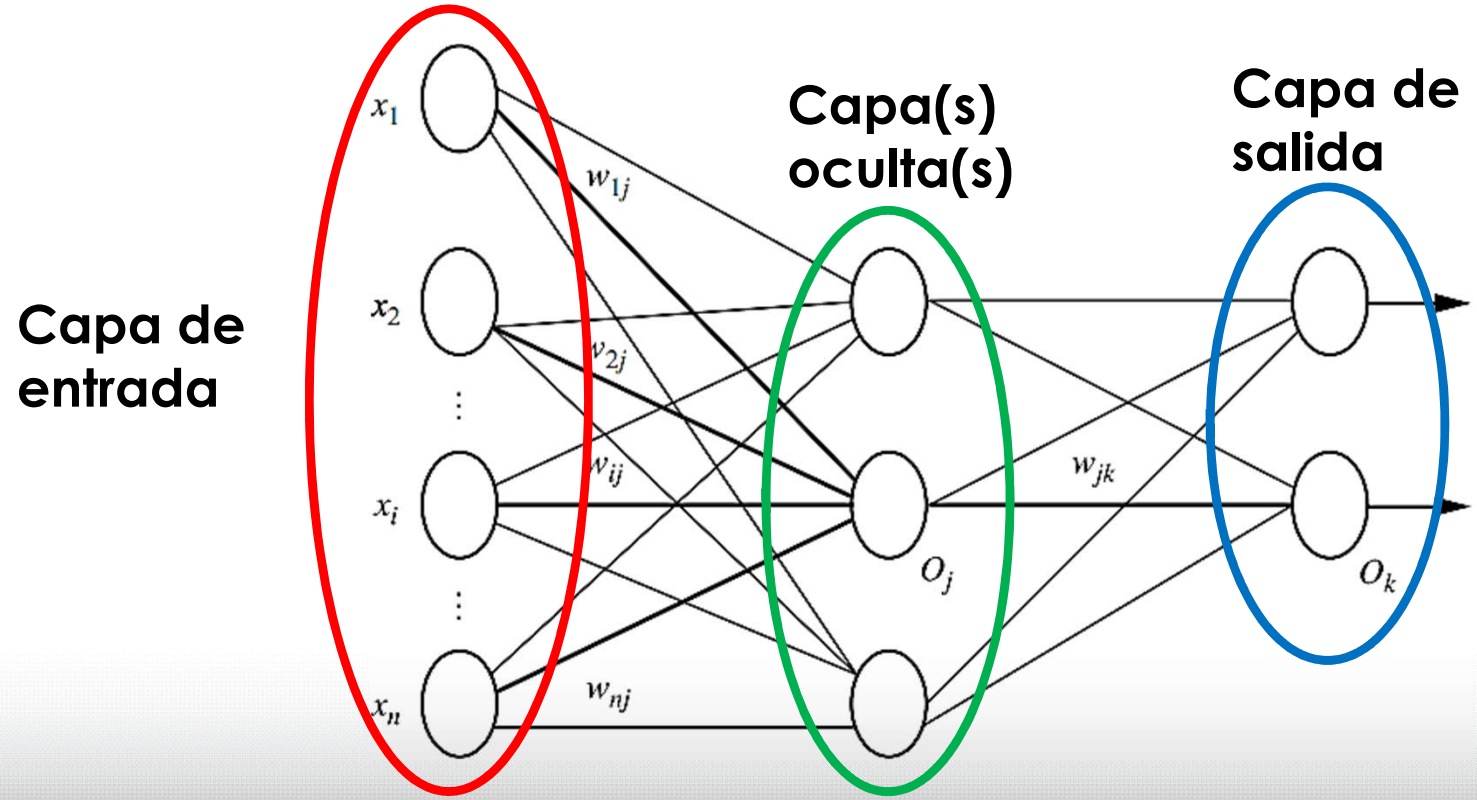
…Introducción

- Sin embargo, **no son muy utilizadas** para Minería de Datos:
  - Las redes neuronales implican **largos tiempos de entrenamiento**, los cuales son **imprácticos** para grandes volúmenes de datos.
  - Requieren una **serie de parámetros** que suelen ser mejores si se determinan empíricamente.
  - Han sido criticadas por su **escasa interpretabilidad**: *es difícil para los seres humanos interpretar el significado simbólico detrás de los pesos aprendidos y de las unidades ocultas en la red.*
  - Estas características que las hicieron inicialmente menos deseables para Minería de Datos.



