Redes Neuronales Artificiales

Fernando Izaurieta y Carlos Saavedra
Departamento de Física, Universidad de Concepción, Concepción, Chile

RESUMEN

En esta charla se entrega una descripción de las características principales del funcionamiento de redes neuronales artificiales. En primer lugar, se presenta un modelo sencillo de red neuronal y las familias de problemas que pueden ser modeladas por ellas. Además, se describe esquemas simples de entrenamiento de redes orientadas al reconocimiento de patrones de información. Se presenta un ejemplo de aplicación de las redes al reconocimiento de texto.

1. Introducción.

Las actividades de investigación desarrolladas en torno al estudio de *redes neuronales artificiales*, simplemente redes neuronales o neuroredes, están motivadas en modelar la forma de procesamiento de la información en sistemas nerviosos biológicos. Especialmente, por la forma de funcionamiento del cerebro humano, que es completamente distinta al funcionamiento de un computador digital convencional. El cerebro humano corresponde al de un sistema altamente *complejo, no-lineal y paralelo*. En términos sencillos lo anterior equivale a decir que puede realizar muchas operaciones *simultáneamente* a diferencia de los computadores comunes que son de tipo *secuencial*, o sea, realizan sólo una operación a la vez. En este sentido, una neurored es un procesador de información, de distribución altamente paralela, constituido por muchas unidades sencillas de procesamiento llamadas neuronas. La neuroredes se caracterizan principalmente por:

- Tener una inclinación natural a adquirir el conocimiento a través de la experiencia, el cual es almacenado, al igual que en el cerebro, en el peso relativo de las conexiones interneuronales.
- Tienen una altísima plasticidad y gran adaptabilidad, son capaces de cambiar dinámicamente junto con el medio.
- Poseen un alto nivel de tolerancia a fallas, es decir, pueden sufrir un daño considerable y continuar teniendo un buen comportamiento, al igual como ocurre en los sistemas biológicos.
- Tener un comportamiento altamente no-lineal, lo que les permite procesar información procedente de otros fenómenos no-lineales.

Entre las motivaciones principales para el estudio del funcionamiento de las redes neuronales se encuentran los fenómenos neurológicos. Nuestro cerebro es un procesador de información muchísimo más eficiente que un computador. La clave de esto se encuentra en la inmensa *plasticidad* del cerebro, existen tareas cotidianas para el cerebro que sería impensable realizar mediante computación tradicional. Un ejemplo de esto es la capacidad reconocer a una persona en un tiempo de 100 a 200 ms. En ese breve lapso, el cerebro es capaz de procesar un patrón de información tridimensional, por ejemplo, de una persona que quizás ha cambiado de aspecto (luce distinto o simplemente envejeció) en un paisaje cambiante (que puede contener muchos otros rostros). En la actualidad, tareas mucho más simples consumen días de trabajo de los computadores más veloces. La plasticidad se percibe también en la capacidad de responder de forma correcta frente a un estímulo nunca antes recibido. Esa capacidad hace que cuando nos presentan por primera vez a alguien, sepamos automáticamente que es una persona y no un objeto u otro ser biológico. Debido a estas características y muchas otras, las neuroredes se han convertido en una gran ayuda en el procesamiento de datos experimentales de comportamiento complejo. Además, su comportamiento iterativo no lineal las une de modo natural al caos y teorías de la complejidad. De hecho, las posibilidades son tan amplias que se empieza a hablar de un nuevo campo, aparte de la Biología, la Matemática y la Física: las Neurociencias. Como ya lo dijimos, lo que se desea inicialmente es imitar, al menos parcialmente, el funcionamiento del cerebro. Para hacerlo revisaremos, superficialmente, algunos conceptos básicos de neurobiología.

2. Neurobiología

Una neurona típica posee el aspecto y las partes que se muestran en la figura 1. Sin embargo, debemos observar que el dibujo no está a escala, el axón alcanza un largo típico de centímetros y a veces de varios metros, las dendritas también y las terminales sinápticas, son más largas, numerosas y tupidas.

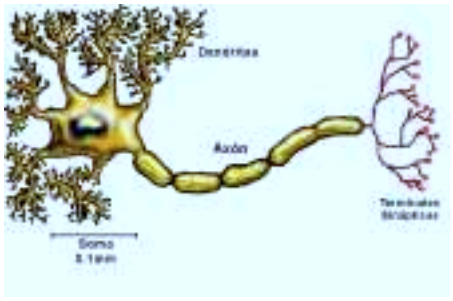


Figura 1: Neurona y sus partes.

