REDES NEURONALES ARTIFICIALES

NATALIA ISAZA SERRANO

ANDRES FELIPE FARFAN HERNANDEZ

TRABAJO DE:

INTELIGENCIA ARTIFICIAL (IA)

PRESENTADO A:

CARLOS ALBERTO LONDOÑO LOAIZA

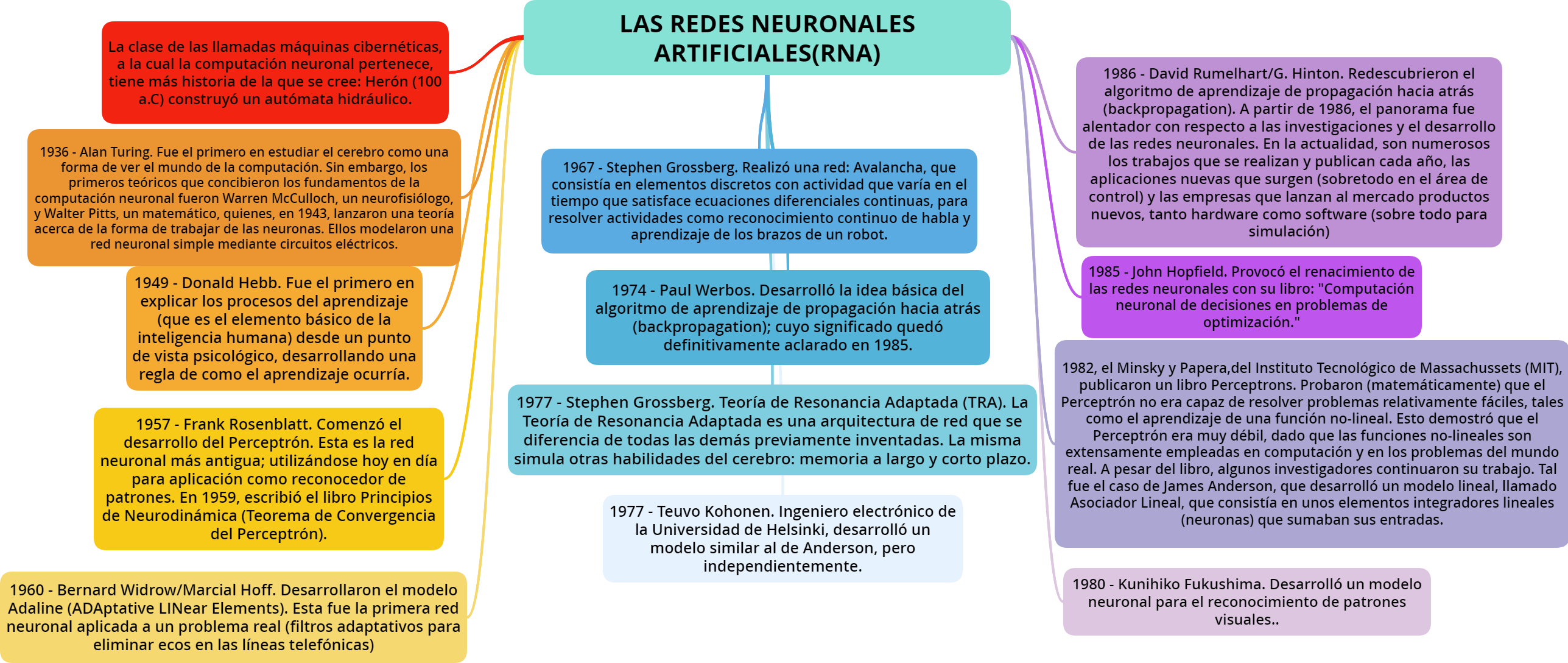
CARTAGO VALLE

CORPORACIÓN DE ESTUDIOS TECNOLÓGICOS DEL NORTE DEL VALLE

FACULTAD DE CIENCIAS INFORMATICAS

Solucion

1.



2. Ventajas:

- Las redes neuronales artificiales pueden reducir algoritmos a través de un proceso de aprendizaje

- Al utilizar las redes neuronales artificiales no es necesario conocer a fondo las fórmulas matemáticas, Solo se requiere estar familiarizado con los datos del trabajo.