Redes multicapa de alimentación

- El **algoritmo de retropropagación** realiza el aprendizaje en una ***red neuronal multicapa de alimentación hacia adelante***.
- Maneja un **aprendizaje iterativo**, es decir, aprende de un **conjunto de pesos** para la predicción de la etiqueta de clase de las tuplas .
- Este tipo de redes consisten en una **capa de entrada**, una o más capas ocultas y una capa de salida



…Redes multicapa de alimentación

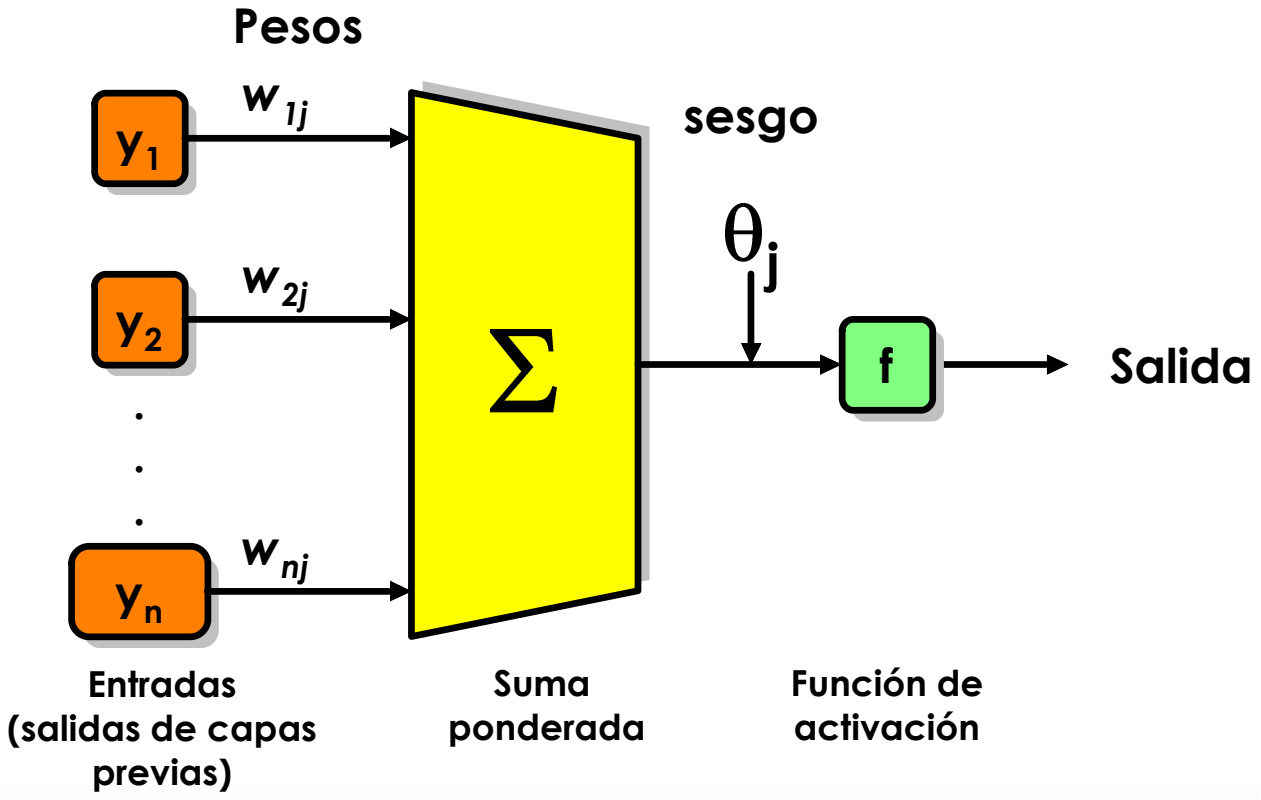
- Cada capa se compone de **unidades**. Las ***entradas*** a la red corresponden a los **atributos medidos** para cada tupla de entrenamiento.
- Estos atributos alimentan simultáneamente las unidades que constituyen la **capa de entrada**; pasan a través de ella para luego **ponderarse** y **alimentar simultáneamente** a una **segunda capa** conocida como **capa oculta.**
- Las **salidas** de las unidades de la **capa oculta** se puede introducir en ***otra capa oculta***, y así sucesivamente:

  El **número de capas ocultas** es **arbitrario**, aunque en la práctica, sólo suele usarse una.

