TALLE N°3 REDES NEURONALES ARTIFICIALES

NATALIA ISABEL HERNANDEZ NAVEROS

LEIDY YULIANA QUINTERO JARAMILLO

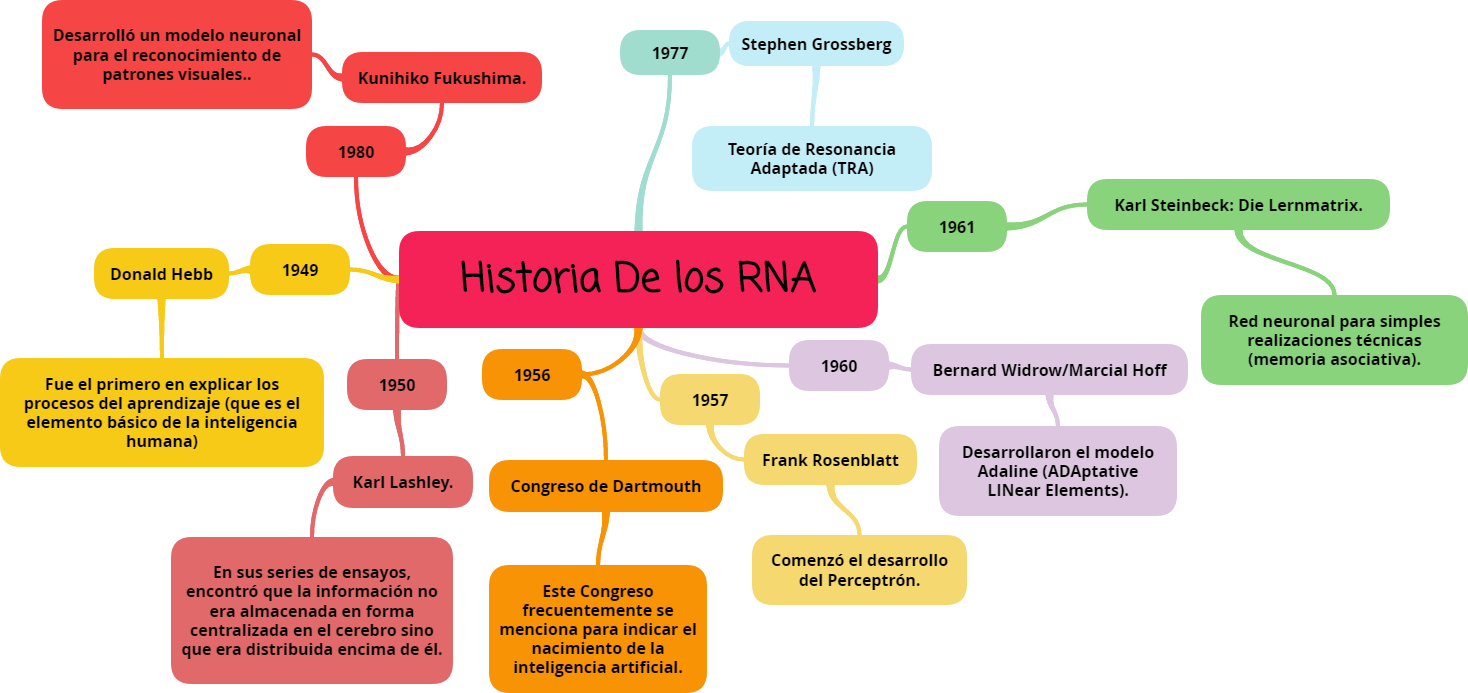
DOCENTE: CARLOS ALBERTO LONDOÑO

CORPORACION DE ESTUDIOS TECNOLOGICOS DEL NORTE DEL VALLE

TECNOLOGÍA EN SISTEMAS DE INFORMACIÓN

CARTAGO-VALLE

2018

**1 Mapa conceptual**

Fuente Propia

**2**. **Ventajas de las redes neuronales artificiales**

Las redes neuronales artificiales (RNA) tienen muchas ventajas debido a que están basadas en la estructura del sistema nervioso, principalmente el cerebro.