- La solución de problemas no lineales es uno de los fuertes de la inteligencia artificial.

- Posee tolerancia a fallos, debido a que una red es dañada o presenta fallos, no se vera afectada en su totalidad, gracias a su redundancia de la información.

- Las redes neuronales artificiales tienen respuesta en tiempo real, debido a que trabajan en paralelo, pero pueden ser mejorado con el diseño del hardware.

- puede solucionar el ruido de los datos de entrada, ya que por su estructura le da manejo a este tipo de datos.

Desventajas:

- Las redes neuronales artificiales se deben entrenar para cada problema, consta de muchas pruebas.

- Debido a que las redes se entrenan en lugar de programarlas, estas necesitan muchos datos.

- Mucho tiempo para poder entrenar o enseñar a la red neuronal, para poder solucionar solo un problema.

- La red neuronal artificial no interpreta el dato de salida, eso depende del programador y la aplicación para que fue creada la red. Donde el programador deberá darle un significado a el dato de salida.

- Complejidad dependiendo del problema o de las actividades que la red neuronal está diseñada, si los problemas son complejos, el entrenamiento de la red va a hacer mucho más.

- No hay reglas claras a la hora de diseñar una red neuronal, son muchos factores que hay que tener en cuenta para poder implementarla, y eso que todo depende de la aplicación que se le valla a dar.

3.

Algunas areas donde se aplican RNA son:

* Automoviles: sistemas piloto automatico. D eteccion de fallas por reconocimiento externo de vibraciones.
* Bancos: lectura de cheques y otros documentos. Evaluacion de aplicaciones de creditos.
* Electronica: prediccion de secuencia de codigos. Distribucion de elementos en CI. Control de procesos.Analisis de fallas. Vision artificial. Reconocimiento de voz.
* Finanzas: tasacion real de los bienes. Asesoria de prestamos. Prevension en la evolucion de precios. Seguimientos de hipotecas. Analisis de uso de linea de creditos.Evolucion de riesgo en creditos. Identificacion de falsificaciones. Interpretacion y reconocimiento de firmas.
* Manufactura: control de la produciion y del proceso. Analisis y diseño de productos. Diagnosticos de fallas en el proceso y maquinarias. Identificaion de particulas en el tiempo real. Inspeccion de calidad mediante sistema visuales. Analisis de mantenimientos de maquinas.
* Medicina: analisis de celulas portadoras de carcer mamario. Analisis de electroencefalograma y de electrocardigrama. Reconocimiento de infartos mediante ECG. Diseño de protesis. Optimizacion en tiempos de trasplantes. Reduccion de gastos hospitalarios.
* Robotica: control dinamico de trasyectorias. Robots elevadores. Controladores. Sistemas opticos.
* Seguridad: codigos de seguridad adaptivos. Criptografia. Reconocimiento de huella digital.
* Telecomunicaciones: comprension de datos e imágenes. Automatizacion de servicios deinformacion. Traslacion en tiempo real de lenguaje hablando.
* Trasporte: diagnostico de frenos en camiones. Sistema de ruteo y seguimiento de flotas.
* Voz: reconocimiento de voz. Comprension de voz. Clasificacion de vocales. Trasformacion de texto escrito a voz.

4. Que es la función de activación?

La función de activación es el cálculo de salida de la neurona a partir del valor neto de la entrada, es un cambio de estado.

¿Cuáles son las funciones de activación que existen?

Función escalón

Función lineal o mixta

Función Tangente Hiperbólica

Función Sigmoidal

Función Identidad

Función Gaussiana

5.

**Perceptrón**

1. Historia

En 1943, Warren McCulloch y Walter Pitts introdujeron una de las primeras neuronas artificiales. La característica principal de su modelo de neurona es que un suma ponderada de las señales de entrada se compara con un umbral para determinar la neurona de salida. Cuando la suma es mayor o igual al umbral, la salida es 1. Cuando la suma es menor que el umbral, la salida es 0. A finales de 1950 Frank Rosenblatt y otros investigadores desarrollaron una clase de redes neuronales llamadas perceptrones. Las neuronas de estas redes eran similares a las de McCulloch y Pitts.