- Las **salidas ponderadas** de la **última capa oculta** se introducen a la **capa de salida**, la cual *fabrica* la **predicción** de la red para las tuplas dadas.
- Las unidades en las capas ocultas y capa de salida se refieren a veces como ***neurodes*** (*base biológica*), o simplemente como **unidades de salida**.

…Redes multicapa de alimentación

- La red se conoce como de ***alimentación hacia adelante*** debido a que **ninguno de los pesos** en cada ciclo **se toma de nuevo** como una unidad de entrada o una unidad de salida de una capa anterior.
- Cada unidad de salida toma como entrada, una **suma ponderada** de las salidas de las unidades de la capa anterior:



…Redes multicapa de alimentación

- Se aplica una **función no lineal** (*de activación*) a la entrada ponderada.
- Este tipo de red es capaz de modelar la predicción de una clase como una combinación lineal de las entradas.
- Desde un punto de vista estadístico, realizan una **regresión no lineal**.

Topología de una red

- Antes de que pueda comenzar el entrenamiento, el usuario debe decidir sobre la **topología de la red**, especificando el **número de unidades** en la capa de entrada, el **número de capas ocultas** (*si hay más de una*), el **número de unidades** en cada **capa oculta**, y el **número de unidades** en la **capa de salida**.
- Una **normalización** de los valores de entrada para cada atributo en las tuplas de entrenamiento puede ayudar a acelerar la fase de aprendizaje:

  Normalmente, los valores de entrada se normalizan para tener valores entre **0.0** y **1.0**.

- Los atributos con valores discretos pueden ser codificados de tal manera que **exista una unidad de entrada** por valor de dominio:
  - Si un atributo **A** tiene tres valores posibles o conocidos **{a_0,a_1,a_2}**, entonces podemos asignar tres unidades de entrada para representar **A**, es decir, **I_0, I_1, I_2. Cada unidad se inicializa con cero**.
  - Si **A = a_0**, entonces **I_0** se establece en **1**. Si **A = a_1**, **I_1** se establece en **1**, y así sucesivamente.

…Topología de una red

- Las **redes neuronales** se pueden utilizar para **clasificación** o para **predicción**.
- En la **clasificación**, una unidad de salida se utiliza para representar **dos clases** (***valor 1*** *y* ***valor 0***). Si hay más de dos clases, se utiliza una unidad de salida por clase.
- No hay reglas en cuanto al **"mejor"** número de unidades de la capa oculta. El **diseño de redes** es un proceso de **ensayo y error**, que puede **afectar la precisión** de la red.
- Los valores iniciales de los pesos también pueden afectar la precisión resultante:
  - Una vez que una red ha sido entrenada y si su precisión no se considera aceptable, se repite el proceso de entrenamiento con una topología diferente o un conjunto diferente de los pesos iniciales.
- Técnicas de **validación cruzada** (*utilizadas para la estimación de la precisión*) se pueden utilizar para ayudar a determinar cuando se ha encontrado una **red aceptable**.