Típicamente, las neuronas son 6 ó 5 órdenes de magnitud **más lentas** que una compuerta lógica de silicio, los eventos en un chip de silicio toman alrededor de nanosegundos (10^{-9} s), mientras que en una neurona este tiempo es del

orden de los milisegundos (10^{-3}). Sin embargo, el cerebro compensa en forma excepcional la lentitud relativa en el funcionamiento neuronal con un número inmenso de neuronas con interconexiones masivas entre ellas. Se estima que el número de neuronas en el cerebro es del orden de 10^{10}, y que el número de conexiones sinápticas es 6×10^{13}. La red resultante que es el cerebro es una estructura enormemente eficiente. Específicamente, la eficiencia energética del cerebro es aproximadamente de 10^{-16} J=(operaciones × s), la cual es del orden de 10^{10} veces mayor que la de los mejores computadores en la actualidad.

La mayoría de las neuronas codifican sus salidas como una serie de breves pulsos periódicos, llamados *potenciales de acción*, que se originan cercanos al soma de la célula y se propagan a través del axón. Luego, este pulso llega a las sinapsis y de ahí a las dendritas de la neurona siguiente.

Una *sinapsis* es una interconexión entre dos neuronas, un dibujo esquemático de ella se incluye en la figura 2. En ella, el botón sináptico corresponde al término del axón de una neurona pre-sináptica, y la dendrita es la correspondiente a una neurona post-sináptica.

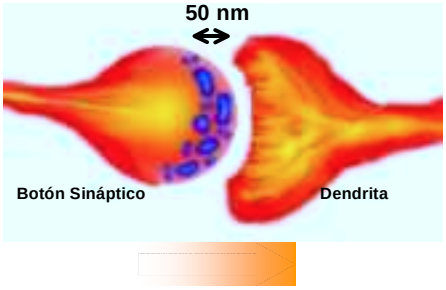


Figura 2: Salto sináptico

El tipo más común de sinapsis es la *sinapsis química*, que funciona como sigue. Una señal neural eléctrica pre-sináptica, llega al botón sináptico de la figura 2. Allí, ésta hace que las vesículas sinápticas (en azul en nuestra figura) se rompan, liberándose así una sustancia llamada *neurotransmisor*. Esta sustancia química se difunde a través del espacio entre las neuronas. Luego, es captada por la dendrita, en donde estimula la emisión de un nuevo impulso eléctrico, post-sináptico, que se propaga hacia la derecha. Así vemos que las dendritas son las zonas receptivas de una neurona, siendo el axón una línea de transmisión, y los botones terminales comunican los impulsos a otras neuronas.

En la neurona, hay dos comportamientos que son importantísimos para nosotros:

- El impulso que llega a una sinapsis y el que sale de ella no son iguales en general. El tipo de pulso que saldrá depende muy sensiblemente de la cantidad de neurotransmisor. Esta cantidad de neurotransmisor cambia durante el proceso de aprendizaje, es aquí donde se almacena la información. Una sinapsis modifica el pulso, ya sea reforzándolo o debilitándolo.

- En el soma se suman las entradas de todas las dendritas. Si estas entradas sobrepasan un cierto umbral, entonces se transmitirá un pulso a lo largo del axón, en caso contrario no transmitirá. Después de transmitir un impulso, la neurona no puede transmitir durante un tiempo de entre 0,5 ms a 2 ms. A este tiempo se le llama período refractario.

En base a estas dos características, construiremos el modelo de red neural.

3. Modelo Neuronal.

Aquí se desea introducir un modelo sencillo de la neurona, para construir redes, nuestro fin último es modelar correctamente el comportamiento global de toda la red. No se pretende modelar exactamente el comportamiento fisiológico de la neurona, sino más bien sólo sus características más relevantes, que entran en juego en su interacción con toda la red.

Tenemos un esquema de neurona en la figura 3. En él nuestra neurona de interés es la y_j. Las n neuronas x_i están enviando señales de entradas, que son los valores numéricos de "algo". Los valores w_{ji} representan los pesos sinápticos en las dendritas de y_j. Obsérvese la notación: el primer índice denota a la neurona hacia donde se dirige la información, el segundo índice denota de qué neurona procede la información.

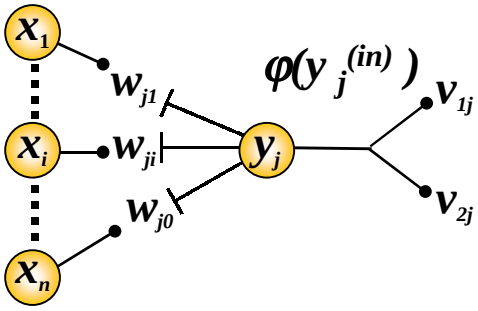


Figura 3: Esquema de Neurona.