La contribución clave de Rosenblatt fue la introducción de una regla de aprendizaje para la formación de redes perceptrón para resolver problemas de reconocimiento de patrones. Demostró que su regla de aprendizaje siempre convergirá a los pesos correctos de la red, si existen pesos que solucionan el problema. El Perceptrón pudo incluso aprender cuando se inicializaba con valores aleatorios de sus pesos y bias.

El Perceptrón es limitado. Dichas limitaciones fueron publicadas en el libro Perceptrons por Marvin Minsky y Seymour Papert. Ellos demostraron que las redes perceptrón eran incapaces de implementar ciertas funciones elementales. No fue sino hasta la década de los 8O’s que estas limitaciones fueron superadas con las redes perceptrón mejoradas (multicapa) asociadas con reglas de aprendizaje.

1. Formula matemática, explique sus términos

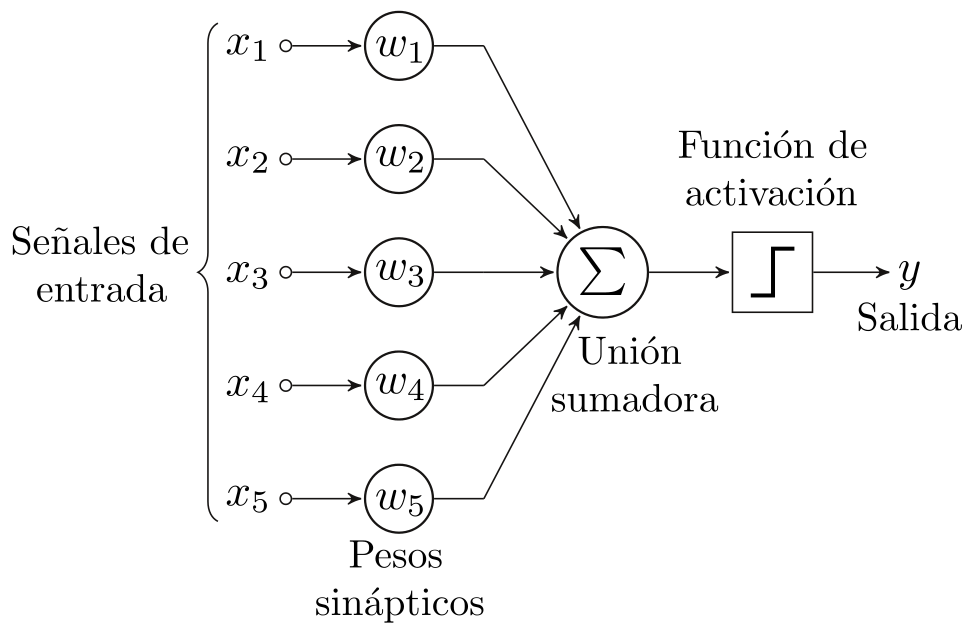
El perceptrón usa una matriz para representar las redes neuronales y es un discriminador terciario que traza su entrada ***{\displaystyle x}x***(un vector binario) a un único valor de salida ***{\displaystyle f(x)}f(x)*** (un solo valor binario) a través de dicha matriz.

https://upload.wikimedia.org/math/c/4/1/c417e0191fa26561f6947ce57c182617.png

Donde ***{\displaystyle w}w***es un vector de pesos reales y ***{\displaystyle w\cdot x}w\*x*** es el producto escalar (que computa una suma ponderada). ***{\displaystyle u}b*** es el 'umbral', el cual representa el grado de inhibición de la neurona, es un término constante que no depende del valor que tome la entrada.

El valor de ***{\displaystyle f(x)}f(x)*** (0 o 1) se usa para clasificar ***{\displaystyle x}x*** como un caso positivo o un caso negativo, en el caso de un problema de clasificación binario. El umbral puede pensarse de como compensar la función de activación, o dando un nivel bajo de actividad a la neurona del rendimiento. La suma ponderada de las entradas debe producir un valor mayor que ***{\displaystyle u}b***para cambiar la neurona de estado 0 a 1.

c. Dibuje la estructura



d. ¿Para que se usa?

