INTELIGENCIA ARTIFICIAL

LUIS EDIN IBARGUEN MOSQUERA

Taller N°. 2

ING. CARLOS ALBERTO LONDOÑO LOAIZA

CORPORACIÓN DE ESTUDIOS

TECNOLÓGICOS

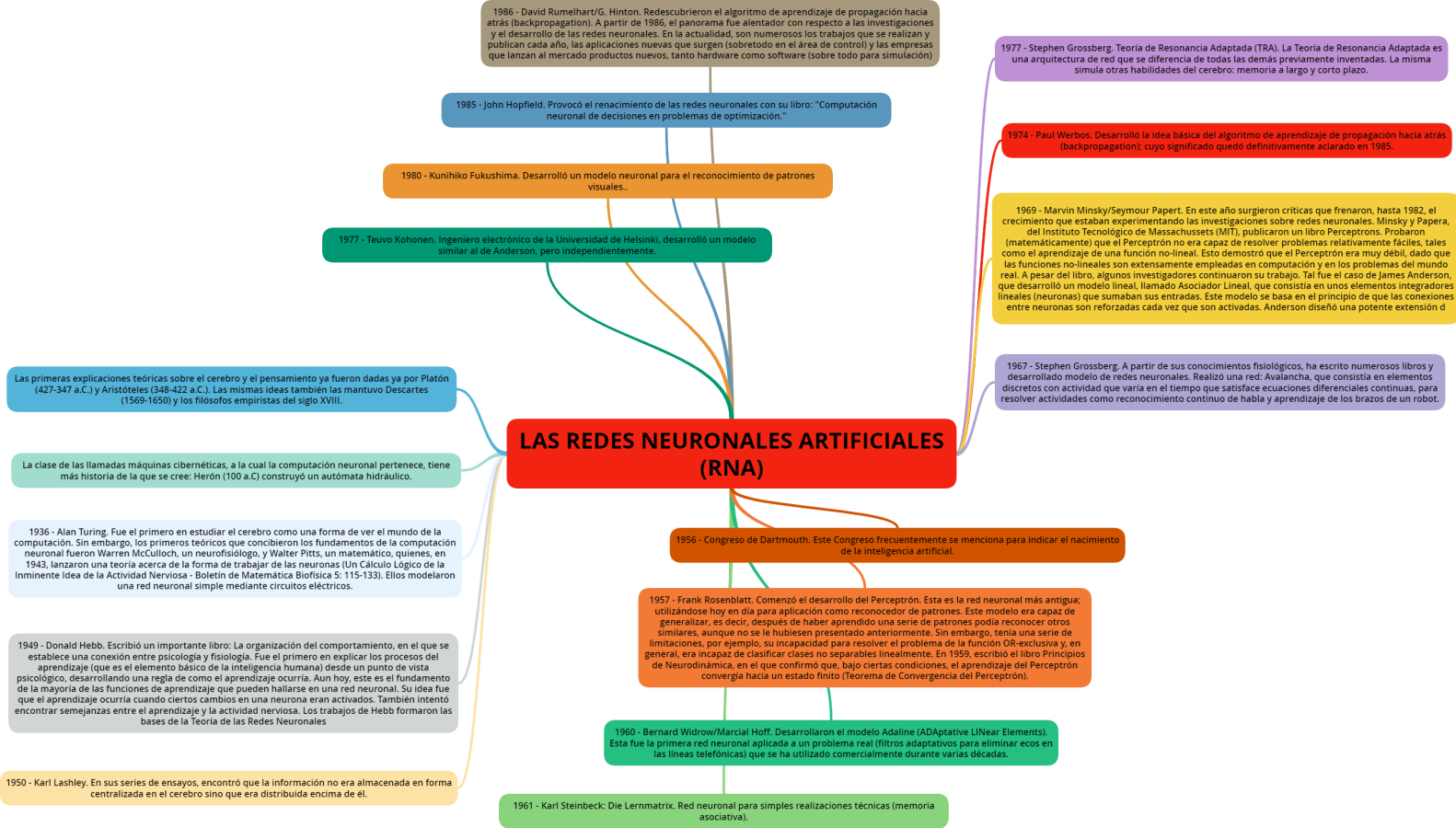
DEL NORTE DEL VALLE

CARTAGO

TECNOLOGÍA EN SISTEMAS DE INFORMACIÓN

2018

**DESARROLLO**

**1.**

https://www.goconqr.com/es/p/12467043-LAS-REDES-NEURONALES-ARTIFICIALES--RNA--mind\_maps

2. **Ventajas de las redes neuronales**

Tienen capacidad de aprender a realizar tareas basadas en un entrenamiento inicial (***Aprendizaje adaptativo***). De esta forma, la máquina puede aprender a llevar a cabo ciertas tareas mediante el entrenamiento con ejemplos ilustrativos, por lo que no es necesario elaborar modelos a priori.

Las redes neuronales permiten organizar por sí mismas lo aprendido. Mientras que el aprendizaje es la modificación de cada elemento procesal, la **autoorganización** consiste en la modificación de la red neuronal completa para llevar a cabo un  objetivo específico.

Además las redes neuronales tienen la capacidad de ser **tolerantes *a* fallos**. Fueron los primeros métodos computacionales con esta capacidad inherente. De este modo, si se produce un fallo en un número no muy grande de neuronas, aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina.

Una de las prioridades principales de las redes neuronales es que son capaces de **operaren *tiempo real***. Las redes neuronales trabajan mediante conexiones en paralelo, lo que permite grandes velocidades de transmisión y respuesta casi al instante.

Hay que destacar como ventaja la **facilidad de inserción en la tecnología existente**. De manera que, por ejemplo, resulta muy sencillo conseguir *chips* especializados que mejoren su capacidad.

En general, las redes neuronales son superiores a otras técnicas porque:

• Se entrenan, auto organizan, aprenden y olvidan.

• Son robustas y tolerantes a fallas; la falla de una o varias neuronas no implica un fallo total en la red neuronal.

• Son flexibles, lo que les permite adaptarse fácilmente a nuevos ambientes, ya que pueden catalogarse como sistemas independientes.

• Se emplean en datos en los cuales el patrón es oscuro e imperceptible, que exhiben comportamiento impredictible o no lineal, como en modelos tradicionales de series temporales y datos caóticos.

• La velocidad de respuesta es menor que la del cerebro humano.

• Son hábiles en el proceso de asociar, evaluar o reconocer patrones

**DESVENTAJAS DE LAS REDES NEURONALES**

* Funcionan como una caja negra, resuelven un problema, pero es difícil saber cómo lo han hecho. ¿Cómo hace, por ejemplo, una red neuronal un algoritmo de scoring? En lugar de ser un sistema de apoyo a la decisión, la caja negra se puede convertir en el "tomador" de la decisión. Puede ocurrir que un director de riesgo niegue un crédito sólo porque se lo dice la caja negra, sin que él pueda argumentar esta decisión ya que no entiende el funcionamiento de la red neuronal.
* No resuelven todos los problemas ni siempre los resuelven de la mejor manera.
* Tienen problemas en la estimación de cálculos precisos. Funcionan bien con problemas complejos de difícil cálculo pero que no requieren respuestas perfectas, sino solo respuestas rápidas y buenas, como ocurre en la bolsa, en donde se necesita saber con rapidez si conviene comprar, vender o mantener.
* En predicción bursátil se emplean con frecuencia en boletines divulgativos de las entidades financieras, lo que significa que su éxito es relativo, puesto que en otro caso su predicción no se publicaría, sino que se aprovecharía.
* Las redes neuronales se están comportando bien en predicción a largo plazo con componentes no lineales; en cambio, no están claras las mejorías observadas en series cortas y estacionales, como son típicas en las predicciones de ventas.
* Complejidad de aprendizaje para grandes tareas, cuanto más cosas se necesiten que aprenda una red, mas complicado será enseñarle.
* Tiempo de aprendizaje elevado. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar y segundo si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir mas tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera enseñar.
* No permite interpretar lo que se ha aprendido, la red por si sola proporciona una salida, un número, que no puede ser interpretado por ella misma, sino que se requiere de la intervención del programador y de la aplicación en si para encontrarle un significado a la salida proporcionada.
* Elevada cantidad de datos para el entrenamiento, cuanto más flexible se requiera que sea la red neuronal, mas información tendrá que enseñarle para que realice de forma adecuada la identificación.
* Otros problemas con las redes neuronales son la falta de reglas definitorias que ayuden a realizar una red para un problema dado.

3. **Aplicaciones de las Redes Neuronales:**

Las redes neuronales son una tecnología computacional que puede utilizarse en un gran número y variedad de aplicaciones, tanto comerciales como militares. Cuando se implementan mediante hardware (redes neuronales en chips VLSI), presentan una alta tolerancia a fallos del sistema y proporcionan un grado de paralelismo en el proceso de datos muy grande.

Hay muchos tipos de redes neuronales pudiéndose clasificar en distintas disciplinas:

**Biología**

* Aprender más acerca del cerebro
* Obtención del modelo de la retina

**Negocios**

* Marketing
* Venta cruzada
* Campanas de venta

**Finanzas**

* Predicción de índices
* Detección de fraudes.
* Riesgo crediticio, clasificación
* Predicción de la rentabilidad de acciones

**Tratamiento de textos y proceso de formas.-**

* Reconocimiento de caracteres impresos mecánicamente.
* Reconocimiento de gráficos.
* Reconocimiento de caracteres escritos a mano.
* Reconocimiento de escritura manual cursiva.