Lo que hace cada peso sináptico es simplemente multiplicar a su entrada correspondiente y define la importancia relativa de cada entrada. Recordemos que en el soma de la neurona biológica se sumaban las entradas provenientes de todas las dendritas. Entonces tenemos que la entrada total a la neurona y_j es:

$$y_j^{(in)} = \sum_{i=1}^{n} w_{ji} x_i \qquad (1)$$

En donde el índice (in) denota "input" o entrada. Como mencionamos la neurona se activa si la entrada total supera un cierto umbral. Lo que se hace para esto es aplicar una *función de activación* φ sobre $y_j^{(in)}$, que puede ser, por ejemplo, una función tipo escalón o sigmoidea, como la tangente hiperbólica. Entonces tenemos que la señal de output o salida de la neurona y_j es:

$$y_j = \varphi\left(y_j^{(in)}\right) \qquad (2)$$

3.1. Funciones de Activación.

Algunas funciones de activación típicas, no lineales, se presentan en las figuras 4 y 5.

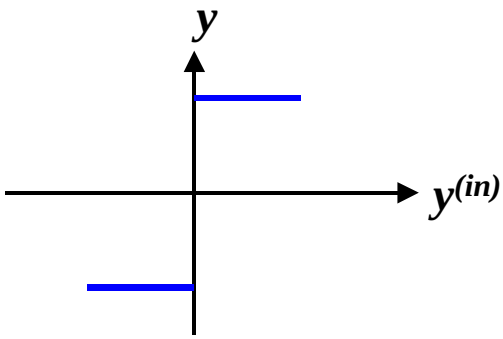


Figura 4: Escalón.

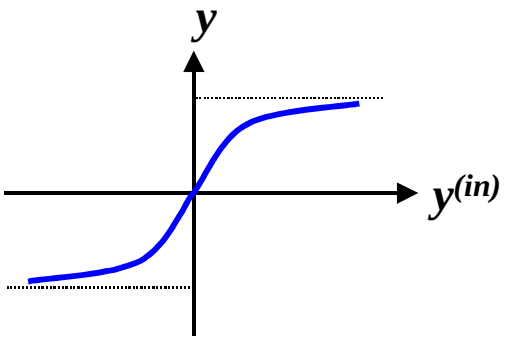


Figura 5: Sigmoidea.

Estas funciones evidentemente transmiten la idea de "disparar sobre un umbral". Las neuronas y sus funciones de activación se dividen en dos tipos: bipolares o antisimétricas y binarias. En las primeras, $-a \cdot y_j \cdot a$, siendo generalmente $a = 1$, y en las segundas, $0 \cdot y_j \cdot 1$. Además, a veces se suele usar como función de activación una relación lineal, generalmente la función identidad. Esta se usa por lo general para neuronas de entrada a la red o sensores. Esto se debe a que evidentemente, lo que esperamos de un sensor es que indique precisamente lo que está percibiendo.

Si la función de activación de una neurona es lineal, decimos que es una *neurona lineal*, en caso contrario, decimos que es una *neurona no lineal*. Aquí, las neuronas lineales se las representa por un cuadrado, y a las no lineales por un círculo.

3.2. Umbrales e Inclinación.

Anteriormente, se explicó que una neurona se activa o "dispara" si su entrada total supera un cierto umbral. Ahora bien, muchas veces es deseable modificar este umbral, haciendo más difícil que la neurona dispare (subir el umbral) o más fácil (bajar el umbral). Es posible hacer esto directamente. Sin embargo, esto suele ser un poco engorroso al programar.

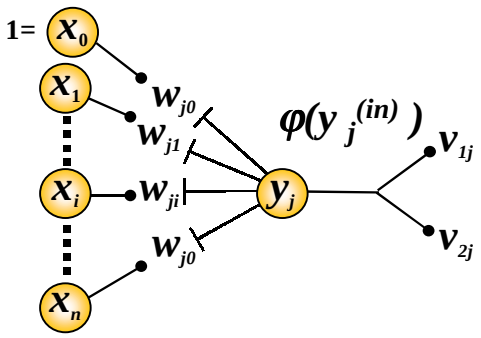


Figura 6: Esquema con Inclinación.

Resulta mucho más compacto y práctico añadir lo que se llama una *neurona de inclinación*, x_0, a la que se asigna un valor fijo de 1, y un peso sináptico w_{j0}. A la neurona y_j le asignamos un umbral fijo de cero.

Se ve claramente que esto es equivalente a que la neurona y_j tenga un umbral de $-w_{j0}$. Entonces se tiene que:

$$y_j^{(in)} = \sum_{i=0}^{n} w_{ji} x_i ; \text{ con } x_0 = 1. \qquad (3)$$

3.3. El Comienzo: McCulloch-Pitts.

Después de las definiciones previas, es conveniente revisar un ejemplo sencillo, pero muy instructivo, sobre el tema. Este consiste en el primer modelo que se creó de red neural, el año 1943, antes de que se construyeran los primeros computadores. McCulloch era un siquiatra y neuroanatomista y Pitts un matemático. El primero pasó 20 años estudiando sobre cuál era la representación de un evento en el sistema nervioso. Su modelo tiene las siguientes características:

- Las neuronas son del tipo binario, $[0; 1]$.
- Los umbrales y las sinapsis se mantienen fijas.
- La función de activación es del tipo escalón.