* **Aprendizaje**: Las RNA tienen la habilidad de aprender mediante una etapa que se llama etapa de aprendizaje. Esta consiste en proporcionar a la RNA datos como entrada a su vez que se le indica cuál es la salida (respuesta) esperada.
* **Auto organización:** Una RNA crea su propia representación de la información en su interior, descargando al usuario de esto.
* **Tolerancia a fallos:** Debido a que una RNA almacena la información de forma redundante, ésta puede seguir respondiendo de manera aceptable aun si se daña parcialmente.
* **Flexibilidad:** Una RNA puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (ej. si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente)
* **Tiempo real:** La estructura de una RNA es paralela, por lo cual si esto es implementado con computadoras o en dispositivos electrónicos especiales, se pueden obtener respuestas en tiempo real.
* **Fácil inserción dentro de la tecnología existente:** Una red individual puede ser entrenada para desarrollar una única y bien definida tareas complejas, que hagan múltiples selecciones de patrones, debido a que una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una implementación hardware de bajo coste ,es fácil insertar redes neuronales para aplicaciones específicas dentro de sistemas existente. (REDES NEURONALES, 2018)

**Desventajas de las redes neuronales**

* **Complejidad de aprendizaje para grandes tareas**, cuantas más cosas se necesiten que aprenda una red, más 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 más 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, más 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. (REDES NEURONALES, 2018)

**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.

* **Biología:**
* Aprender más acerca del cerebro
* Obtención del modelo de la retina.
* **Empresa**
* Evaluación de probabilidad de formaciones geológicas y petrolíferas
* Explotación de bases de datos
* Optimización de plazas y horarios en líneas de vuelo
* Reconocimiento de caracteres escritos
* **Medio ambiente**
* Analizar tendencias y patrones
* Previsión del tiempo
* **Finanzas**
* Previsión de la evolución de los precios
* Valoración del riesgo de los créditos
* Interpretación de firmas
* Identificación de falsificaciones
* **Manufacturación**
* Robots y sistemas de control (visión artificial y sensores de presión, temperatura, etc)
* Control de producción en líneas de proceso
* **Medicina**
* Analizadores del habla para la ayuda de audición de sordos
* Monitorización en cirugía
* Predicción de reacciones adversas a los medicamentos
* Lectores de rayos X
* Diagnóstico y tratamiento a partir de síntomas y/o datos analíticos
* **Militares**
* Clasificación de las señales de radar
* Creación de armas inteligentes
* Reconocimiento y seguimiento en el tiro al blanco (Wordpress, 2018)

**4. FUNCION DE ACTIVACION** (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 Las funciones de activación más comúnmente utilizadas se detallan a continuación:

1) Función lineal:

x

− 1/ a < x < 1/ a

X ≥ 1/ a

con x = gini - Θi, y a > 0.

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.

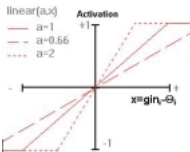


Figura 1 1funcion de activación lineal

2) Función sigmoidea:

f( x) = , con x = gini - Θi.

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.

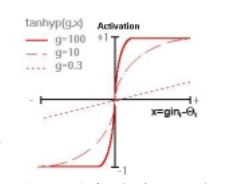


Figura 1 2 función de activación sigmoidea

3) Función tangente hiperbólica:

f(x) =, con x = gini – Θi

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

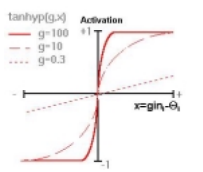


Figura 1 3funcion de activación tangente hiperbólica (Matich, 2018)

**5. PERCEPTRON**

**A. Historia:**

La primera red neuronal conocida, fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; esta consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicadas por los pesos es mayor o igual que el patrón preestablecido la salida de la red es uno (1), en caso contrario la salida es cero (0). Al inicio del desarrollo de los sistemas de inteligencia artificial, se encontró gran similitud entre su comportamiento y el de los sistemas biológicos y en principio se creyó que este modelo podía computar cualquier función aritmética o lógica.