En el campo de las Redes Neuronales, el perceptrón , según Frank Rosenblatt, se refiere a:

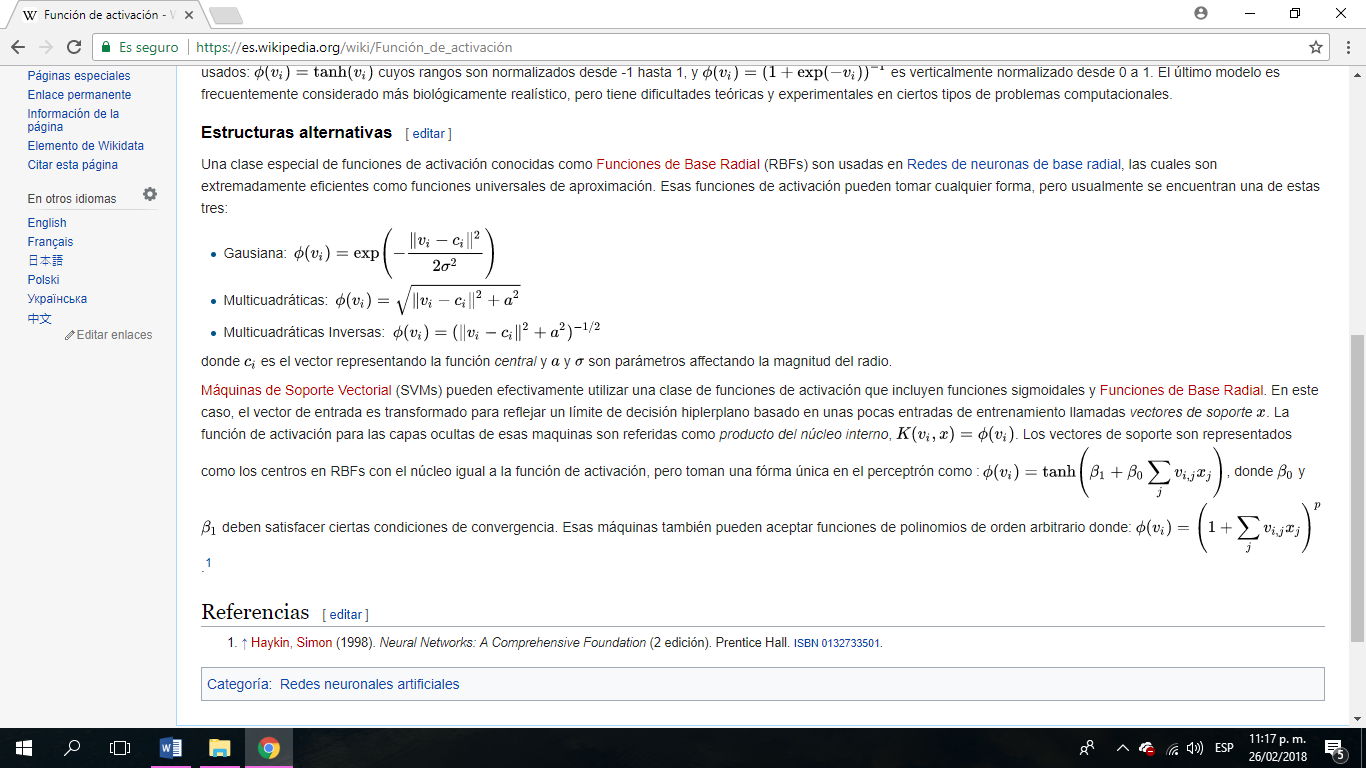
• la neurona artificial o unidad básica de inferencia en forma de discriminador lineal, a partir de lo cual se desarrolla un algoritmo capaz de generar un criterio para seleccionar un sub-grupo a partir de un grupo de componentes más grande.

La limitación de este algoritmo es que si dibujamos en un plot estos elementos, se deben poder separar con un hiperplano únicamente los elementos "deseados" discriminándolos (separándolos) de los "no deseados".