**Alimentación**

* Análisis de olor y aroma.
* Perfilamiento de clientes en función de la compra.
* Desarrollo de productos.
* Control de Calidad–Predicción consumo de gas ciudad

**Industria manufacturera.**

* Control de procesos.
* Control de calidad.
* Control de robots.

**Medicina y salud**

* Ayuda al diagnóstico.
* Análisis de Imágenes.
* Desarrollo de medicamentos.
* Distribución de recursos.

**Ciencia e Ingeniería.**

* Análisis de datos y clasificación
* Ingeniería Química.
* Ingeniería Eléctrica.
* Climatología.

**Transportes y Comunicaciones.**

* Optimización de rutas.
* Optimización en la distribución de recursos

**Militares**

* Clasificación de las señales de radar
* Creación de armas inteligentes
* Reconocimiento y seguimiento en el tiro al blanco

**Aplicaciones Reales**

Pronunciación: NETtalk (87), aprende a pronunciaer texto escrito. 29 unidades de entrada (26 letras, más espacios, puntos, comas, ...). 80 unidades ocultas. 1024-palabras de entrenamiento y 95% de éxito en el entrenamiento, pero 78% en la prueba.

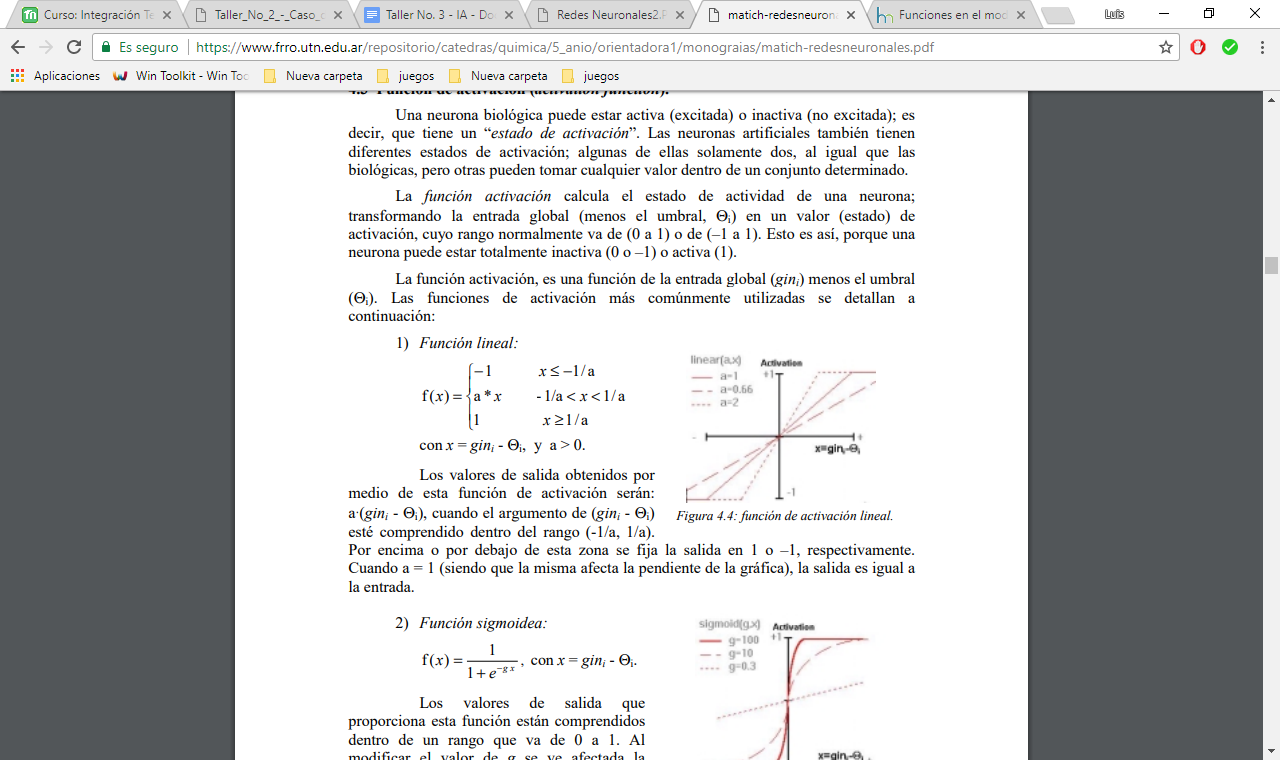
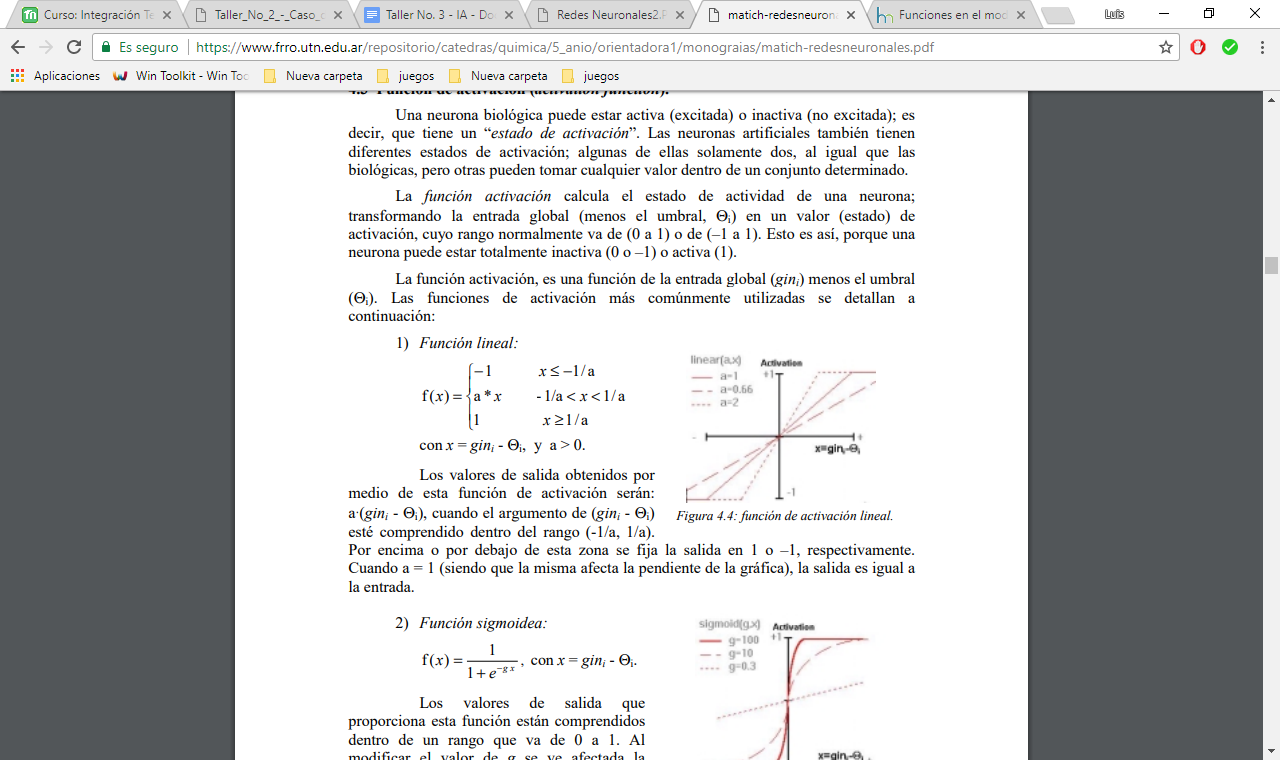
Reconocimiento de caracteres: una de las aplicaciones más grandes de redes neuronales actuales (Le Cun et al. 89). Una red que lee códigos postales en cartas escritas a mano. El sistema tiene un prepocesador para localizar los números, y la red los descifra. 3 capas ocultas (768, 192 y 30 unidades cada una). No se conectaron todas las unidades contra todas, si no que se dió un efecto de detectores de atributos, dividiendo las unidades ocultas en grupos (un total de 9,760 conecciones). Logra un 99% de éxito, adecuado para un sistema de correo automático y se ha implementado en un chip.

Manejar: ALVINN (Autonomous Land Vehicle In a Neural Network) (Pomerleau 93) es una red neuronal que aprende a manejar un vehículo viendo como maneja un humano. Maneja dos vehículos equipados especialmente. Se utiliza una cámara que alimenta una rejilla de entradas a la red. La salida (30 unidades) controla la dirección del volante. La red tiene 5 capas ocultas totalmente conectadas. Después de que gente maneja el vehículo y se entrena al sistema (con retro-propagación, por cerca de 10 min.) el sistema está listo para manejar. Puede manejar hasta a 70 mph por distancias de hasta 90 millas. Extensiones: MANIAC.

4.