Ellos demostraron que todas las funciones lógicas se pueden describir mediante combinaciones apropiadas de neuronas de este tipo, y que por lo tanto, se podía crear, en principio, una red capaz de resolver cualquier función computable. Además, el modelo sirve para explicar algunos fenómenos biológicos sencillos. De esta forma es posible describir algunas funciones lógicas como:

x_1	x_2	y
1	1	1
0	1	0
1	0	0
0	0	0

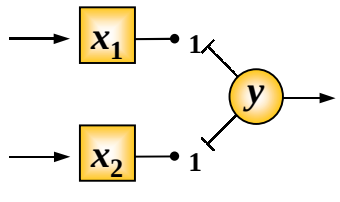


Figura 7: Función And

x_1	x_2	y
1	1	1
0	1	1
1	0	1
0	0	0

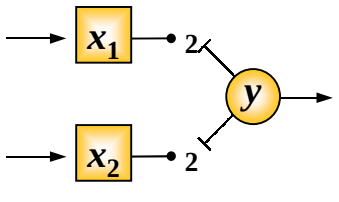


Figura 8: Función Or

En todos estos ejemplos, se supone que el umbral de cada neurona no lineal es 2. O sea,

$$y = \begin{cases} 0 \text{ si } y^{in} < 2 \\ 1 \text{ si } y^{in} \geq 2 \end{cases} \quad (4)$$

Ahora es muy fácil comprobar que las tablas de verdad efectivamente se cumplen[1], por ejemplo, la primera línea de la tabla de verdad para el **And**:

$$1 \cdot 1 + 1 \cdot 1 = 2 = y^{in} \Rightarrow y = 1$$

Veamos una función lógica más: el **xOr** u **Or excluyente**.

x_1	x_2	z
1	1	0
0	1	1
1	0	1
0	0	0

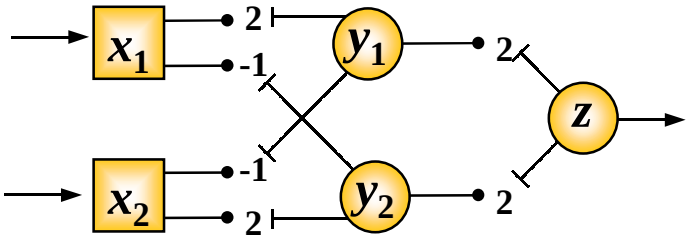


fig 9: Función xOr.

Es fácil comprobar que la red mostrada efectivamente cumple con la tabla de verdad. Sin embargo, llama la atención el que su red sea más compleja que la de las funciones **And** u **Or,** pese a que sólo se diferencia de la función **Or** en la primera línea. Pudiéramos darnos el trabajo de buscar una red diferente para representar **xOr**, buscando algo más sencillo. Existen varias otras redes que también la representan, pero ninguna de ellas sencillas como la para **And** u **Or**.

Fijémonos primero en que consiste la "complejidad". En las redes **And** u **Or** las neuronas de entrada y la de salida están conectadas directamente, en cambio, se puede demostrar que para la función **xOr** habrá **siempre** por lo menos, una conexión indirecta. Para entender esta diferencia se debe incorporar dos nuevos conceptos: Problemas linealmente separables y Capas Neurales.

3.4. Problemas Linealmente Separables y Capas Neurales.

3.4.1. Problemas Linealmente Separables.

Volvamos a una red simple, como la del And u Or, pero más general, como la de la figura 10. En ella, hemos añadido una neurona de inclinación, en vez de un umbral.

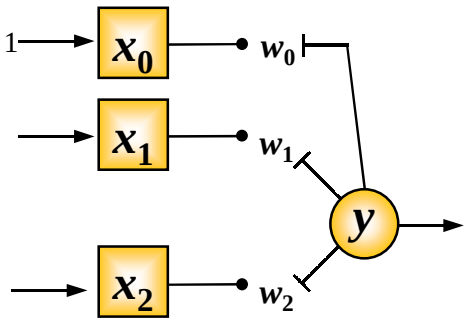


Figura 10: Función Lógica "simple".

Sabemos que la entrada $y^{(in)}$ estará dada por:

$$y^{(in)} = w_0 + w_1 x_1 + w_2 x_2; \quad (5)$$

y la respuesta, por:

[1] Se asume que 1=Verdadero y 0=Falso.

$$y = \begin{cases} 0 \text{ si } y^{(in)} < 0 \\ 1 \text{ si } y^{(in)} \geq 0 \end{cases} \quad (6)$$

Esto divide al plano formado por x_1 y x_2 en dos regiones: en una, se tendrá que $y = 0$ e $y^{(in)} < 0$, en la otra se tendrá que $y = 1$ e $y^{(in)} \geq 0$. La **frontera** entre ambas está dada por la ecuación **lineal** de la recta:

$$w_0 + w_1 x_1 + w_2 x_2 = 0:$$

Veamos por ejemplo que ocurre con la función **And**. Tenemos que $y^{(in)} = x_1 + x_2 - 2$, la frontera es $x_1 + x_2 = 2$: Si superponemos las respuestas que nos debe arrojar la red con el gráfico de las regiones, obtenemos la figura 11.