El primer modelo de Perceptrón fue desarrollado en un ambiente biológico imitando el funcionamiento del ojo humano, el fotoperceptrón como se le llamo era un dispositivo que respondía a señales ópticas; figura Perceptrón 1.1 la luz incide en los puntos sensibles (S) de la estructura de la retina, cada punto S responde en forma todo-nada a la luz entrante, los impulsos generados por los puntos S se transmiten a las unidades de asociación (A) de la capa de asociación; cada unidad A está conectada a un conjunto aleatorio de puntos S, denominados conjunto fuente de la unidad A, y las conexiones pueden ser tanto excitatorias como inhibitorias. Las conexiones tienen los valores posibles +1, -1 y 0, cuando aparece un conjunto de estímulos en la retina, una unidad A se activa si la suma de sus entradas sobrepasa algún valor umbral; si la unidad esta activada, A produce una salida que se envía a la siguiente capa de unidades. (Morales, 2018)

**B. Fórmula matemática:**

http://www.tecnohobby.net/ppal/images/stories/news/IA/RP/perceptron_net.png

Este net es la suma de las multiplicaciones del valor de salida de los nodos de entrada por su correspondiente peso. Por ejemplo, si tuvieramos un patrón de dos características más el valor de salida del bias que es 1: p1 = [2,3,1] y tuvieramos un vector de pesos w=[0.2,0.3,0.4], el net sería

net = (2\*0.2)+(3\*0.3)+(1\*0.4)

net = 0.4 + 0.9 + 0.4

net = 1.7 (Soberon, 2018)

C. **Estructura del perceptron**

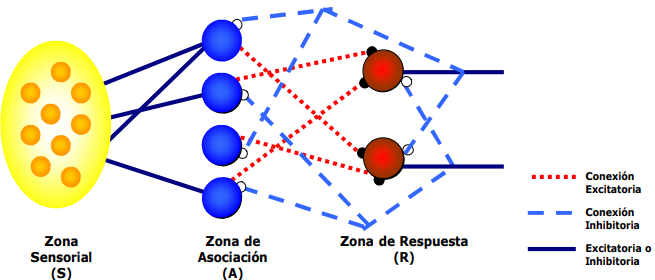


Ilustración 1 Estructura del Perceptron

De forma similar, las unidades A están conectadas a unidades de respuesta (R) dentro de la capa de respuesta y la conectividad vuelve a ser aleatorio entre capas, pero se añaden conexiones inhibitorias de realimentación procedentes de la capa de respuesta y que llegan a la capa de asociación, también hay conexiones inhibitorias entre las unidades R. Todo el esquema de conexiones se describe en forma general en un diagrama de Venn, para un Perceptrón sencillo con dos unidades de respuesta.

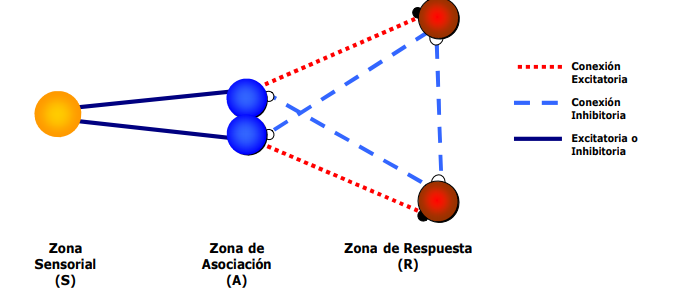


Ilustración 2 Estructura del Perceptron

El Perceptrón era inicialmente un dispositivo de aprendizaje, en su configuración inicial no estaba en capacidad de distinguir patrones de entrada muy complejos, sin embargo, mediante un proceso de aprendizaje era capaz de adquirir esta capacidad. En esencia, el entrenamiento implicaba un proceso de refuerzo mediante el cual la salida de las unidades A se incrementaba o se decrementaba dependiendo de si las unidades A contribuían o no a las respuestas correctas del Perceptrón para una entrada dada. Se aplicaba una entrada a la retina, y el estímulo se propagaba a través de las capas hasta que se activase una unidad de respuesta. Si se había activado la unidad de respuesta correcta, se incrementaba la salida de las unidades A que hubieran contribuido. Si se activaba una unidad R incorrecta, se hacía disminuir la salida de las unidades A que hubiesen contribuido. Mediante estas investigaciones se pudo demostrar que el Perceptrón era capaz de clasificar patrones correctamente, en lo que Rosenblatt denominaba un entorno diferenciado, en el cual cada clase estaba formada por patrones similares.