Retropropagación

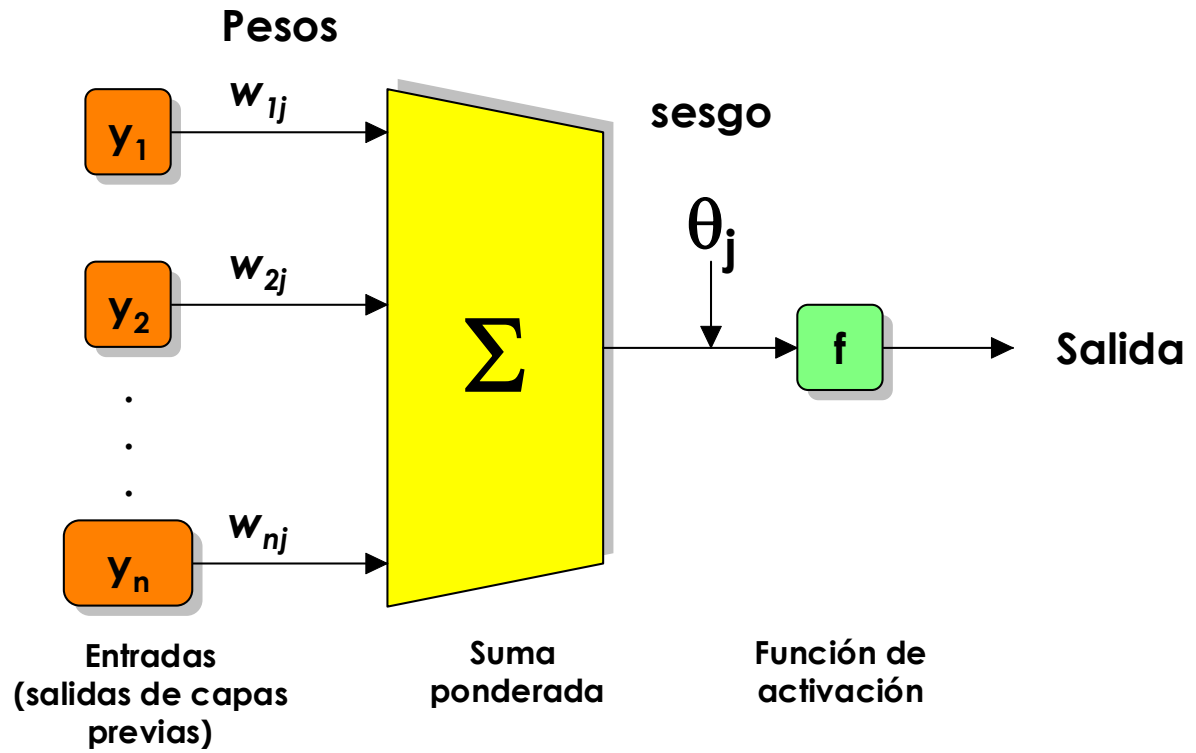
- La **retropropagación** aprende a través de un **proceso iterativo** de los datos de un conjunto de tuplas de entrenamiento, comparando la predicción de la red para cada tupla con el valor actual conocido como valor objetivo.
- El **valor objetivo** puede ser la **etiqueta de clase conocida** de la tupla de entrenamiento (*para problemas de clasificación*) o un **valor continúo** (*para problemas de predicción*).
- Para cada tupla de entrenamiento, los **pesos son modificados** para **minimizar el error cuadrático medio** entre la predicción de la red y el valor objetivo actual.
- Estas modificaciones se realizan en la dirección ***"hacia atrás"***, es decir, de la capa de salida hacia las capas ocultas (*de la última a la primera capa oculta*)
- A pesar de que no se garantiza, los pesos eventualmente convergen y el proceso de aprendizaje se detiene.

…Retropropagación

- ***Inicializando los pesos***
  - Los pesos en la red son inicializados con **números aleatorios pequeños** (*por ejemplo de* ***-1.0 a 1.0 o de -0.5 a 0.5***)
  - Cada unidad tiene un **sesgo** y estos sesgos son similarmente inicializados con números aleatorios pequeños.