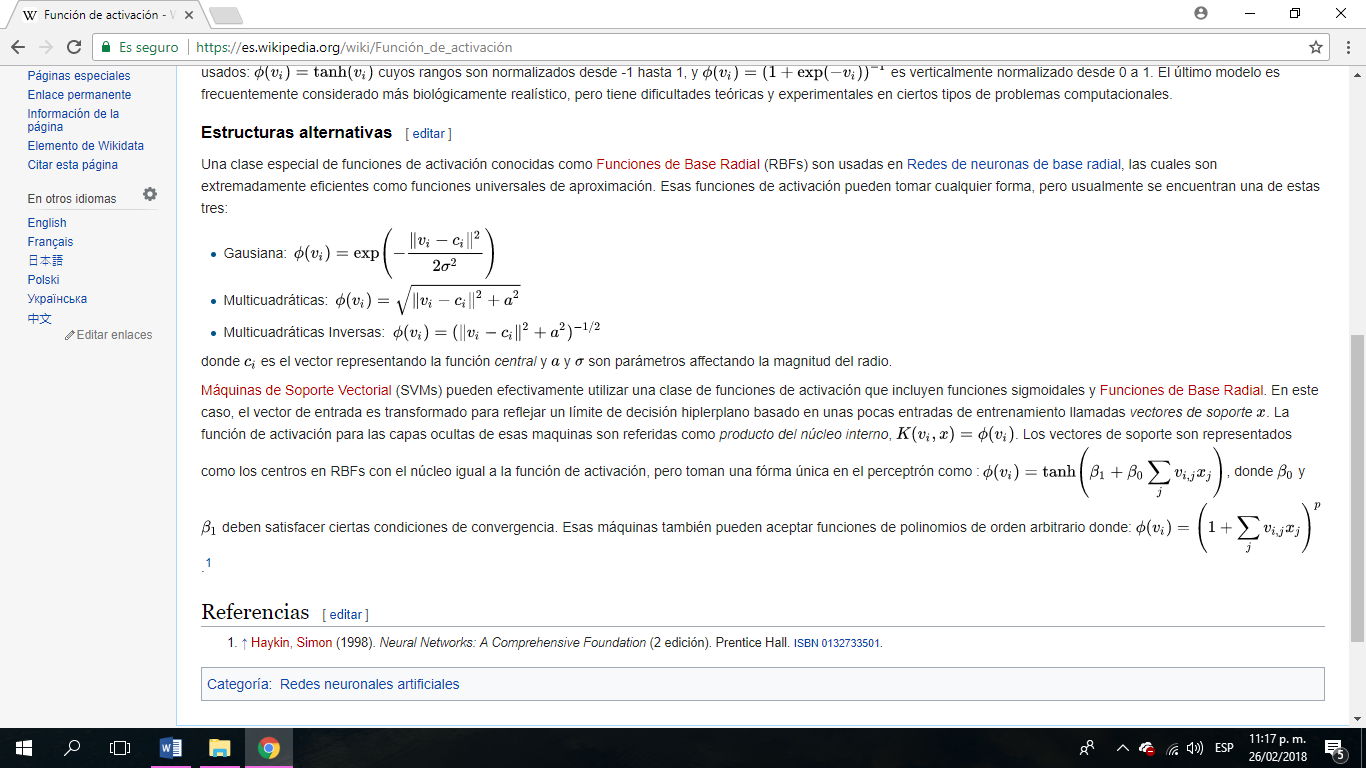
• El perceptrón puede utilizarse con otros tipos de perceptrones o de neurona artificial, para formar una red neuronal artificial más compleja.

e. ¿ Cual es su función de activación?