**Función de activación (activation function).** Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un “estado de activación”. Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado. La función activación calcula el estado de actividad de una neurona; transformando la entrada global (menos el umbral, Θi) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1). Esto es así, porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1). La función activación, es una función de la entrada global (gini) menos el umbral (Θi). Las funciones de activación más comúnmente utilizadas se detallan a continuación:

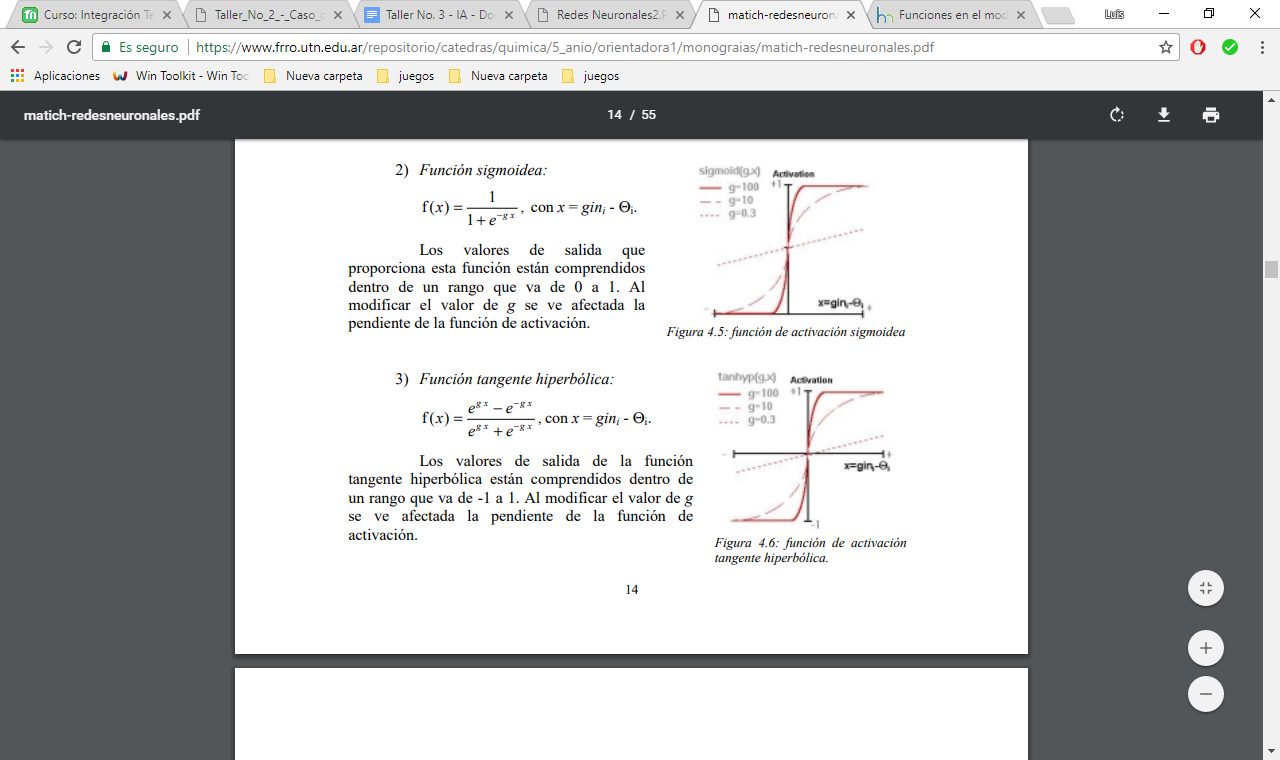
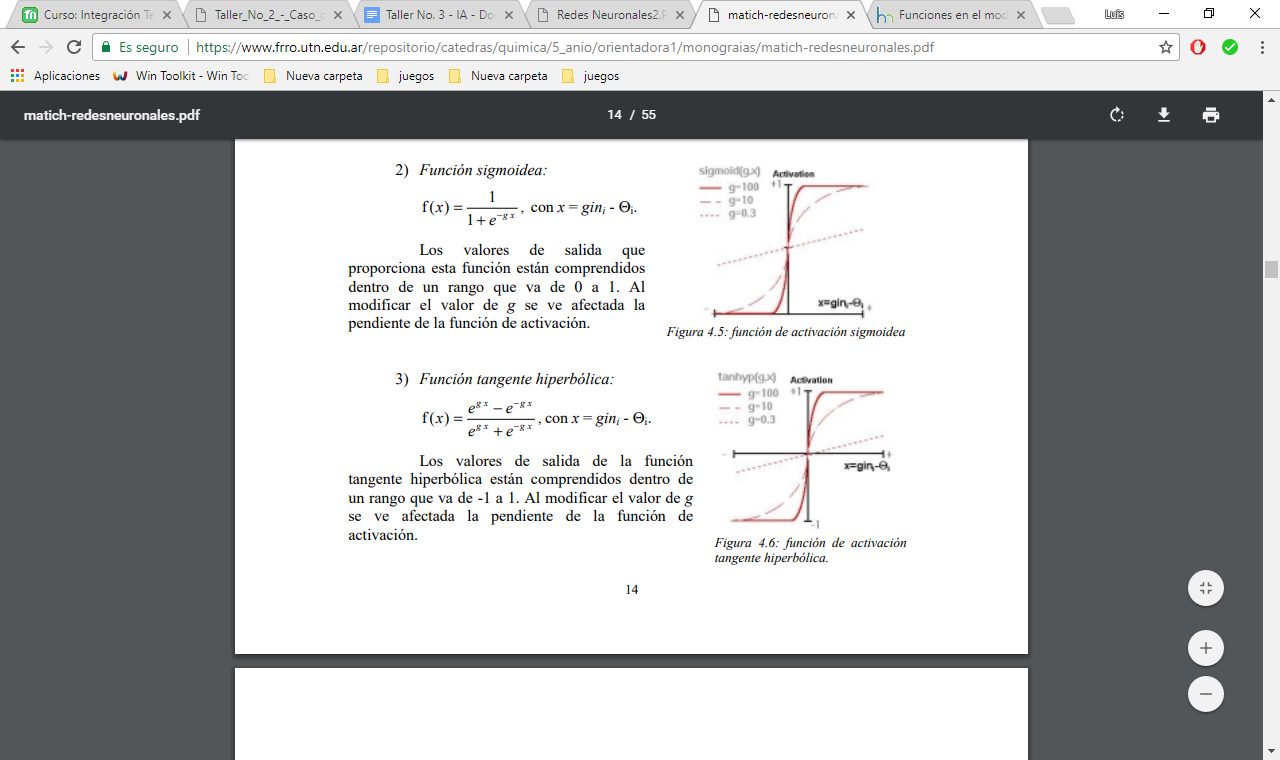
1) Función lineal:



1 función de activación lineal.

Los valores de salida obtenidos por medio de esta función de activación serán: a·(gini - Θi), cuando el argumento de (gini - Θi) esté comprendido dentro del rango (-1/a, 1/a). Por encima o por debajo de esta zona se fija la salida en 1 o –1, respectivamente. Cuando a = 1 (siendo que la misma afecta la pendiente de la gráfica), la salida es igual a la entrada.