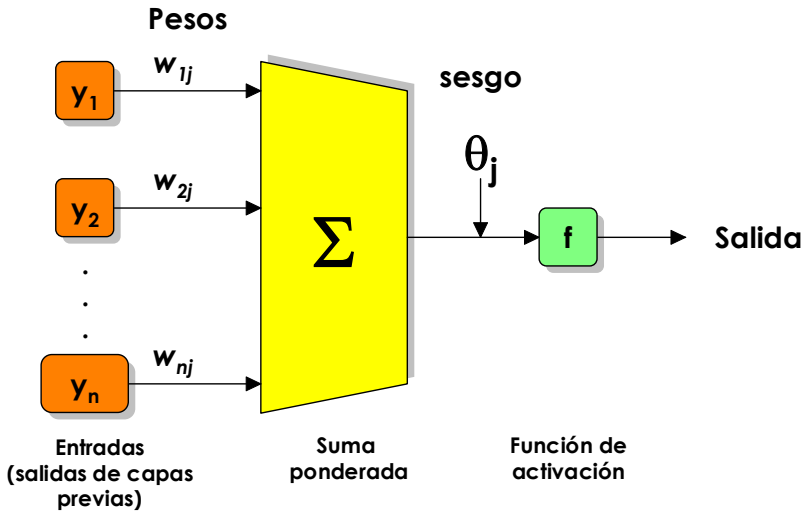
- ***Propagando las entradas hacia adelante***
  - En primer lugar, la tupla de entrenamiento se alimenta a la capa de entrada de la red.
  - Las entradas pasan a través de las unidades de entrada, sin cambios. Es decir, para una unidad de entrada **j**, su salida, $\mathbf{O_j}$, es igual a su valor de entrada, $\mathbf{I_j}$. A continuación, la entrada de red y la salida de cada unidad en las capas ocultas y de salida son calculadas.
  - La entrada de red a una unidad en las capas ocultas o de salida se calcula como una **combinación lineal** de sus entradas.

...Retropropagación



- Cada una de las unidades tiene un número de entradas a la misma que son, de hecho, las salidas de las unidades conectadas a la unidad en la capa anterior.
- **Cada conexión tiene un peso**. Para calcular la entrada de red a la unidad, cada entrada conectada a la unidad se multiplica por su correspondiente peso, y ésto es sumado.

...Retropropagación



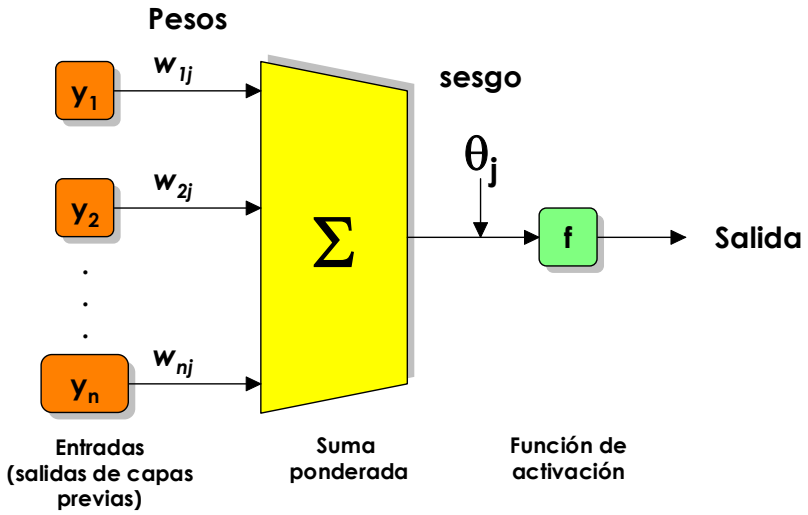
- Dada una unidad **j** en una capa oculta o de salida, la entrada a la red $\mathbf{I_j}$ a la unidad **j** es:

$$I_j = \sum_i w_{ij} O_j + \theta_j$$

donde

- $\mathbf{w_{ij}}$ es el peso de la conexión de la unidad **i** en la capa anterior a la unidad **j**;
- $\mathbf{O_i}$ es la salida de la unidad **i** de la capa anterior y,
- θ_j es el sesgo de la unidad. El **sesgo** actúa como un **umbral** que sirve para variar la actividad de la unidad.