[Máquinas de Soporte Vectorial](https://es.wikipedia.org/w/index.php?title=M%C3%A1quinas_de_Soporte_Vectorial&action=edit&redlink=1) (SVMs) pueden efectivamente utilizar una clase de funciones de activación que incluyen funciones sigmoidales y [Funciones de Base Radial](https://es.wikipedia.org/w/index.php?title=Funciones_de_Base_Radial&action=edit&redlink=1). En este caso, el vector de entrada es transformado para reflejar un límite de decisión hiplerplano basado en unas pocas entradas de entrenamiento llamadas *vectores de soporte****{\displaystyle x}x***. La función de activación para las capas ocultas de esas maquinas son referidas como *producto del núcleo interno*, ***{\displaystyle K(v\_{i},x)=\phi (v\_{i})}.*** Los vectores de soporte son representados como los centros en RBFs con el núcleo igual a la función de activación, pero toman una fórma única en el perceptrón como :



donde {\displaystyle \beta \_{0}}β0 y {\displaystyle \beta \_{1}}β1 deben satisfacer ciertas condiciones de convergencia. Esas máquinas también pueden aceptar funciones de polinomios de orden arbitrario donde:{\displaystyle \,\phi (v\_{i})=\left(1+\sum \_{j}v\_{i,j}x\_{j}\right)^{p}}



f. ¿Cómo se entrena una red Perceptron indique el nombre del algoritmo y sus pasos?

**Entrenamiento de un perceptron.**

Para comprender como se realiza el entrenamiento de una red de neuronas, lo primero que debemos de hacer es comprender el entrenamiento de una sola neurona, para luego acoplar varios entrenamientos individuales en uno colectivo para toda la red.

Para el entrenamiento de un perceptrón lo primero que debemos hacer es inicializar aleatoriamente su vector de pesos asociado, y después ir actualizando este para conseguir mejores resultados.

|  |
| --- |
|  |
|  | http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales_archivos/image178.gif |

Para cada iteración del algoritmo se actualizará el vector de pesos:

|  |
| --- |
|  |
|  | http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales_archivos/image180.gif |

De manera que:

Siendo η un parámetro al que denominaremos tasa de entrenamiento con un valor positivo y que hará la función de hacer que la convergencia sea más o menos rápida. Es recomendable que la tasa de entrenamiento sea pequeña (entre 0.1 y 0.2) para que los resultados sean correctos.

Esta forma de recalcular los pesos se basa en el gradiente descendente de la función de error.

**Algoritmo “backpropagation”.**

Cuando necesitamos representar problemas complejos, no nos basta únicamente con un simple perceptrón, sino que necesitamos una red de perceptrones interconectados entre ellos.

Para el entrenamiento de una red hemos de tener en cuenta que la salida de cada neurona no va a depender únicamente de las entradas del problema, sino que también depende de las salidas que ofrezcan el resto de las neuronas. Por este mismo motivo también podemos afirmar que el error cometido por una neurona no solo va a depender de que sus pesos sean los correctos o no, sino que dependerá del error que traiga acumulado del resto de neuronas que le precedan en la red.

Para controlar el error cometido hemos de redefinir la función de error, de tal forma que la nueva función de error es la siguiente:

|  |
| --- |
|  |
|  | http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales_archivos/image193.gif |

Donde cada parámetro significa lo siguiente:

            -    http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales_archivos/image195.gif es el vector de pesos.

-        D es el conjunto de ejemplos de entrenamiento.

-        d es un ejemplo de entrenamiento concreto.

-        Salidas es el conjunto de neuronas de salida.

-        k es una neurona de salida.

-        tkd es la salida correcta que debería dar la neurona de salida k al aplicarle a la red el ejemplo de entrenamiento d.

-        okd es la salida que calcula la neurona de salida k al aplicarle a la red el ejemplo de entrenamiento d.

Dentro de una red tenemos varios tipos de neuronas:

            - Neuronas tipo n. Serán las neuronas de entrada a la red.

            - Neuronas tipo h. Serán las neuronas internas de la red.

- Neuronas tipo n. Serán las neuronas de salida de la red.

Nótese que el número de neuronas de entrada a la red ha de ser igual al número de entradas del problema a resolver, y que es recomendable que el número de neuronas de salida sea igual al número de patrones distintos entre los que pretendamos clasificar los ejemplos de entrenamiento, así, un determinado ejemplo de entrenamiento pertenecerá al patrón que indique la neurona de salida cuya salida se aproxime más a una cota que nosotros definamos (por ejemplo: aquella cuya salida se aproxime más a 1).