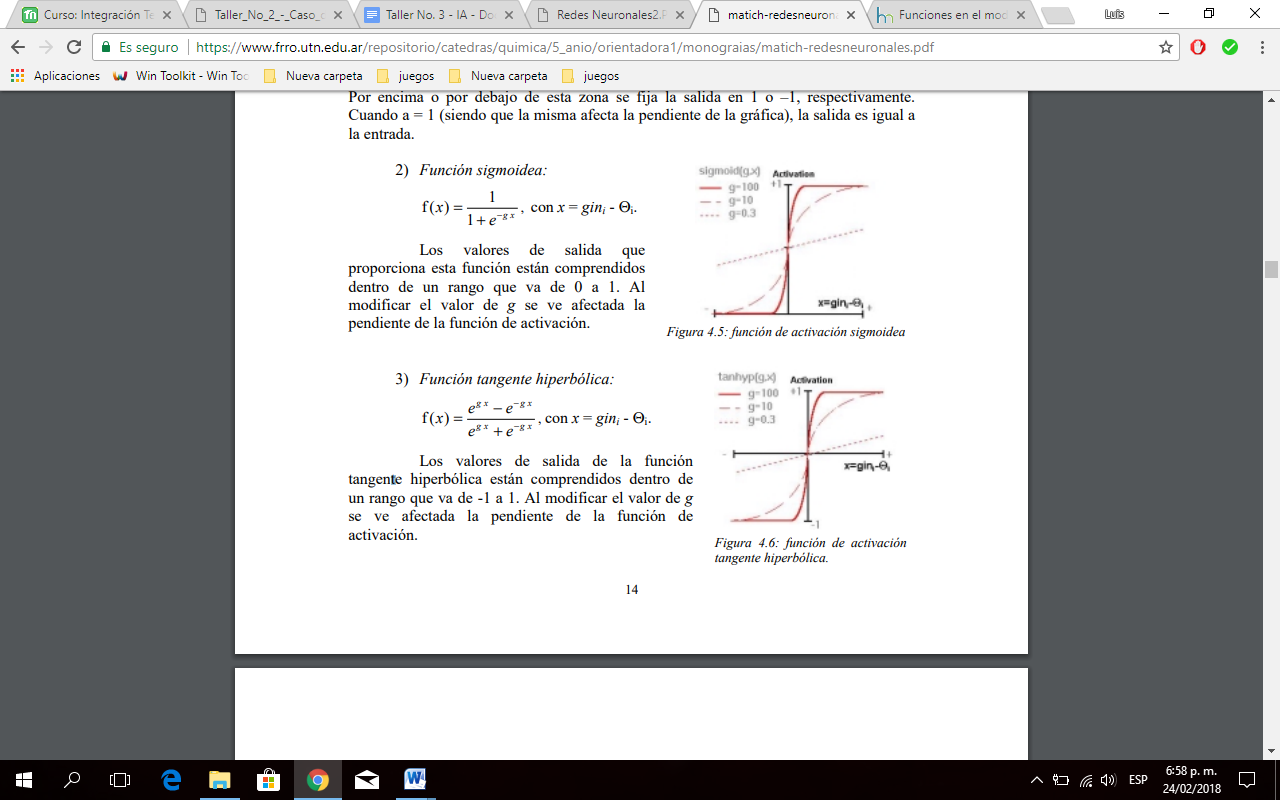
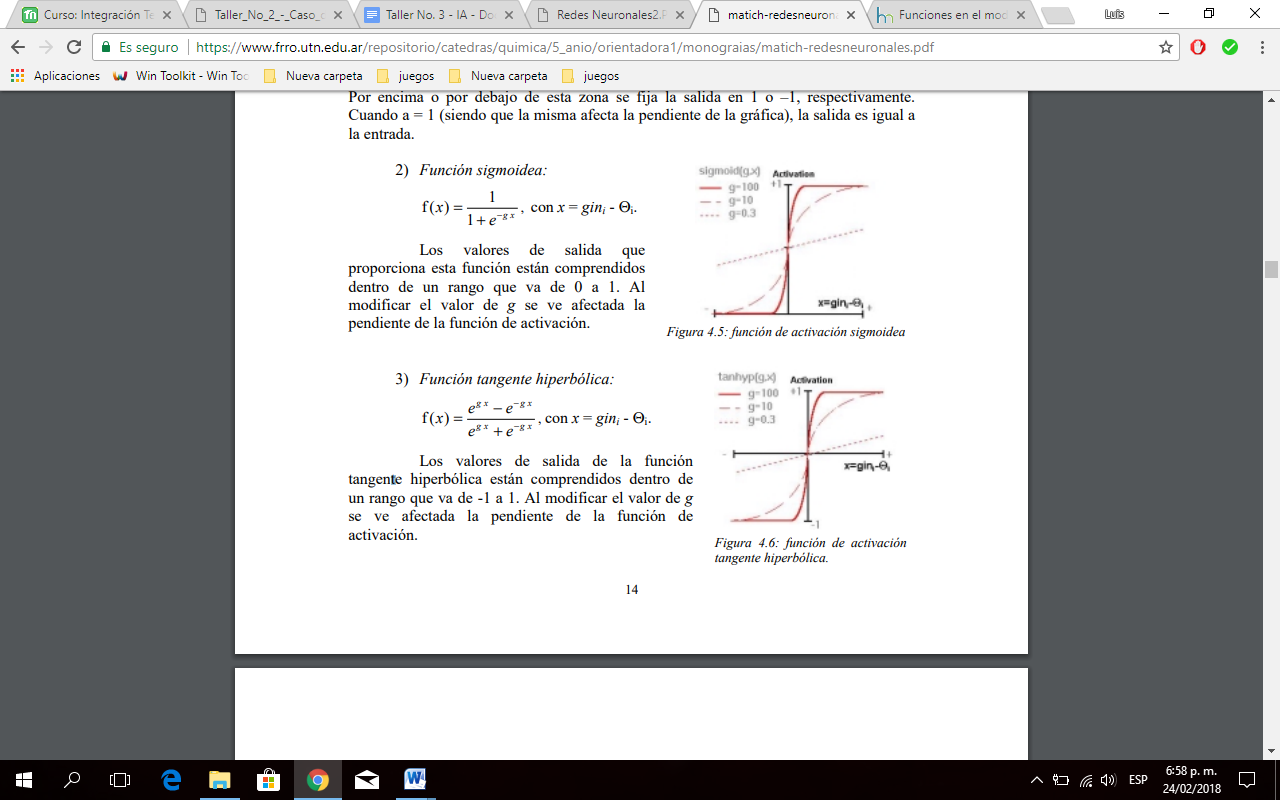
2) Función sigmoidea:



2 función de activación sigmoidea.

Los valores de salida que proporciona esta función están comprendidos dentro de un rango que va de 0 a 1. Al modificar el valor de g se ve afectada la pendiente de la función de activación.

3) Función tangente hiperbólica:



3 función de activación tangente hiperbólica.

Los valores de salida de la función tangente hiperbólica están comprendidos dentro de un rango que va de -1 a 1. Al modificar el valor de g se ve afectada la pendiente de la función de activación.

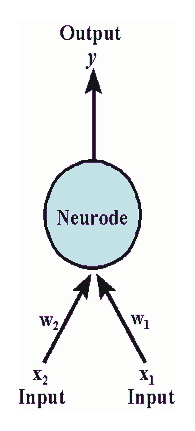
Para explicar porque se utilizan estas funciones de activación se suele emplear la analogía a la aceleración de un automóvil. Cuando un auto inicia su movimiento necesita una potencia elevada para comenzar a acelerar. Pero al ir tomando velocidad, este demanda un menor incremento de dicha potencia para mantener la aceleración. Al llegar a altas velocidades, nuevamente un amplio incremento en la potencia es necesario para obtener una pequeña ganancia de velocidad. En resumen, en ambos extremos del rango de aceleración de un automóvil se demanda una mayor potencia para la aceleración que en la mitad de dicho rango.

**5. Perceptrón**

Historia del Perceptrón

La evolución de la neurona artificial ha progresado a través de varias etapas.   Las raíces de las cuales, están cimentadas dentro trabajo neurológica realiza principalmente por [Santiago Ramón y Cajal](http://www.uic.edu/depts/mcne/founders/page0076.html) y [Sir Charles Scott Sherrington](http://www.uic.edu/depts/mcne/founders/page0086.html) .   Ramón y Cajal fue una figura prominente en la exploración de la estructura de tejido nervioso y demostró que, a pesar de su capacidad para comunicarse entre sí, las neuronas se separan físicamente de otras neuronas.   Con una mayor comprensión de los elementos básicos del cerebro, se hicieron esfuerzos para describir cómo estas neuronas básicas podrían dar lugar a conductas manifiestas, a la que [William James](http://www.emory.edu/EDUCATION/mfp/james.html) fue un destacado colaborador teórico.

Trabajando a partir de los comienzos de la neurociencia, [Warren McCulloch](http://www.csulb.edu/~cwallis/artificialn/warren_mcculloch.html) y [Walter Pitts](http://www.csulb.edu/~cwallis/artificialn/walter_pitts.html)   en su artículo de 1943, "Un cálculo lógico de las ideas inmanentes en la actividad nerviosa", sostuvieron que las neuronas con una función de activación de umbral binario eran análogas a las oraciones lógica de primer orden.   La McCulloch básica y Pitts neurona se veía algo como lo siguiente:



La neurona McCulloch-Pitts trabajado mediante la introducción de un 1 o 0 para cada una de las entradas, en donde 1 representa verdadero y 0 falsa.   Del mismo modo, el umbral se le dio un valor real, por ejemplo 1, lo que permitiría un 0 o 1 de salida si el umbral se alcanza o se supera.   Por lo tanto, con el fin de representar la "y" función, fijar el umbral en 2,0 y llegar a la siguiente tabla de verdad:

|  |  |  |
| --- | --- | --- |
| **De entrada x 1** | **De entrada x 2** | **Salida** |
| **0** | **0** | **0** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

Esta tabla muestra la "y" función básica tal que, si x1 y x2 son falsas, entonces la salida de la combinación de estos dos también será falsa.   Del mismo modo, si x1 es verdadera o igual a 1 y x2 es verdadera o igual a 1 , entonces el umbral del 2 será recibido y la salida será 1.

Esto se deduce también de la función "o", si cambiamos el valor de umbral a 1.   La tabla de la "o" ser función,

|  |  |  |
| --- | --- | --- |
| **De entrada x 1** | **De entrada x 2** | **Salida** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **1** |

Este tipo de neurona artificial también se podría utilizar para resolver la función "no", lo que tendría sólo una entrada, así como, las funciones NOR y NAND.   La neurona McCulloch-Pitts, por lo tanto, fue muy instrumental en el progreso de la neurona artificial, pero tiene algunas limitaciones graves.   En particular, se podría resolver ni la función "exclusiva o" función (XOR), ni la "exclusiva ni" (XNOR).  Limitado a código binario, las siguientes tablas de verdad podría *no* ser resuelto con precisión el uso de esta neurona artificial temprano.

**XOR**

|  |  |  |
| --- | --- | --- |
| **De entrada x 1** | **De entrada x 2** | **Salida** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

**XNOR**

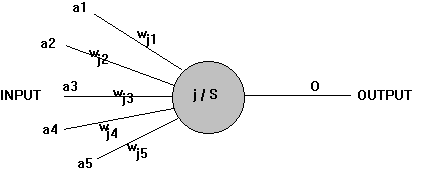
|  |  |  |
| --- | --- | --- |
| **De entrada x 1** | **De entrada x 2** | **Salida** |
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

Una de las dificultades con la neurona de McCulloch-Pitts fue su sencillez.   Sólo se permite para las entradas y salidas binarias, sólo se utiliza la función de activación de la etapa de umbral y no ha incorporado la ponderación de las diferentes entradas.