...Retropropagación

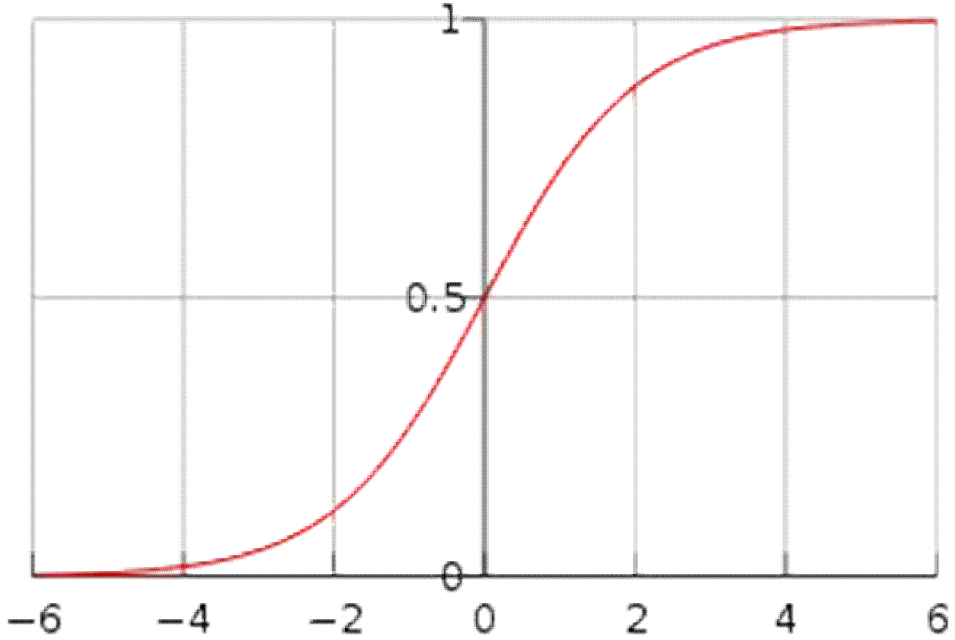


- Cada unidad en las capas ocultas y de salida toma su entrada de red y a continuación, se aplica una función de activación:
  - **La función simboliza la activación de una neurona,** representada por la unidad.
  - Se suele utilizar la **función sigmoide** (caso particular de la **función logística**).

- Dada la entrada de red $\mathbf{I_j}$ a la unidad **j**, la salida de la unidad, $\mathbf{O_j}$, se calcula como:

$$O_j = \frac{1}{1 + e^{-I_j}}$$

...Retropropagación



- Esta función también se conoce como una **función de aplastamiento**, porque **mapea** un dominio de entrada de gran tamaño en un rango pequeño (***de 0 a 1***).
- La **función logística** es **no lineal** y **diferenciable**, permitiendo que el algoritmo de retropropagación pueda aplicarse a problemas de clasificación que son linealmente inseparables.

…Retropropagación

- Se deben calcular los valores de salida, $\mathbf{O_j}$, para cada capa oculta, incluyendo la capa de salida, lo que da la predicción de la red.
- En la práctica, es una buena idea **guardar los valores de salida** intermedios en cada unidad, ya que se requieren más tarde, para calcular el error de retropropagación.
- Este truco puede reducir sustancialmente la cantidad de cálculos requeridos.

Retropropagación del error

- El error se propaga hacia atrás mediante la actualización de los pesos y sesgos para reflejar el error de la predicción de la red.
- Para una **unidad j** en la **capa de salida**, el error $\mathbf{Err_j}$ se calcula:

$$Err_j = O_j\left(1 - O_j\right)\left(T_j - O_j\right)$$

Donde:

- $\mathbf{O_j}$ es la salida actual de la unidad **j**
- $\mathbf{T_j}$ es el valor objetivo conocido de la tupla de entrenamiento.

- Es importante notar que $\mathbf{O_j(1 - O_j)}$ es la derivada de la función logística.

…Retropropagación del error

- Para calcular el error de una capa oculta unidad **j**, se considera la suma ponderada de los errores de las unidades conectadas a la unidad **j** en la siguiente capa.
- El error de una unidad **j** en una capa oculta es:

$$Err_j = O_j\left(1 - O_j\right)\sum_k Err_k w_{jk}$$

Donde:

- $\mathbf{w_{kj}}$ es el peso de la conexión de la unidad **j** a la unidad **k** en la siguiente capa más alta
- $\mathbf{Err_k}$ es el error de la unidad **k**.

…Retropropagación del error

- Los sesgos y los pesos se actualizan para reflejar el error de **propagación**.
- El **peso** se actualiza a través de las siguientes ecuaciones, donde Δw_{ij} es el cambio en el peso w_{ij} :

$$\Delta w_{ij} = (\ell) Err_j O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

Donde:

- ℓ es la **tasa de aprendizaje** (constante que tiene típicamente un valor entre **0.0** y **1.0**.)