El Perceptrón también era capaz de responder de manera congruente frente a patrones aleatorios, pero su precisión iba disminuyendo a medida que aumentaba el número de patrones que intentaba aprender.

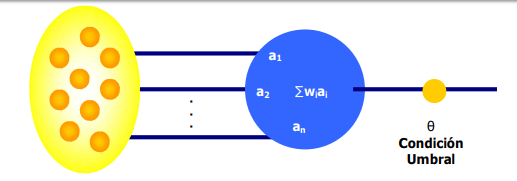
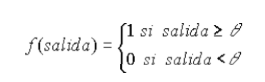


Ilustración 3 Estructura del Perceptron

La estructura de un Perceptrón sencillo es similar a la del elemento general de procesamiento que se muestra en la figura, en la que se observa la adición de una condición umbral en la salida. Si la entrada neta, a esta condición es mayor que el valor umbral, la salida de la red es 1, en caso contrario es 0. La función de salida de la red es llamada función umbral o función de transferencia



A pesar de esta limitación, el Perceptrón es aún hoy una red de gran importancia, pues con base en su estructura se han desarrollado otros modelos de red neuronal como la red Adaline y las redes multicapa. (Redes De una capa, 2018)

**E. función de activación**

**Función escalón:**

La función que representa el comportamiento ideal de una neurona es la función escalón: dependiendo de la entrada se activa la conexión entre neuronas (salida = 1) o no (salida = 0). Esta decisión puede depender de un cierto umbral (que la salida sea capaz de superarlo), representado en las expresiones anteriores por que introducir este término negativo equivale a considerar que la función vale uno a partir de q y no a partir de cero como se representa en la tabla.

Aunque éste sea el comportamiento ideal, en la práctica se suele utilizar la función sigmoidea. Esto se debe a que es una función muy parecida al escalón y derivable en el entorno de cualquier punto del eje x. Como se verá, esto es necesario en el algoritmo de retropropagación ya que está basado en la minimización del error con respecto al valor de los pesos, por lo que hay que derivar e igualar a cero.

El algoritmo de aprendizaje es de tipo supervisado. En el proceso de entrenamiento, el Perceptron se expone a un conjunto de patrones de entrada, y los pesos de la red son ajustados de forma que al final del entrenamiento se obtenga las salidas esperadas para cada uno de esos patrones de entrada.

**F. como se entrena un perceptron**

Debe ser entrenada con un conjunto de patrones previamente clasificados de manera que, si los clasifica incorrectamente, se pueda corregir el error mediante una regla de aprendizaje. En general, para entrenar la red, necesitaremos un conjunto de patrones previamente clasificados llamado muestra de entrenamiento y un conjunto de patrones llamado muestra de control

El entrenamiento se da en iteraciones; una iteración comienza cuando se le presenta el primer patrón de la muestra de entrenamiento y finaliza cuando se le muestra el último patrón de la muestra de entrenamiento. En cada iteración, se presentan uno por uno los patrones de la muestra de entrenamiento a la red y, si lo clasifica bien, se prosigue con el siguiente patrón pero si lo clasifica mal, se corrigen los pesos mediante la regla de aprendizaje y se sigue con el siguiente patrón utilizando los nuevos pesos.

El entrenamiento concluye cuando pasa una iteración en la que clasificó correctamente todos los patrones o bien, se alcanza un límite de iteraciones previamente definido por nosotros. Los pesos iniciales son aleatorios y se recomienda que se encuentren en un rango de -0.5 a 0.5, que no se repitan y que ninguno sea 0. (Soberon, 2018)

**Algoritmo del perceptron DISCRETO**

**VARIABLES**

I (nº de iteraciones)

K (contador de ejemplos)

W[i:1..n+1] (pesos del perceptron)

X[i:1..n] (valores de la entrada en el ejemplo actual)

Y (valor de la salida en el ejemplo actual)

SP (salida del perceptrón con la entrada del ejemplo actual)

Error (error en el ejemplo actual)

T (Variable para indicar si el vector es correcto)

PASO 1: Inicialización de variables

PASO 2: Bucle de iteraciones (hasta condición de parada)