En 1949, [Donald Hebb](http://cogprints.ecs.soton.ac.uk/~harnad/Archive/hebb.html) podría ayudar a revolucionar la forma en que se perciben las neuronas artificiales.   En su libro, *La organización de la conducta*, propuso lo que ha llegado a ser conocido como la regla de Hebb.   Él afirma: "Cuando un axón de la célula A está cerca suficiente para excitar a una célula B y repetidamente o persistentemente toma parte en la cocción, algunos procesos de crecimiento o cambio metabólico se lleva a cabo en una o ambas células de tal manera que la eficiencia de a, como una de las celdas de disparo B, se incrementa. " [[1]](https://web.csulb.edu/~cwallis/artificialn/History.htm" \l "_ftn1" \o ")   Hebb se propone no sólo eso, cuando dos neuronas se activan conjuntamente la conexión entre las neuronas se fortalece, sino también que esta actividad es una de las operaciones fundamentales necesarias para el aprendizaje y la memoria.

Para la neurona artificial, esto significaba que la neurona de McCulloch-Pitts tuvo que ser modificado para permitir al menos la esta nueva propuesta biológica.   El método empleado fue ponderar cada una de las entradas.   Por lo tanto, una entrada de 1 puede darse más o menos peso, con respecto a la suma de umbral total.

Frank Rosenblatt, utilizando la neurona de McCulloch-Pitts y los resultados de Hebb, pasó a desarrollar el primer perceptrón.   Este perceptrón, lo que podría aprender en el sentido Hebbean, a través de la ponderación de los insumos, jugó un papel decisivo en la formación posterior de las redes neuronales.   Se refirió a la perceptrón en su libro de 1962, *Principios de Neurodinámico*.   Un perceptrón básico se representa como sigue:



Este perceptrón tiene un total de cinco entradas A1 a A5 y cada una tiene un peso de W1 a través w5. [[2]](https://web.csulb.edu/~cwallis/artificialn/History.htm" \l "_ftn2" \o ")   Cada una de las entradas son ponderados y sumados en el nodo.   Si se alcanza el umbral, se produce una salida.   De gran importancia es que cada una de las entradas que no se puede dar el mismo peso.   El perceptrón puede haber "aprendido" a peso A1 a A2 y así sucesivamente.

La fórmula de sumación para determinar si es o no el umbral ( *θ* ) se reunió por la neurona artificial con N entradas (un 1 , un 2 ... a N ) y sus respectivos pesos de w 1 , w 2 , ... w Nes:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **norte** |  |  |  |  |  |  |  |
| segundo | = | (Σw jun j ) | + | *θ* |  |  |  |  |  |
|  |  | **j = 1** |  |  |  |  |  |  |  |

La función de activación se convierte en:

                        x = f (b)

La función de activación utilizado por McCulloch y Pitts fue la función de paso umbral.   Sin embargo, otras funciones que se pueden utilizar son el sigmoide, lineal por partes y funciones de activación de Gauss.   Estas funciones se muestran a continuación. [[3]](https://web.csulb.edu/~cwallis/artificialn/History.htm" \l "_ftn3" \o ") (Ver el glosario adjunto a este subprograma para las fórmulas matemáticas correspondientes.)

|  |  |  |  |
| --- | --- | --- | --- |
| Paso umbral | Sigmoideo | Lineal a trozos | gaussiano |
| https://web.csulb.edu/~cwallis/labs/artificialn/History_files/image007.gif | https://web.csulb.edu/~cwallis/labs/artificialn/History_files/image009.gif | https://web.csulb.edu/~cwallis/labs/artificialn/History_files/image011.gif | https://web.csulb.edu/~cwallis/labs/artificialn/History_files/image013.gif |

A pesar de los numerosos cambios realizados en el original de la neurona de McCulloch-Pitts, el perceptrón todavía estaba limitado a la solución de ciertas funciones   Desafortunadamente, Rosenblatt era demasiado entusiasta sobre el perceptrón e hizo el anuncio inoportuno que:

"Dada una α-perceptrón elemental, un mundo de estímulos W, y cualquier clasificación C (W) para los que existe una solución, dejar que todos los estímulos en W se producen en cualquier orden, siempre que cada estímulo debe volver a ocurrir en un tiempo finito; a continuación, a partir de un estado inicial arbitrario, un procedimiento de corrección de errores siempre dará lugar a una solución a C (W) en un tiempo finito

Con estos tipos de observaciones Rosenblatt había trazado una línea en la arena entre los de apoyo de perceptrón estilo de investigación y los proyectos de manipulación de símbolos más tradicionales que se realiza por [Marvin Minsky](http://web.media.mit.edu/~minsky/) .   Como resultado, en 1969, Minsky coautoría con [Seymour Papert](http://papert.www.media.mit.edu/people/papert/) , *perceptrones : Una introducción a la geometría computacional* .   En este trabajo se atacó a las limitaciones del perceptrón.   demostraron que el perceptrón sólo se podía resolver funciones linealmente separables.   de particular interés fue el hecho de que el perceptrón todavía no podía resolver las funciones NXOR XOR y.   de la misma manera, Minsky y Papert indicaron que el estilo de la investigación que se realiza en el perceptrón estaba condenado al fracaso debido a estas limitaciones.   Esto fue, por supuesto, la observación igualmente inoportuna de Minsky.   Como resultado, hay muy poca investigación se llevó a cabo en la zona hasta acerca de la década de 1980.

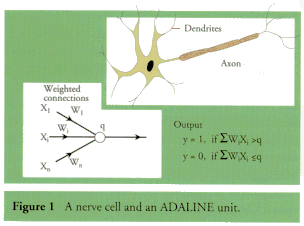
Lo que vendría a resolver muchas de las dificultades fue la creación de redes neuronales.   Estas redes se conectan las entradas de las neuronas artificiales con las salidas de otras neuronas artificiales.  Como resultado, las redes fueron capaces de resolver los problemas más difíciles, pero han crecido considerablemente más complejo.   Sin embargo, muchas de las redes neuronales artificiales en uso hoy en día todavía se derivan de los primeros avances de la neurona de McCulloch-Pitts y el perceptrón de Rosenblatt.

**6. Adaline**

En la década de 1950, con el avance de las computadoras, se hizo posible simular una red neuronal. Nathanial Rochester, de los laboratorios de investigación de la IBM, dio el primer paso en la simulación de la red neuronal, pero desafortunadamente fracaso. Posteriormente se hicieron otras simulaciones ya con éxito.

En 1959, Bernard Widrow y Marcian Hoff de la universidad de Stanford desarrollaron un modelo llamado “ADALINE” y “MADALINE”. ADALINE proviene de ADAptive LINear Element y MADALINE de Multiple ADAptive LINear Element (Many Adalines).

ADALINE fue desarrollado para el reconocimiento de patrones binarios, por ejemplo, predecir el siguiente bit en una línea telefónica. En la figura 1 podemos observar la representación de una ADALINE y su analogía con una Neurona.



ADALINE funciona tomando la suma de los pesos de las entradas y produce una salida con 0 o 1 dependiendo si pasa o no un umbral, esto haciendo analogía al funcionamiento de una neurona que se dispara si la actividad total procedente de las conexiones con las otras neuronas sobrepasa un nivel.

Varias ADALINE pueden ser organizadas en capas de tal manera que se obtengan grandes arquitecturas formando una red MADALINE la cual produce funciones más complicadas.

MADALINE fue la primera red neuronal aplicada a un problema real, se usó como un filtro para eliminar el eco en las líneas telefónicas.

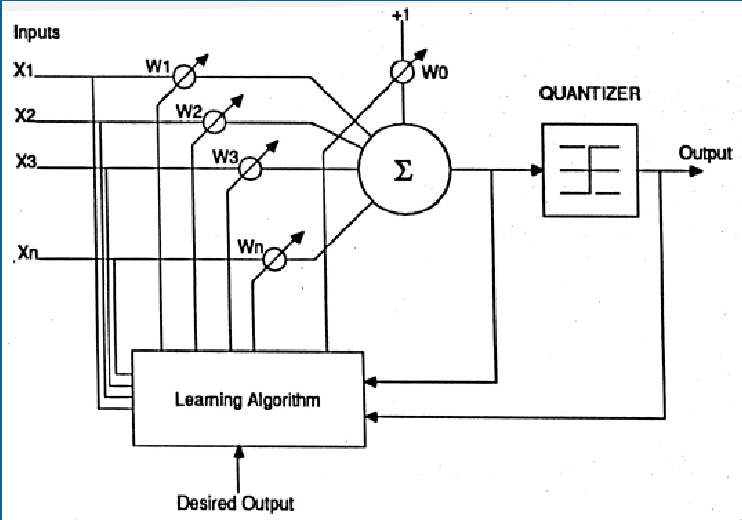
En los años siguientes el neuro-biólogo Frank Rosenblatt comenzó a trabajar con el perceptrón. Se encontró que una simple capa de perceptrones era suficiente para clasificar un conjunto de valores continuos en una de dos clases.