- **Retropropagación** aprende utilizando un método de **descenso de gradiente** para buscar un conjunto de pesos que se ajuste a los datos de entrenamiento con el fin de **minimizar el cuadrado de la distancia** entre predicción de clase de la red y el valor objetivo conocido de las tuplas.

…Retropropagación del error

- La **tasa de aprendizaje** ayuda a evitar **quedar atrapado** en un **mínimo local** en el margen de decisión (*es decir, donde los pesos parecen converger, pero no son la solución óptima*) y alienta a encontrar el mínimo global:

  - Si la **tasa de aprendizaje** es **demasiado pequeña**, entonces el aprendizaje se produce a un **ritmo muy lento**.
  - Si la **tasa de aprendizaje** es **demasiado grande**, entonces puede producirse **la oscilación entre las soluciones inadecuadas**.

- Una **regla de oro** es ajustar la tasa de aprendizaje a **1/t**, donde **t** es el número de iteraciones a través del conjunto de entrenamiento hasta el momento.

…Retropropagación del error

- Los **sesgos** se actualiza a través de las siguientes ecuaciones, donde $\Delta\boldsymbol{\theta}_j$ es el cambio en el sesgo $\boldsymbol{\theta}_j$:

$$\Delta\theta_j = (\ell)Err_j$$

$$\theta_j = \theta_j + \Delta\theta_j$$

- En este caso se están actualizando los pesos y sesgos después de la presentación de cada tupla → **actualización de casos**.
- Alternativamente, los incrementos de peso y sesgo se podrían **acumular en variables**, de modo que éstos actualicen después de que todas las tuplas se presentaron → **actualización de épocas.**
- **Una iteración** a través del conjunto de entrenamiento es una **época**.
- En la práctica, la **actualización de casos** es más común, ya que tiende a dar resultados más precisos.

Condiciones de terminación

- El aprendizaje termina cuando:
  - Todos Δw_{ij} en la época anterior son tan pequeños como para estar por debajo de un umbral especificado.
  - El **porcentaje de tuplas mal clasificados** en la época anterior está por debajo de cierto umbral.
  - Un **número de épocas**, especificado de antemano, ha expirado.
- En la práctica, varios cientos de miles de épocas pueden ser necesarios antes de los pesos lleguen a converger.

Eficiencia

- La eficiencia computacional depende del tiempo que se gaste en entrenar la red.
- Dadas **|D|** tuplas y **w** pesos, cada época requiere un tiempo **O(|D|w)**.
- En el peor de los casos, el número de épocas puede ser exponencial a **n**, donde **n** es el número de entradas.
- En la práctica, el tiempo requerido para las redes puedan converger es muy variable.

Cantidad de neuronas

- La **cantidad de neuronas** de **entrada y salida** están definidas por el problema.
- La **cantidad de neuronas** en las **capas ocultas** determinan los **grados de libertad** del modelo:
  - Un **número muy pequeño** de neuronas pueden **no ser suficientes** para problemas complejos.
  - Un **número muy grande** de neuronas puede **sobre-entrenar** el modelo y tener una **perdida de generalidad** ante nuevas observaciones.
- Se suele usar (para comenzar el **análisis de sensibilidad**) la regla:

$$M = \frac{A + K}{2}$$

Donde:

- **M** es la **cantidad de neuronas** en la **capa media.**
- **A** es la **cantidad de neuronas** en la **capa de entrada.**
- **K** la **cantidad de neuronas** en la **capa de salida**.

…Cantidad de neuronas

- La **cantidad de neuronas** en la **capa oculta** depende de una serie de factores:
  - La **cantidad de observaciones** en el conjunto de entrenamiento.
  - La **cantidad de ruido**, **complejidad** del problema de clasificación.
  - La **cantidad de atributos de entrada** y **clases** (**neuronas de salida**)
  - Las **funciones de activación** entre las capas y algoritmo de entrenamiento.
- Una opción es ir **evaluando** varias redes neuronales para ir determinando el número apropiado de neuronas.
- Otra opción es comenzar con un **número grande** de **neuronas** y **conexiones**, y a medida que se va construyendo la red neuronal, se van **"podando"** aquellas conexiones que son innecesarias.