Cuando haya varias neuronas con un valor de aproximación a la cota establecida que sea muy similar, podremos decir que “lo más probable” es que la neurona sea de uno de esos patrones, sin embargo cuando solo haya una salida que se aproxime claramente más que el resto a la cota clasificaremos esa ejemplo dentro del patrón que indique esa neurona.

g. Nombre 5 ejemplos donde se evidencie el uso de adaline

1. clasificación de patrones
2. reconocimiento de patrones
3. segmentación de imágenes
4. comprensión de datos
5. asociación de patrones

6. Adaline

a. Historia

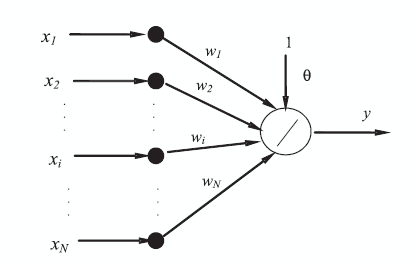
1960 - Bernard Widrow/Marcial Hoff. Desarrollaron el modelo Adaline (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real.

b. Formula Matemática, Explique sus términos.

* Las *n* entradas representan un vector {\displaystyle x}x de entrada que pertenece al espacio {\displaystyle R^{m}}.
* Por cada neurona, existe un vector {\displaystyle w}w de pesos sinápticos que indican la fuerza de conexión entre los valores de entrada y la neurona. En la práctica representan la ponderación de cada entrada sobre la neurona.
* Una constante {\displaystyle \theta }
* La salida {\displaystyle y}y de la neurona se representa por la función de activación, que se define como {\displaystyle y=\sum \_{i=1}^{n}x\_{i}w\_{i}+\theta }

c. Dibuje la estructura

Estructura de Adaline



e. ¿Cuál es su función de activación?

La función de activación es lineal

f. ¿Cómo se entrena una red Adaline indique el nombre del algoritmo y sus pasos?

El algoritmo que se utiliza es el de Mínimos Cuadrados (LMS), Tambien llamado algoritmo de Widrow-Hoff. La idea del algoritmo es ajustar los pesos, para minimizar el error cuadrático medio.

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 h 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 dependen de cada problema particular.

g. Nombre 5 ejemplos donde se evidencie el uso de adaline

Procesamientos de Señales

Canceladores de ECO, en señales telefónicas

Sistemas de predicción

Reconocimientos de patrones

7. ¿Qué es y para que se usa las redes de retropropagacion(backpropagation)?

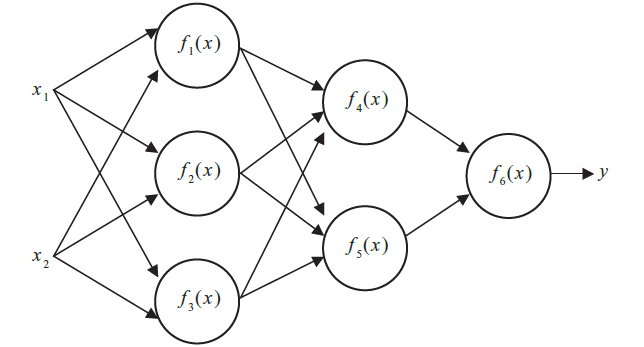
La propagación hacia atrás de errores o retropropagación (del inglés backpropagation) es un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado utilizados para entrenar redes neuronales artificiales. El método emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

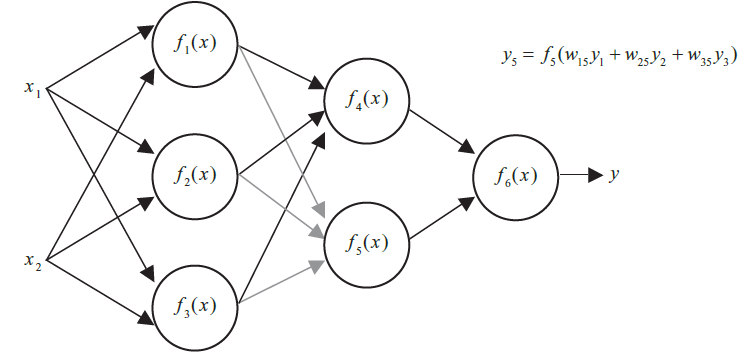
La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