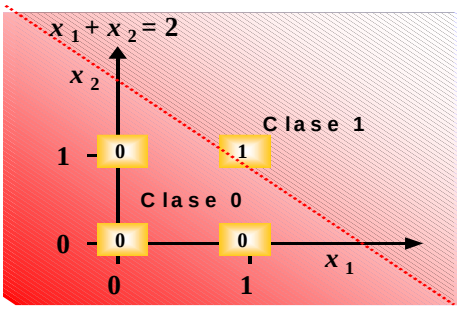


Figura 11: And sobre el plano.

Si la entrada está en la región “Clase 1” producirá una salida 1, si está en la “Clase 0”, una salida de 0. Vemos que se pueden separar las entradas que deben producir una salida 1 de las que deben producir una salida 0 por una línea recta. Se dice entonces que el problema es linealmente separable. Para resolver un problema linealmente separable, nos basta con una red “sencilla”.

Revisemos en cambio, como es la función **xOr** sobre el plano:

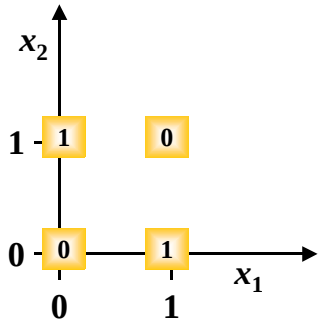


Figura 12: xOr sobre el plano.

Mediante simple inspección podemos comprobar que efectivamente es imposible encontrar una línea recta que deje a un lado las entradas que deben producir 0, y al otro, las que deben producir 1. En este caso, decimos que el problema **no es linealmente separable**. Por eso no nos basta con una red “sencilla” para resolver el **xOr**.

Lo que en realidad estamos haciendo es un caso muy sencillo del problema general de **clasificación de patrones**. Estamos clasificando las entradas en “Clase 1” o “Clase Verdadera” y “Clase 0” o “Clase Falsa”.

El concepto de separabilidad lineal se extiende de modo natural a entradas de más dimensiones. Las entradas que pertenecen a una clase y las que no pertenecen a esta simplemente tienen que poder separarse por el hiperplano $\sum_{i=0}^{n} w_{ji} x_i = 0$ en el espacio x de las entradas.

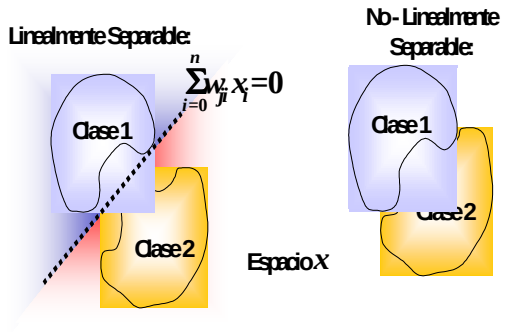


Figura 13: Separabilidad Lineal.

Para aclarar el concepto de redes sencillas primero revisaremos otro concepto: las Capas Neurales.

3.4.2. Capas Neurales

Cuando trabajamos con grandes cantidades de neuronas, es natural ordenar aquellas que tienen comportamientos similares en “capas”, como en la figura 14. De ahí que se usen los subíndices para las neuronas. Cada capa es un vector de neuronas.

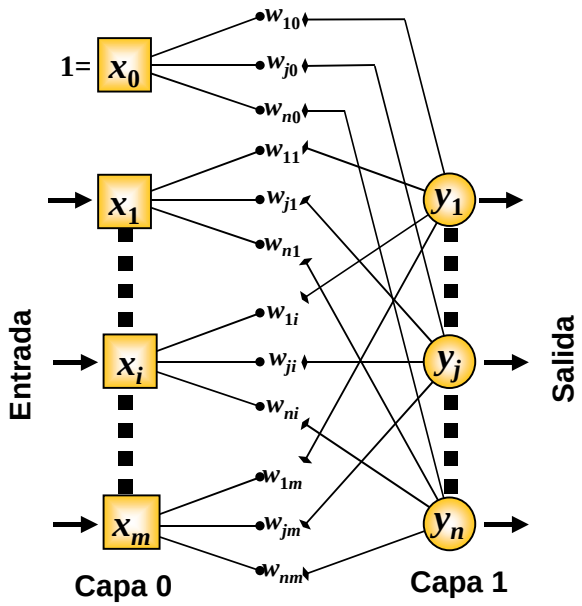


Figura 14: Red Unicapa.