Paso 2.1: Bucle de paso por todos los ejemplos

2.1.0 Leer valores del ejemplo

2.1.1 Calcular error en ese ejemplo

2.1.2 Actualizar los pesos según el error de ese ejemplo

2.1.2.1 Actualizar los pesos de las entradas

2.1.2.2 Actualizar el bias (= -umbral)

2.1.3 Incrementar contador de ejemplos

Paso 2.2.: Ver si el vector de pesos es correcto

Paso 2.3.: Incrementar el contador de iteraciones

PASO 3: Salida

PASO 1: Inicialización de variables

I=0;

Para i=1..n+1

W[i]=<nº aleatorio (normalmente entre –1 y 1)> (muy importante que el nº sea aleatorio,

distinto en cada ejecución)

T=false

PASO 2: Bucle de iteraciones

Mientras (I < NºMaxIteraciones) y (no T)

Paso 2.1: Bucle de paso por todos los ejemplos

Abrir (<fichero de ejemplos>)

K=0

Mientras no esté en el final del fichero

2.1.0 Leer valores del ejemplo

Leer(x[i] (i=1..n), y)

2.1.1 Calcular error en ese ejemplo

SP = W[n+1] + ∑ x[i] \* W[i] (salida del perceptrón con la entrada del ejemplo h)

Si SP < 0, entonces SP= 0, en otro caso SP=1

Error = y - SP (diferencia entre el valor de y en el ejemplo h y SP)

2.1.2 Actualizar los pesos según el error de ese ejemplo

2.1.2.1 Actualizar los pesos de las entradas

Para cada i=1..n

W[i] = W[i] + x[i]\*Error

2.1.2.2 Actualizar el bias (= -umbral)

W[n+1]=W[n+1] + Error

2.1.3 Incrementar contador de ejemplos

k=k+1

Cerrar (<fichero de ejemplos>)

Paso 2.2.: Ver si el vector de pesos es correcto

T= True

Abrir (<fichero de ejemplos>)

Mientras (no esté en el final del fichero y T)

Leer(x[i] (i=1..n), y)

SP = W[n+1] + ∑ x[i] \* W[i]

Si SP < 0, entonces SP= 0, en otro caso SP=1

Error = y – SP

Si Error ≠ 0, T = False

Cerrar (<fichero de ejemplos>)

Paso 2.3.: Incrementar el contador de iteraciones

I=I+

PASO 3: Salida

Escribe ( “El perceptrón aprendido es el de pesos:” )

Para i=1..n

Escribe (“ W”,i,”=”,W[i])

Escribe(“Con bias =”, W[n+1]) (Aprendizaje, 2018)

**G. ejemplos del uso del perceptron**

* El perceptrón multicapa puede ser utilizado en diversas aplicaciones en donde se necesite procesar información compleja, entre estas se puede mencionar codificación de datos, reconocimiento del habla, reconocimiento de imágenes visuales, control de robots, reconocimiento de caracteres, pronóstico de eventos y control automático de procesos.
* codificación de datos puede utilizarse un perceptrón multicapa con un número de entradas igual o menor al número de salidas. Para codificar información digital debe utilizarse una tabla con los diferentes valores de entrada con sus respectivos códigos de salida; los datos de la tabla deben utilizarse para el entrenamiento de la red, así cuando a la red se le presente una entrada, se obtendrá el correspondiente código salida.
* Una aplicación muy común del perceptrón multicapa es el reconocimiento de imágenes visuales. En esta aplicación, los píxeles que forman la imagen son usados como entradas a la red neuronal y los valores de salida deben corresponder con la clase a la que pertenece esta imagen.
* Otra aplicación del perceptrón multicapa es el control automático de un proceso industrial; en este caso, las salidas de los sensores son conectadas a la entrada de la red neuronal y las salidas de la red son conectadas a los actuadores.