Desafortunadamente el perceptrón estaba limitado y fue probado por Marvin Minsky y Seymour Papert durante los años de desilusión en el libro “Perceptrons”.

Durante los próximos 20 años se dejó atrás el estudio de las redes neuronales y fue hasta 1982 con John Hopfield que se retomó el interés en la materia. Hopfield presentó un documento a la Academia Nacional de Ciencias en el cual a través de análisis matemático mostraba como las redes neuronales funcionan y para que podrían servir.

ADALINE

ADALINE que por sus siglas en inglés significa ADAptive LINear Element es un dispositivo que consta de un solo elemento de procesamiento, por lo que técnicamente no es una red. Sin embargo, es un elemento muy importante, ya que de él se derivan redes más complejas. En la siguiente figura se muestran los elementos de una ADALINE.



ADAptive LINear Element

Una ADALINE consiste de un ALC (Adaptive Linear Combiner) y un cuantizador (función bipolar de salida). Se alimenta con un vector de entrada (valores observados) y con una entrada constante igual a 1 denominada sesgo (bias). Posteriormente se efectúa una suma ponderada de los valores de entrada con sus pesos asociados; si el resultado es positivo, la salida del ADALINE es 1, en caso contrario es 0 (o -1). En consecuencia, ADALINE sólo resuelve adecuadamente problemas binarios linealmente separables.

Algoritmo de Entrenamiento

El entrenamiento del ADALINE está basado en la regla LMS (Least Mean Square) que busca minimizar el error cuadrático medio por medio de la regla delta.

Dado un vector de entrada (x0, ..., xn) donde x0 =1, el correspondiente valor de pesos (w0, ..., wn) y el valor deseado de salida d, el error cuadrático es el siguiente:



El objetivo del método es minimizar dicho error mediante la modificación del vector de pesos (w0, ..., wn) sumándole un **w** de tal forma que nos acerquemos al error mínimo en la dirección del gradiente negativo, es decir, lo más rápidamente posible.

El procedimiento de derivación se presenta a continuación:



Por lo que el error local será reducido más rápidamente si se ajustan los pesos de acuerdo a la regla delta:



El algoritmo de entrenamiento se presenta a continuación:

1.Inicializar pesos (w1, ..., wn) y threshold (w0)

2.Presentar vector de entrada (x1,...,xn) y la salida deseada d(t)

3.Calcular la salida



donde Fh(a) = 1 sí a>0 y

= -1 sí a<=0

4. Adaptar los pesos



donde 0 < i < n y  es la tasa de aprendizaje

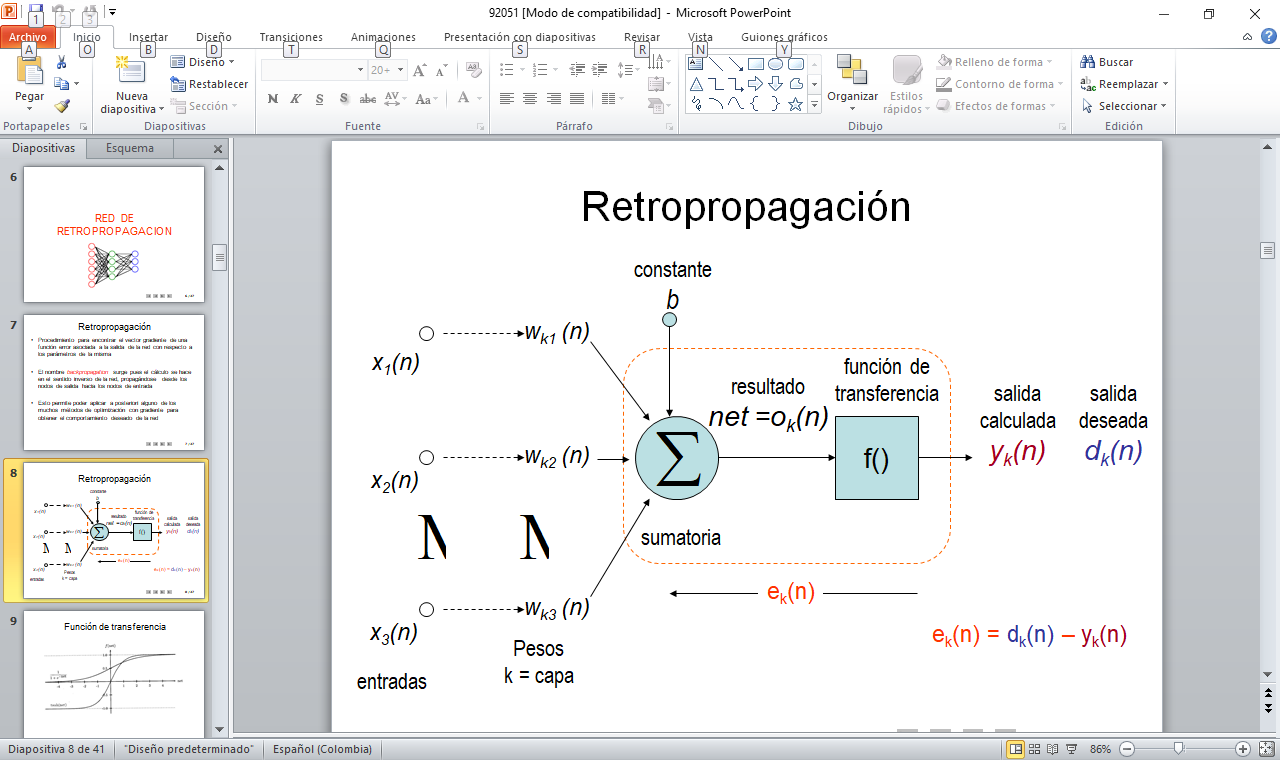
5. Repetir los pasos 2 a 4 hasta que las salidas reales y las deseadas sean iguales para todos los vectores del conjunto de entrenamiento

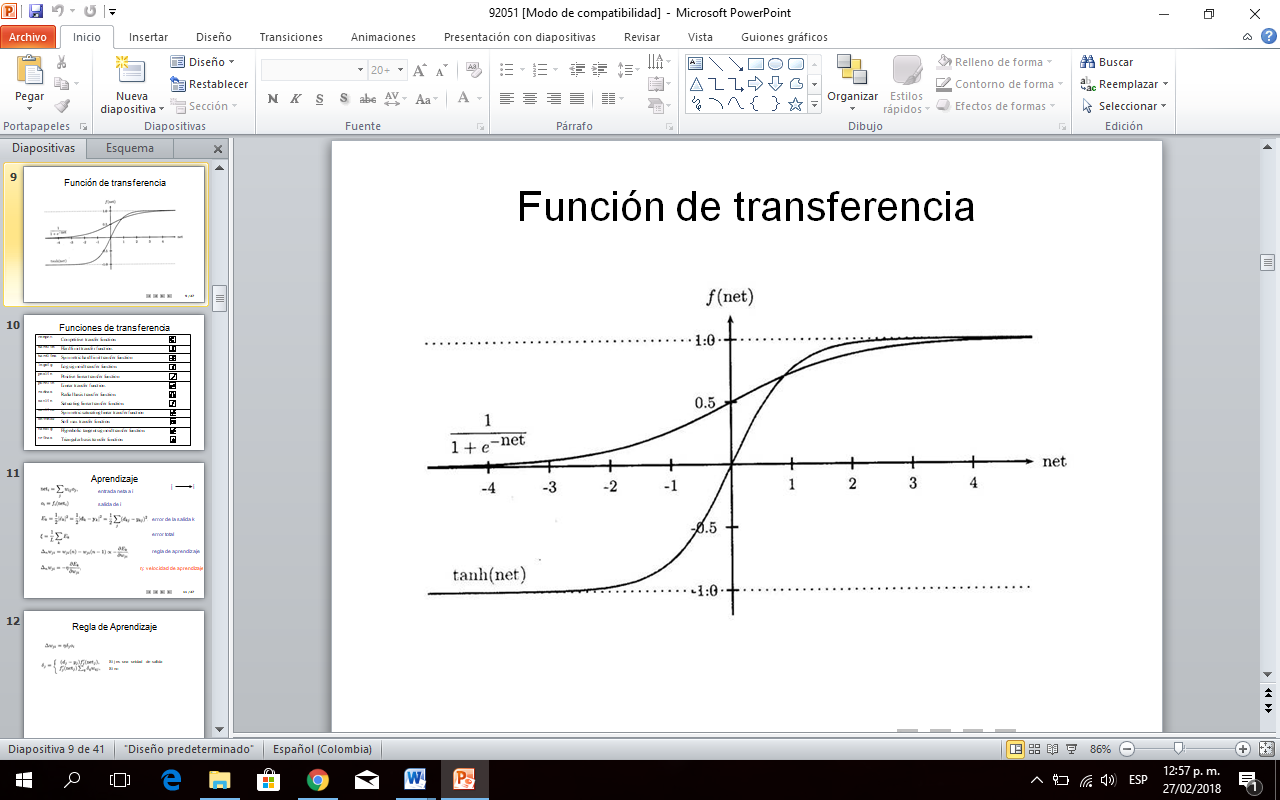
Siguiendo este método se garantiza que, para un conjunto de entrenamiento adecuado, después de un número finito de iteraciones el error se reduce a niveles aceptables. El número de iteraciones necesarias y el nivel de error deseado depende de cada problema particular.