Se acostumbra no contabilizar la capa de entrada, por lo tanto se dice que la red de la figura 14 es "Unicapa". Las sinapsis obviamente están ordenadas en una matriz w_{ji} de $n \times (m+1)$. Evidentemente, de nuestro análisis anterior, tenemos que una red unicapa sólo puede resolver problemas linealmente separables. En una red unicapa, las neuronas de salida pueden ser lineales o no lineales.

Pero es evidente que podemos seguir añadiendo capas, como se muestra en la figura 15.

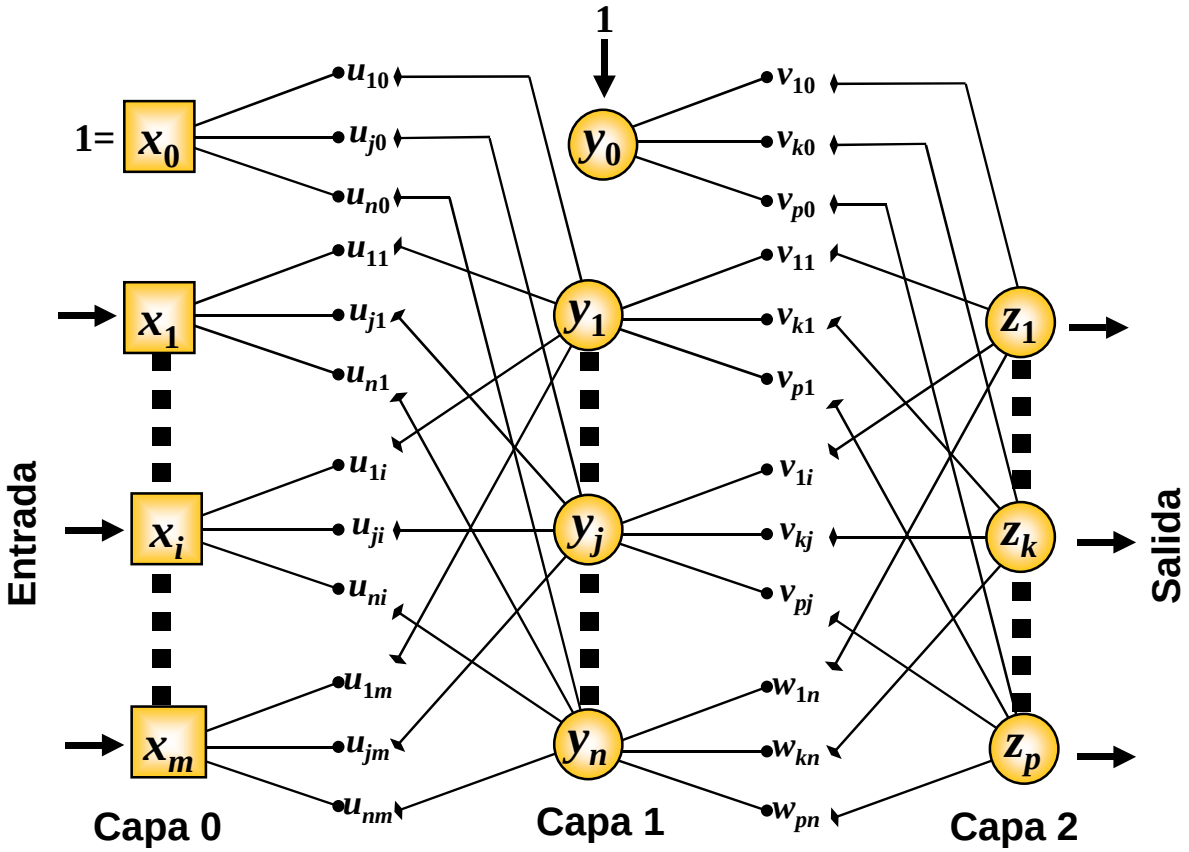


Figura 15: Red Multicapa.

En una red multicapa, las capas ocultas, que en nuestra figura corresponde a la Capa 2, **siempre** son no lineales. Se puede demostrar muy fácilmente que si se construye una red multicapa con capas ocultas lineales, ésta es equivalente a una red unicapa.

Podemos ver fácilmente la idea de paralelismo al observar las capas de las redes. Cada neurona de una capa no necesita de las demás en su misma capa para trabajar, son capaces por lo tanto de trabajar simultáneamente. Esta cualidad se ha aprovechado al diseñar chips paralelos con la nueva tecnología VLSI (**V**ery **L**arge **S**cale **I**ntegrated), en donde se han implementado varios tipos de neuroredes.

Una red multicapa es capaz de resolver problemas más complejos, pero su proceso de aprendizaje también es más complicado.