**6. ADALINE**

1. **Historia.**

Al mismo tiempo que Frank Rosenblatt trabajaba en el modelo del Perceptrón Bernard Widrow y su estudiante Marcian Hoff introdujeron el modelo de la red Adaline y su regla de aprendizaje llamada algoritmo LMS (Least Mean Square). El término Adaline es una sigla, sin embargo, su significado cambió ligeramente a finales de los años sesenta cuando decayó el estudio de las redes neuronales, inicialmente se llamaba Adaptive Linear Neuron (Neurona Lineal Adaptiva), para pasar después a ser Adaptive Linear Element (Elemento Lineal Adoptivo), este cambio se debió a que la Adaline es un dispositivo que consta de un único elemento de procesamiento, como tal no es técnicamente una red neuronal. (Adaline)

1. **Fórmula matemática, explique sus términos.**

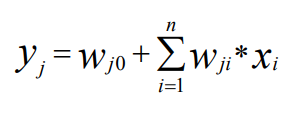


Ilustración 4. Formula ADALINE

* Estructura prácticamente idéntica al perceptron, pero es un mecanismo físico capaz de realizar aprendizaje.
* Elemento combinador adaptativo lineal, que recibe todas las entradas, las suma ponderadamente, y produce una salida.
* La diferencia con el perceptron es la manera de utilizar la salida en la regla de aprendizaje. (ADALINE)

1. **Dibuje la estructura.**

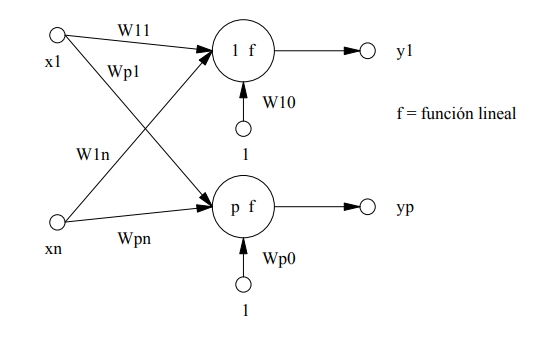
****

Ilustración 5. Estructura ADALINE

1. **¿Para qué se usa?**

Se utiliza para entrenar un Elemento Simple de Procesado, con una función de transferencia lineal**.** (Características)

Se utiliza el método de gradiente decreciente para saber en qué dirección se encuentra el mínimo global de dicha superficie.

Las modificaciones que se realizan a los pesos son proporcionales al gradiente decreciente de la función de error, por lo que cada nuevo punto calculado está más próximo al punto mínimo.

1. **¿Cuál es su función de activación?**

Consiste en hallar el vector de pesos W deseado, único, que deberá asociar cada vector de entrada con su correspondiente valor de salida correcto o deseado. La regla minimiza el error cuadrático medio definido como:

Donde es la función de error

Esta función de error está definida en el espacio de pesos multidimensional para un conjunto de entradas, y la regla de Widrow-Hoff busca el punto de este espacio donde se encuentra el mínimo global.

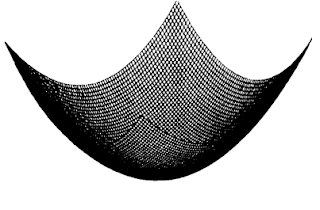


Ilustración 6. Adaline

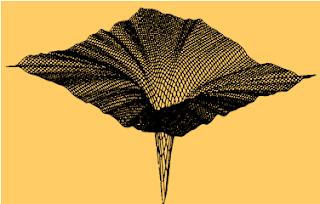


Ilustración 7. Adaline 2

Con función de activación lineal Con función de activación sigmoidal

Se utiliza el método de gradiente decreciente para saber en qué dirección se encuentra el mínimo global de dicha superficie. Las modificaciones que se realizan a los pesos son proporcionales al gradiente decreciente de la función de error, por lo que cada nuevo punto calculado está más próximo al punto mínimo.

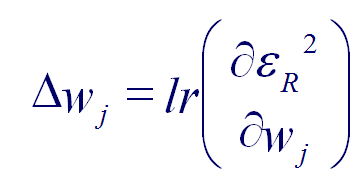


Ilustración 8. Formula

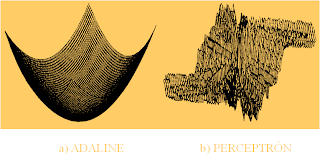


Ilustración 9. Adaline Y Perceptron

La regla de Widrow-Hoff es implementada realizando cambios a los pesos en la dirección opuesta en la que el error está incrementando y absorbiendo la constante -2 en lr.