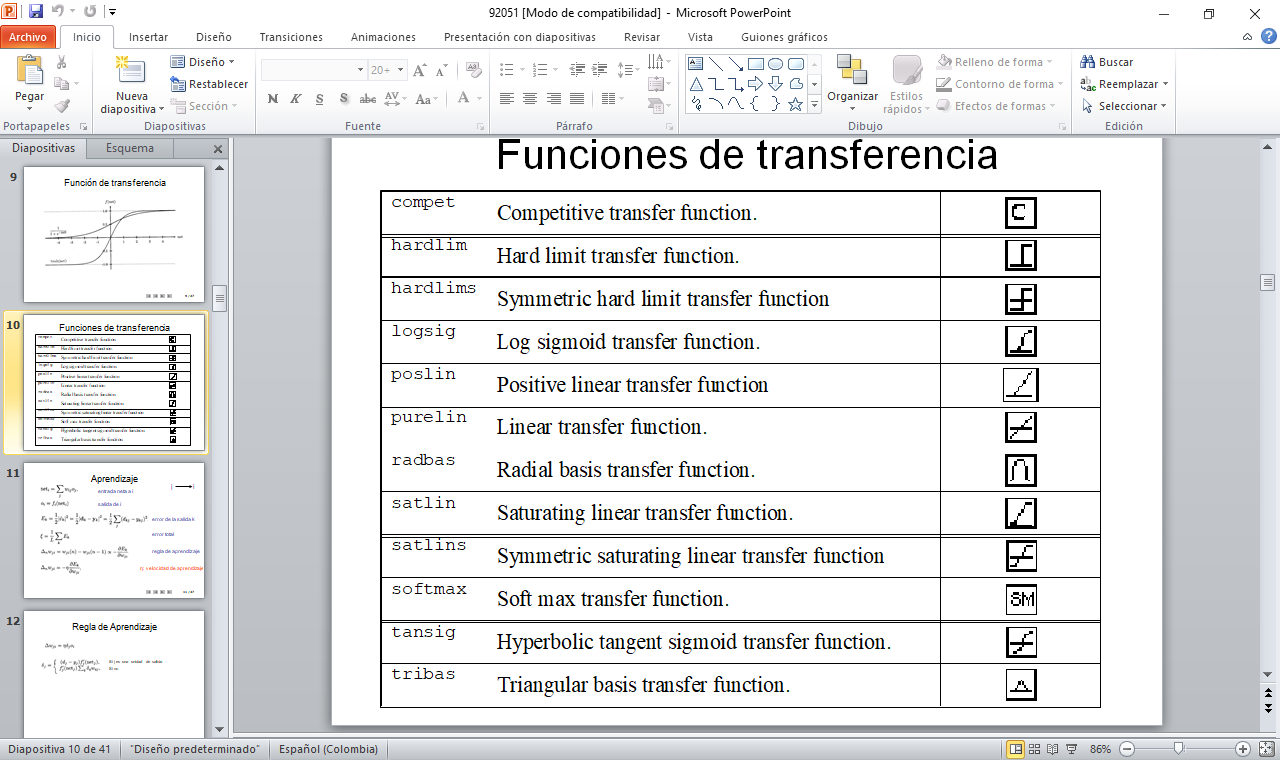
7. La Red Backpropagation En 1986, Rumelhart, Hinton y Williams, formalizaron un método para que una red neuronal aprendiera la asociación que existe entre los patrones de entrada y las clases correspondientes, utilizando varios niveles de neuronas. El método backpropagation (propagación del error hacia atrás), basado en la generalización de la regla delta, a pesar de sus limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las redes neuronales. El funcionamiento de la red backpropagation (BPN) consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de ji i Netj = ∑yi w 6 manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada[1][3][4].

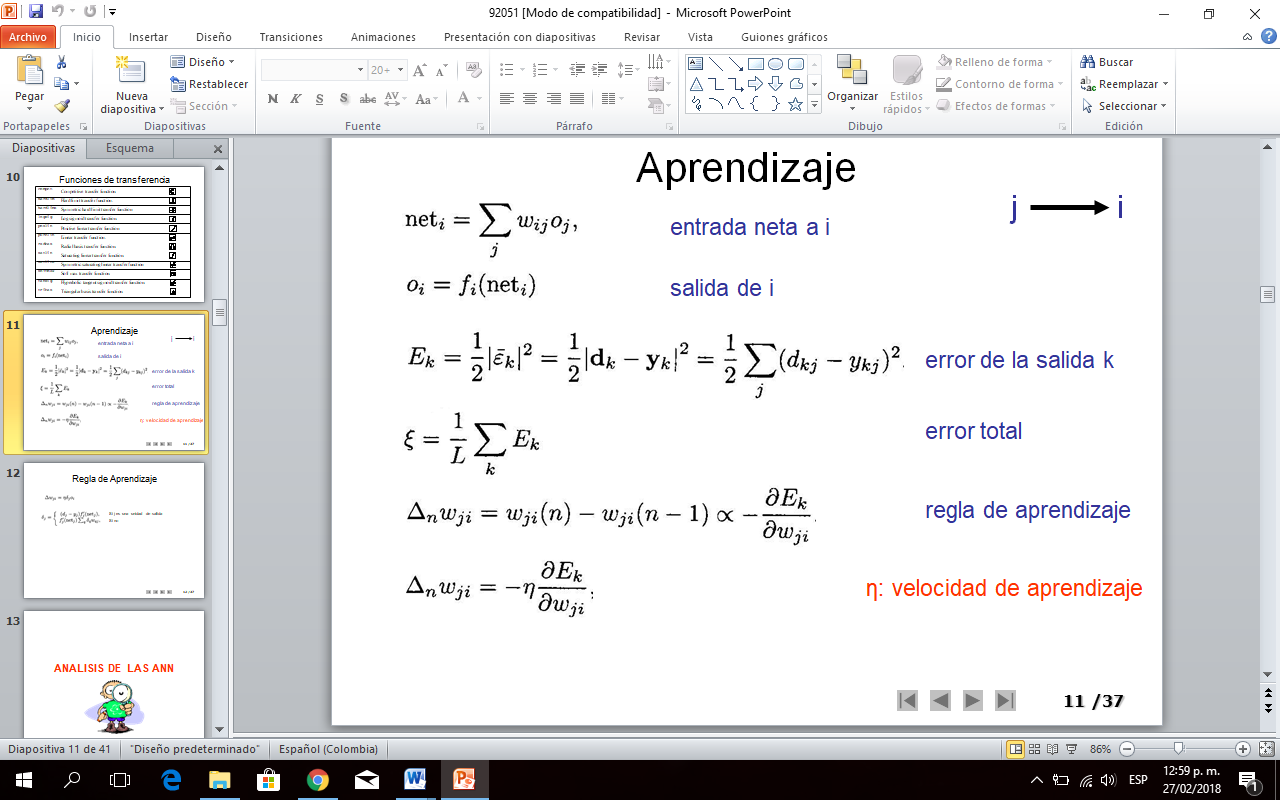
La importancia de la red backpropagation consiste en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe ente un conjunto de patrones de entrada y sus salidas correspondientes. Es importante la capacidad de generalización, facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas. Estructura y Aprendizaje de la Red Backpropagation En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás feedback ni laterales entre las neuronas de la misma capa. La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose éstos valores con la salida esperada para así obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropopagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primera capa. De esta manera se han modificado los pesos de las conexiones de la red para cada patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón. La técnica Backpropagation requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal.

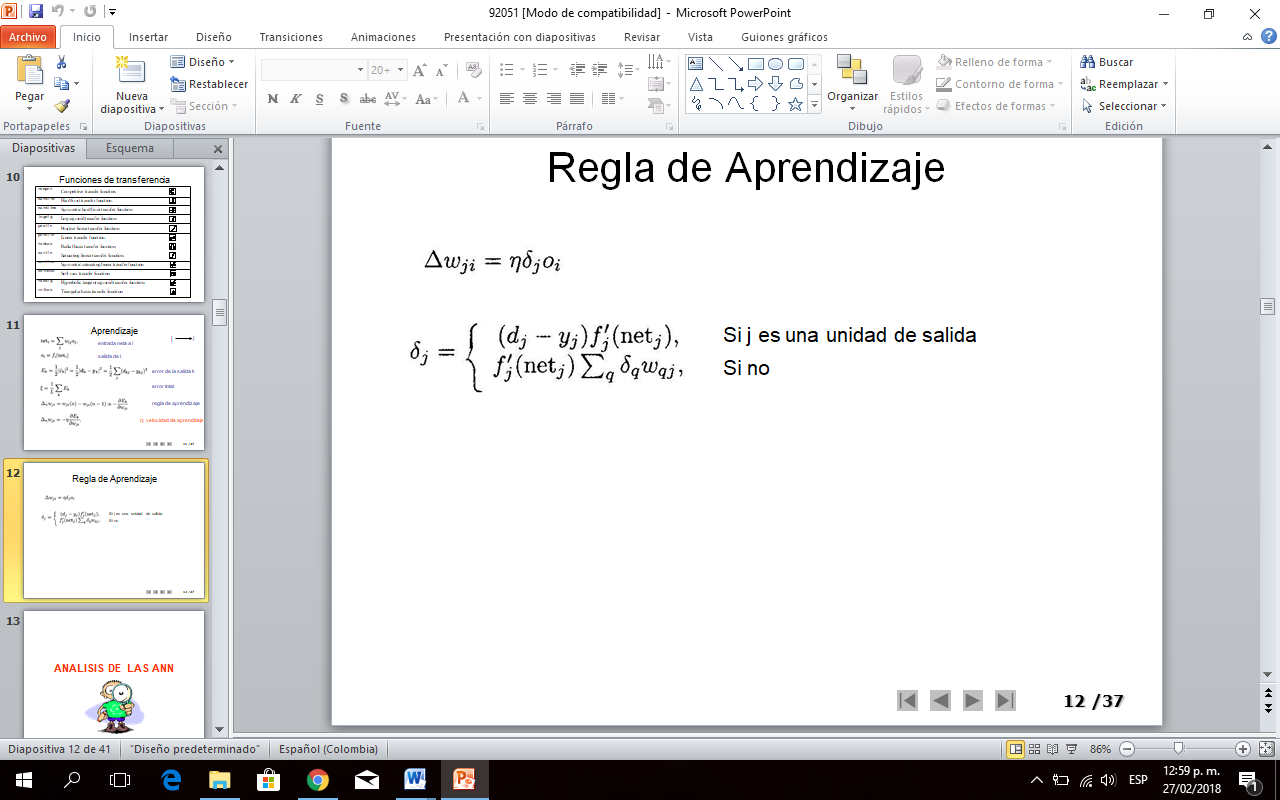
8.