4. Aprendizaje o Entrenamiento.

El aprendizaje es la clave de la plasticidad de una neurored y esencialmente es el proceso en el que se adaptan las sinapsis, para que la red responda de un modo distinto a los estímulos del medio. Recordemos que en una neurored, toda la información adquirida se guarda en el valor de cada peso sináptico. De hecho, las neuronas de la mayor parte de los seres vivos con sistema nervioso, desde un caracol hasta el hombre son esencialmente iguales. Lo que nos hace más inteligentes que un caracol es el número, organización y modo de cambio de las conexiones sinápticas. El aprendizaje se divide principalmente en dos tipos: Aprendizaje con Profesor o Supervisado y sin Profesor o No Supervisado. Nosotros sólo estudiaremos aprendizaje con profesor y algunas variantes de éste.

4.1. Aprendizaje con Profesor o Supervisado.

El proceso es completamente análogo a enseñarle algo a un niño, digamos por ejemplo, a reconocer las vocales. Los pasos del proceso son los siguientes:

- El profesor dispone de un conjunto de N pares de entrenamiento, $\{x_i(n); d_j(n)\}_{n=1}^{N}$, en donde $x_i(n)$ es la n-ésima entrada y $d_j(n)$ es la respuesta correcta a esa entrada. En nuestro ejemplo, significa que tenemos todas las vocales dibujadas en un papel ($x_i(n)$) y que nosotros sabemos las respuestas correctas ($d_j(n)$) a cada una de las figuras, los sonidos A,E,I,O,U.

- Introducimos una de las entradas $x_i(n)$ y esperamos que nuestra red nos responda. Sería como mostrarle al niño la letra A y preguntarle: "Dime, ¿Qué letra es esta?".

² La neurored responde mediante una salida $o_j(n)$. Digamos, el niño nos respondió "Esa es una E".

² Luego comparamos ambas señales, la respuesta deseada $d_j(n)$ y la respuesta de la red $o_j(n)$, creando una señal de error, $e_j(n) = d_j(n) - o_j(n)$. "Mmm... el niño no está tan despierto como esperaba...".

² Luego, con la señal de error $e_j(n)$, corrijo las sinapsis de la red mediante algún algoritmo de los que se verá a continuación. "No hijo, esta no es una E, es una A...".

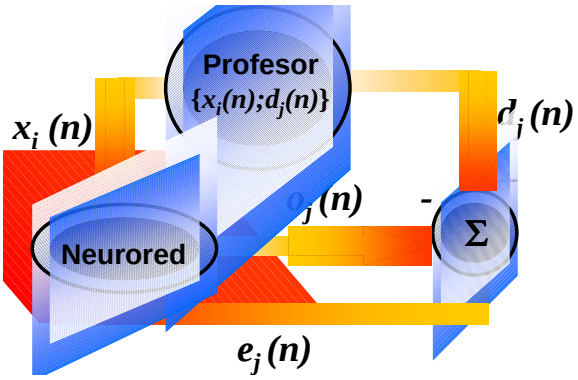


Figura 16: Aprendizaje con Profesor o Supervisado.

La secuencia completa de los N pares de entrenamiento es conocida como una **Época**. En general, pueden haber muchas épocas, y el aprendizaje se detiene cuando la red responda correctamente a todos los pares de entrenamiento.

En general, cuando adaptemos las sinapsis, la forma de hacerlo será mediante la siguiente ecuación:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \qquad (7)$$

en donde $w_{ji}(n)$ son los pesos sinápticos con los que la red responderá al n-ésimo ejemplo. Esto equivale a no cambiar los pesos sinápticos en forma radical, sino que simplemente los variamos en una cantidad "pequeña" $\Delta w_{ji}(n)$ con respecto a su estado anterior. Lo que diferencia a los algoritmos o reglas de aprendizaje, es básicamente como encontrar $\Delta w_{ji}(n)$. El que hayan distintos algoritmos tiene cierta base biológica. Neuronas de distintas partes del cerebro aprenden de forma distinta también.

4.2. Regla de Hebb.

Esta es la más antigua y la más famosa de las reglas de aprendizaje, su base es completamente biológica. Fue encontrada por el neurofisiologo Hebb en 1949, quien descubrió que si dos neuronas a ambos lados de la sinapsis estaban activas (o inactivas) simultáneamente, entonces las sinapsis entre ellas se reforzaban, y si se activaban (o desactivaban) asincrónicamente, se debilitaban. Una forma de expresar esta idea de forma sencilla es la siguiente:

$$\Delta w_{ji}(n) = \eta y_j(n) x_i(n); \qquad \eta > 0; \qquad (8)$$

donde las capas de neuronas x_i e y_j están distribuidas como en la figura 14. A la constante de proporcionalidad η se le llama "razón de aprendizaje". Para ver como funciona, supongamos que x_i e y_j son bipolares o antisimétricas, con $a = 1$. Si x_i

e y_j toman ambas **simultáneamente** el valor de 1 (o de -1), $\Delta w_{ji}(n) = \eta$, y esa sinapsis se reforzará. En cambio, si una tomase el valor -1 y la otra el de 1, $\Delta w_{ji}(n) = -\eta$, y esa sinapsis se debilitará.