En forma de matriz:

Transformando a la expresión del bias (considerando que el bias son pesos con entradas de 1):

(Chavez, 2010)

1. **¿Cómo se entrena una red ADALINE indique el nombre del algoritmo y sus pasos?**

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

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 (Villalobos, y otros, 2002)

1. **Nombre 5 ejemplos donde se evidencie en uso de ADALINE.**

* Se utiliza en el procesamiento de señales.

Filtros que eliminen el ruido, en señales portadoras de información

Filtros de ecualización adaptativos en módems de alta velocidad

Cancelación del ruido materno de grabaciones ECG de latidos del feto humano

Eliminación de ecos en circuitos informáticos

Canceladores adaptativos para el eco de filtrado de señales en comunicaciones telefónicas de larga distancia y comunicaciones vía satélite (Aplicaiones de ADALINE, 2002)

**7. ¿Qué es y para que se usa las redes de retropropagación?**

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 la red hayan recibido, se reajusta os pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada.

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. (Valencia, Yañez, & Sanchez, 2006)

**8. Explique el algoritmo de backpropagation, usando imágenes.**

Paso 1. Inicializar los pesos de la red con valores pequeños aleatorios.

Paso 2. Presentar un patrón de entrada y especificar la salida deseada que debe generar la red.

Paso 3. Calcular la salida actual de la red. Para ello se presentan las entradas a la red y se calcula la salida de cada capa hasta llegar a la capa de salida, ésta será la salida de la red. Los pasos son los siguientes:

Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

Para una neurona j oculta:

en donde el índice h se refiere a magnitudes de la capa oculta; el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

Se calculan las salidas de las neuronas ocultas:

Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida:

Paso 4. Calcular los términos de error para todas las neuronas.

Si la neurona k es una neurona de la capa de salida, el valor de la delta es:

La función f debe ser derivable. En general disponemos de dos formas de función de salida:

La función lineal:

La función sigmoidal:

La selección de la función depende de la forma que se decida representar la salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, en otros casos, la lineal.

Para una función lineal, tenemos: , mientras que la derivada de una función sigmoidal es:

por lo que los términos de error para las neuronas de salida quedan:

para la salida lineal.

para la salida sigmoidal.

Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente, por tanto, se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es: donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término propagación hacia atrás.

Paso 5. Actualización de los pesos: para ello utilizamos un algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma:

Para los pesos de las neuronas de la capa de salida:

Para los pesos de las neuronas de la capa oculta:

En ambos casos, para acelerar el proceso de aprendizaje se puede añadir un término momento.

Paso 6. El proceso se repite hasta que el término de error resulta aceptablemente pequeño para cada uno de los patrones aprendidos. (Valencia, Yañez, & Sanchez, 2006)

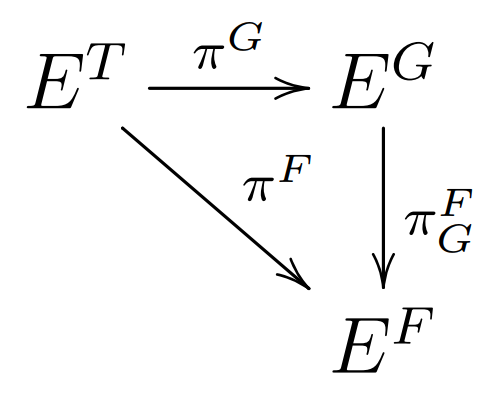
**10. ¿Qué es el teorema de Kolmogorov, explicar?**

Sea un espacio medible, T un conjunto, se define , y es la menor álgebra que hace medible a todas la proyecciones , donde es la proyección en la t-ésima coordenada. Además, si T es finito, entonces es la álgebra producto.

Sea subconjunto finito, consideremos las siguientes proyecciones sobre F dadas por se le conoce como la proyección finito dimensional.

Sea subconjuntos finitos, entonces se define las proyecciones dadas por

Se puede observar lo siguiente



Sea (Ω, F, P) un espacio de probabilidad, un espacio de medible, sea X : Ω → E una variable aleatoria, entonces la distribución de X con respecto a P es PX = P ◦ X−1.