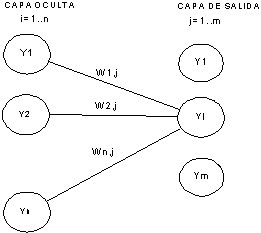


**Descripción matemática del algoritmo de retropropagación**

Se explica una versión del algoritmo ([Hinton, 1992](http://magomar.webs.upv.es/rna/tutorial/RNA_bibliografia.html" \l "8)) para redes con las siguientes características:

* No recurrentes
* Función de activación [sigmoidal](http://magomar.webs.upv.es/rna/tutorial/RNA_intro.html" \l "Funci%C3%B3n sigmoidal o log%C3%ADstica)
* Capas totalmente interconectadas
* Operación totalmente síncrona

1. Aleatorizamos los pesos de las conexiones.
2. Presentamos un patrón de entrada y calculamos la salida.
3. Dada una unidad j-ésima de la capa de salida y unidades i-ésimas de la capa oculta inmediatamente anterior, calculamos la entrada total ponderada y la salida o activación de la misma.



http://magomar.webs.upv.es/rna/images_tutorial/Image25.gif

http://magomar.webs.upv.es/rna/images_tutorial/Image26.gif

1. Una vez computadas las actividades de todas las unidades de salida se calcula una estimación del error, generalmente una función cuadrática de los errores individuales cometidos por cada unidad, siendo cada error individual la diferencia entre la salida deseada y la obtenida.

http://magomar.webs.upv.es/rna/images_tutorial/Image27.gif siendo dj la salida deseada para la unidad j-ésima

Nota: Se van a indicar por un lado las expresiones matemáticas y por otro lado la explicación intuitiva de cada paso. Conviene recordar que nuestro objetivo es calcular como varía el error al variar el peso de cada conexión (tasa de variación del error respecto al peso de una conexión, EP)

1. Cómputo de la rapidez de variación del error al cambiar la actividad de cada unidad de salida (EA, error respecto a la actividad)

http://magomar.webs.upv.es/rna/images_tutorial/Image28.gif

Es justamente la diferencia entre la salida deseada y la salida real obtenida, es decir, la diferencia entre la actividad deseada y la actividad real

1. Cómputo de la rapidez de variación del error al cambiar la entrada total que recibe cada unidad de salida.

http://magomar.webs.upv.es/rna/images_tutorial/Image29.gif

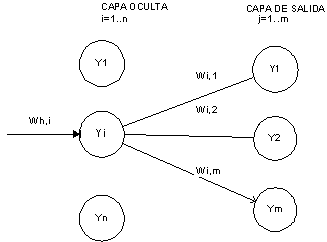
Es igual a la tasa de variación del error al variar su activación multiplicado por la tasa de variación de la activación al cambiar su entrada ( que es justamernte la derivada de la función sigmoidal )

1. Cómputo de la rapidez de variación del error al ser modificado un peso de la conexión aferente a una unidad de salida.

http://magomar.webs.upv.es/rna/images_tutorial/Image30.gif

Es igual a la tasa de variación del error al variar su entrada, por la tasa de variación de la entrada al variar ese peso.

Hasta ahora sabemos calcular el EA sólo para las unidades de salida, ¿ qué pasa con las unidades ocultas?. En este caso no tenemos una estimación directa del error aportado por cada unidad oculta; aquí es donde interviene la retropropagación o propagación hacia atrás del error:

La unidad i-ésima de la capa oculta afecta a todas las unidades de salida, por lo tanto, para estimar como varía el error al variar la actividad de esa unidad oculta, habrá que sumar los efectos individuales de su actividad sobre todas las neuronas de salida. Cada efecto individual sobre la variación del error, será igual a la tasa de variación del error de la unidad de salida al cambiar su entrada total, multiplicado por la tasa de variación de su entrada al variar la actividad de la unidad oculta.

http://magomar.webs.upv.es/rna/images_tutorial/Image33.gif

1. Conociendo EA para las unidades de cualquier capa podemos calcular d y EP con las expresiones ya conocidas.

http://magomar.webs.upv.es/rna/images_tutorial/Image34.gif

http://magomar.webs.upv.es/rna/images_tutorial/Image35.gif

1. Disponiendo de la tasa de variación del error respecto al peso de una conexión (EP), podemos usar distintas reglas para modificar ese peso en aras a reducir dicho de error. Una de las primeras reglas que aprovechó este algoritmo es la *regla delta generalizada*, que calcula el incremento a aplicar a un peso como una proporción directa de la tasa de variación del error.   
   siendo h el coeficiente de aprendizaje, típicamente con valores comprendidos entre 0.01 y 1.0  
   http://magomar.webs.upv.es/rna/images_tutorial/Image36.gif  
   http://magomar.webs.upv.es/rna/images_tutorial/Image37.gif

**Extensiones de la regla delta generalizada**

La regla DBD (delta-bar-delta) (Jordan, 1988) consiste en usar un coeficiente de aprendizaje propio y variable para cada conexión.

http://magomar.webs.upv.es/rna/images_tutorial/Image38.gif

Una extensión propuesta por Rumelhart, Hinton y Williams (1986) consiste en añadir un término proporcional a la cantidad del último cambio realizado sobre un peso. Al coeficiente que pondera dicha cantidad se le llama momentum (??

http://magomar.webs.upv.es/rna/images_tutorial/Image39.gif

http://magomar.webs.upv.es/rna/images_tutorial/Image40.gif

La propuesta EDBD (extended delta-bar-delta) ([Minai y Williams,1990](http://magomar.webs.upv.es/rna/tutorial/RNA_bibliografia.html" \l "13)) consiste en añadir el momentum a la regla DBD.

10. Teorema de Kolmogorov

La ley cero-uno de Kolmogórov es un teorema de la teoría de las probabilidades. Llamado así en honor al matemático ruso Andréi Kolmogórov, dice que la probabilidad de cierto tipo de eventos llamados eventos de cola (tail event en inglés) es cero o uno. Los eventos de cola son aquellos eventos definidos por una sucesión infinita de eventos independientes, pero que son independientes de cualquier subconjunto finito de estos.

Por ejemplo, supongamos infinitas tiradas de una moneda. El evento: "que salga en total una cantidad finita de caras" es independiente de cualquier número finito de tiradas. Examinando una cantidad finita de tiradas no podemos concluir nada respecto a si la cantidad de caras fue finita o infinita.

Dada cualquier función continua f: [0 1] n  R m, y = f(x), f puede ser implementada exactamente por una red neuronal de tres capas sin retroalimentación que tiene una capa de entrada de n elementos que unicamente copian las entradas a la siguiente capa, (2n + 1) elementos de procesamiento en la capa intermedia y me elementos de procesamiento en la capa de salida

**BIBLIOGRAFÍA**

https://gruporedesneuronales.wordpress.com/definiciones-basicas/

https://sites.google.com/site/proyectointeligenciaartificial/indice/las-redes-neuronales/ventajas-de-las-redes-neuronales

http://redes-neuronales.wikidot.com/definicion-ventajas-desventajas

http://www.redes-neuronales.com.es/tutorial-redes-neuronales/funciones-de-las-neuronas-artificiales.htm

http://www.itnuevolaredo.edu.mx/takeyas/apuntes/Inteligencia%20Artificial/Apuntes/tareas\_alumnos/RNA/Redes%20Neuronales2.pdf

https://web.csulb.edu/~cwallis/artificialn/History.htm

http://repositoriodigital.ipn.mx/bitstream/123456789/8628/1/Archivo%20que%20incluye%20portada,%20%C3%ADndice%20y%20texto.pdf

Adaline/ADALINEYMADALINE2.doc