8. Explique el algoritmo backpropagation, usando imágenes

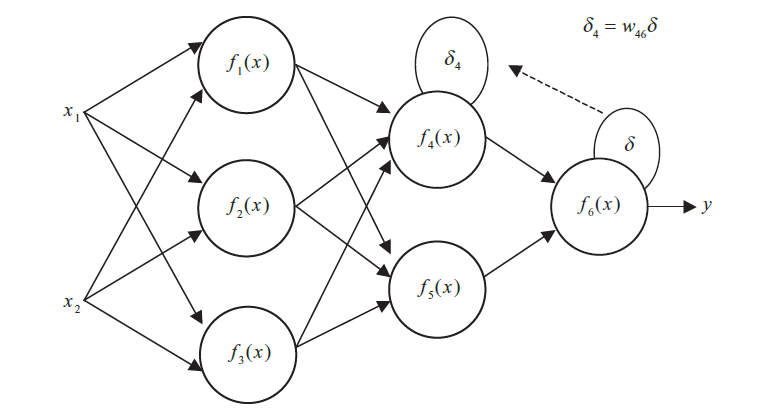
a. Red de 2 capas



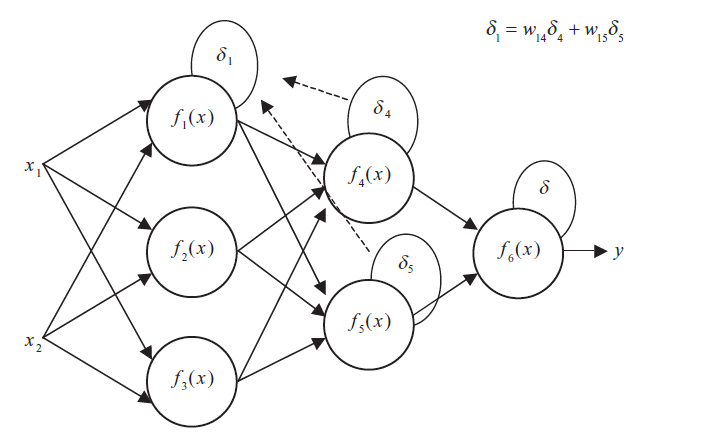
1. Propagación de señales en las neuronas



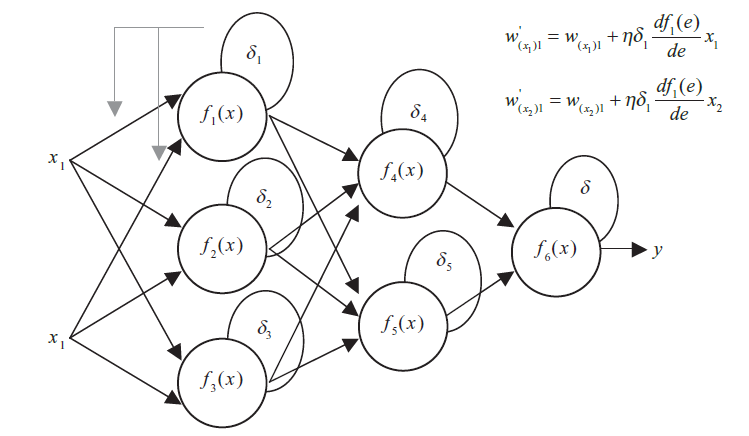
1. Retro-propagación del error



1. Retro-propagación del error



1. Actualización de los pesos



9. Haciendo uso del lenguaje de programación Python graficar las siguientes funciones

a. Funcion escalon

b. Funcion lineal y mixta

c. Funcion tangente hiperbólica

d. Funcion Sigmoidal

e. Funcion de gauss

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

Cualquier función continua definida en , n≥2, se puede representar

mediante la expresión:



donde las funciones gi son funciones continuas y reales de una sola variable, elegidas adecuadamente, y las funciones φij son continuas y monótonas crecientes independientes de f.

Este resultado establece que cualquier función vectorial continua de en definida

sobre un conjunto compacto se puede expresar en términos de sumas y composiciones de un

número finito de funciones de una sola variable.