Sea un espacio de medible y T un conjunto. Supongamos que para todo F ⊂ T, finito PT: F → [0, 1] es medida de probabilidad. Decimos que {PF: F ⊂ T, F finito} es un sistema proyectivo si F ⊂ G ⊂ T, F, G finitos, entonces

Sea {Xt}, Xt : Ω → E un proceso estocástico. Observemos que podemos poner a {Xt}t∈X como función X : ω → ET por medio de X(ω)(t) = Xt(ω). Para cada ω ∈ Ω, la función X(ω) se conoce como la ω-trayectoria. Se puede verificar que X es F-ET-medible. (Acevedo, 2011)

**Bibliografía**

Acevedo, N. (Junio de 2011). *Teorema de consistencia de Kolmogorov.* Obtenido de Teorema de Consistencia de Kolmogorov y algunas aplicaciones: http://mat.izt.uam.mx/mat/documentos/produccion\_academica/toda\_la\_produccion/Tesis%20dirigidas-17-11.pdf

*Adaline.* (s.f.). Obtenido de Redes Neuronales: http://www.konradlorenz.edu.co/images/stories/suma\_digital\_sistemas/2009\_01/APLICACI%D3N%20DE%20REDES%20NEURONALES%20MLP/Cap%202.%20(parte1).pdf

*ADALINE.* (s.f.). Obtenido de Perceptron y Adaline: http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema2-PerceptronAdaline.pdf

*Aplicaiones de ADALINE.* (2002). Obtenido de SlideShare: https://www.slideshare.net/mentelibre/redes-neuronales-adaline

*Aprendizaje.* (25 de 02 de 2018). Obtenido de Percetron: file:///C:/Users/Leidy/Downloads/Aprendizaje%20Perceptron%20(1).pdf

*Características.* (s.f.). Obtenido de ADALINE: http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema3-0.pdf

Chavez, R. (21 de Noviembre de 2010). *Regla de Widrow-Hoff*. Obtenido de RED ADELINE: http://ronaldchavezblog.blogspot.com.co/2010/11/red-adeline.html

Matich, D. J. (25 de 02 de 2018). *Redes Neuronales.* Obtenido de Conceptos Basicos y Aplicaciones: https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5\_anio/orientadora1/monograias/matich-redesneuronales.pdf

Morales, E. G. (25 de 02 de 2018). *Redes Neuronales Artificiales.* Obtenido de Rincon del vago: https://html.rincondelvago.com/redes-neuronales-artificiales\_1.html

*Redes De una capa.* (25 de 02 de 2018). Obtenido de El Perceptron-Decsai: ftp://decsai.ugr.es/pub/usuarios/castro/Actividades/Redes-Neuronales/Apuntes/Apuntes%20Javier%20Rodriguez%20Blazquez/Redes%20de%20una%20capa.pdf

*REDES NEURONALES.* (25 de 02 de 2018). Obtenido de DEFINICION,VENTAJAS Y DESVENTAJAS: http://redes-neuronales.wikidot.com/definicion-ventajas-desventajas

Soberon, L. F. (25 de 02 de 2018). *Percepton.* Obtenido de Tecno Hobby net: http://www.tecnohobby.net/ppal/index.php/inteligencia-artificial/redes-neuronales/11-perceptron

Valencia, M., Yañez, C., & Sanchez, L. (2006). *La Red Backpropagation.* Obtenido de INSTITUTO POLITÉCNICO NACIONAL CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN: http://repositoriodigital.ipn.mx/bitstream/123456789/8628/1/Archivo%20que%20incluye%20portada,%20índice%20y%20texto.pdf

Valencia, M., Yañez, C., & Sanchez, L. (2006). *Pasos para aplicar el Algoritmo de Entrenamiento.* Obtenido de INSTITUTO POLITÉCNICO NACIONAL CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN: http://repositoriodigital.ipn.mx/bitstream/123456789/8628/1/Archivo%20que%20incluye%20portada,%20índice%20y%20texto.pdf

Villalobos, I., Murillo, S., Alvarez, J., Corona, A., Martinez, E., & Juarez, U. (29 de Agosto de 2002). *Adaline y Madaline.* Obtenido de Redes Neuronales: cursos.itam.mx/akuri/2002/S22002/RNS/.../Adaline/ADALINEYMADALINE2.doc

*Wordpress.* (25 de 02 de 2018). Obtenido de Aplicaciones De Las Redes Neuronales: https://gruporedesneuronales.wordpress.com/definiciones-basicas/