Decaimiento de los pesos

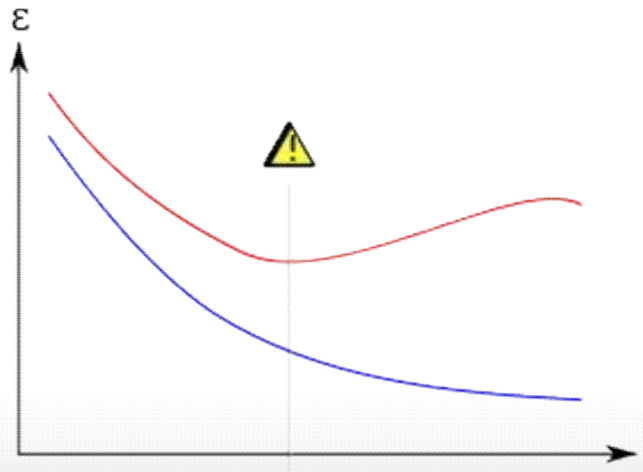
- Para prevenir que los pesos vayan **creciendo sin control** alguno a valores muy grandes (señal de sobre-entrenamiento), es conveniente agregar un decaimiento a los pesos de la forma:

$$w_{t+1} = (1 - \varepsilon) w_t$$

- Pesos que no son necesarios y no se van actualizando en cada iteración del algoritmo, van a **decaer hasta anularse**, mientras que aquellos que si son necesarios y se van actualizando de manera continua con **retropropagacion** y ajustándose con el decaimiento.

Número de épocas

- Para evitar el s**obre-entrenamiento** y el **tiempo computacional** necesario para entrenar la red, se puede **fijar** un **cierto número de épocas** de entrenamiento de acuerdo al comportamiento observado del error de entrenamiento y de prueba.
- Se puede dar el caso que sea necesario **agregar ruido** a las observaciones de entrenamiento de manera de entregar una **mayor generalidad** al modelo.
- Una **red neuronal MPL** (*Perceptrón Multicapa*) entrenada con **retropropagación** entrena generalmente más rápido si se utiliza una función de **activación antisimétrica** (f(-x) = -f(x)).



Pre-procesamiento de datos

- Los datos de entrada deben estar **pre-procesados** de manera que su **media sea cero** (o un valor muy bajo) con respecto a la **varianza**.
- Los datos de entrada **no deben estar correlacionados**. Intentar utilizar variables que no expliquen las mismas características de las observaciones de una BD *(se puede utilizar PCA o algún método que asegure variables independientes)*.
- Las **variables de entrada** deben tener **varianza similar**.
- Los **pesos iniciales** deben ser **valores pequeños** para ***evitar la saturación de las neuronas***:
  - Los valores de los pesos de **Capa Entrada → Capa media** deben ser **mayores**, que aquellos que van de **Capa media → Capa de salida.**

Actualización de pesos

- Existen dos aproximaciones básicas para actualizar los pesos durante el entrenamiento de la red:
  - **On-Line:** Los pesos son actualizados con **retropopagación** luego de que **cada observación es presentada** a la red neuronal.
  - **Batch:** Los pesos son actualizados una vez que **todas las observaciones son presentadas** a la red neuronal.
- Se recomienda el entrenamiento **Batch** dado que se utiliza la **dirección de máximo descenso**, la más adecuada para el problema de optimización no lineal que se desea resolver.
- El entrenamiento **On-line** tiene una menor **complejidad computacional** del entrenamiento de una orden y **es sensible al orden** que se presentan las observaciones.

Tasa de aprendizaje y Momentum

- Se recomienda utilizar **una combinación de tasas de aprendizaje** sobre distintas redes.
- Este parámetro, a grandes rasgos, permite **definir la velocidad** por sobre la cual se va acercando al óptimo del problema definido sobre una red neuronal.
- Se puede incluir un parámetro llamado **momentum** que suele utilizarse para la **actualización de los** pesos en el algoritmo de retropropagación.
- Permite considerar la **cantidad de movimiento** que cada peso tiene al irse actualizando.
- **No existe una regla general** para los valores de ambos parámetros, pero para el **momentum** se recomiendan valores cercanos a **0.2**.