Este aprendizaje explica el famoso experimento de Pavlov. Él le daba alimento a un perro y simultáneamente tocaba una campanilla. Después de repetir esto muchas veces, Pavlov tocó sólo la campanilla, sin alimento. Y el perro, sólo oyendo la campanilla, salivaba. La explicación es muy simple. Al activarse simultáneamente las neuronas que controlan la salivación y las que perciben la campanilla, las sinapsis entre ellas se refuerzan.

4.3. Aprendizaje para redes Unicapa.

4.3.1. Regla de Aprendizaje perceptrónico.

Objetivo y funcionamiento general: Esta regla de aprendizaje está diseñada especialmente para el **reconocimiento de patrones**, pero por ser red unicapa, sólo se pueden usar patrones linealmente separables. El perceptrón nació como un primer intento de modelar la retina, en 1958, por Rosenblatt. Es usual pensar que la retina es simplemente un receptor (como el detector CCD de una cámara de vídeo), pero en realidad es una red altamente compleja. Sólo ha podido ser reproducida en ojos para robots y retinas artificiales para ciegos en la última década, mediante los llamados circuitos neuromórficos. La retina, además de ser un receptor, es capaz de reconocer el movimiento y bordes, y puede adaptarse a cambios locales en el brillo.

Un perceptrón es una red de una sola capa, como la de la figura 14. Las neuronas de salida son no lineales, con función de activación tipo escalón. En nuestros experimentos numéricos, utilizamos funciones de activación bipolares o antisimétricas, como la siguiente:

$$y_j = \begin{cases} -1 \text{ si } y_j^{(in)} < 0 \\ 0 \text{ si } y_j^{(in)} = 0 \\ 1 \text{ si } y_j^{(in)} > 0 \end{cases} \qquad (9)$$

Nótese que se incluyó un punto neutro. A este se le suele llamar **punto de indeterminación**. A veces incluso se usa una región en torno al origen que produce una salida de cero, a la cual se le llama **banda de indeterminación**. Simplemente dice que la neurona no sabe que responder. Cada neurona de salida representa a una clase determinada, si una de ellas se activa con una entrada, significa que pertenece a esa clase, si está desactiva, que no pertenece. Aquí, incluimos dos experimentos al respecto, clasificando **imágenes** de letras. La entrada x_i corresponde al i-ésimo píxel de la imagen. Digamos por ejemplo que tenemos una red que nos clasifica una entrada como X u O. Lo que queremos es que funcione como se muestra en la figura 17, en donde la neurona marcada con X reconoce a la clase X, y la con O, a la clase O:

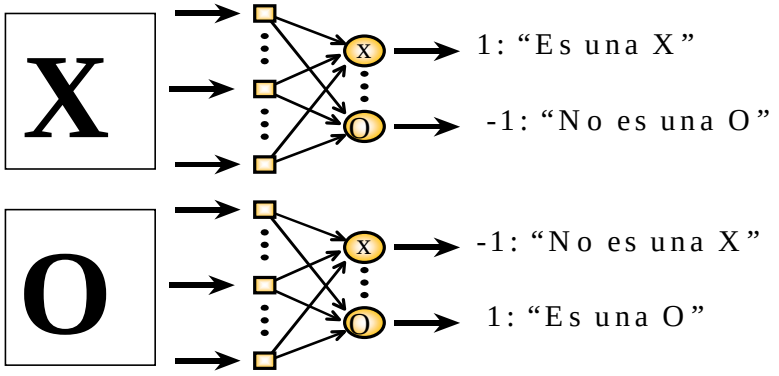


Figura 17: Funcionamiento de un Perceptrón

Algoritmo Perceptrónico. Veamos ahora como entrenar esta red que cuenta m_0 y m_1 número de neuronas de entrada y salida respectivamente. Además, existen N pares de entrenamiento $\{x_i(n); d_j(n)\}_{n=1}^{N}$. De esta forma el algoritmo es:

Paso 0: Inicializar las sinapsis de la red, se puede elegir $w_{ji}(0) = 0$ ó valores aleatorios. se elige una razón de aprendizaje η, $0 < \eta \leq 1$.

Paso 1: Mientras la condición de parada del paso 5 sea falsa, realizar los pasos del 2 al 5.

Paso 2: Para cada par de entrenamiento, $(x_i(n); d_j(n))$; $n = 1; \ldots; N$, hacer los pasos del 3 y 4.

Paso 3: $j = 1; \ldots; m_1$

$$y_j^{(in)}(n) = \sum_{i=0}^{m_0} w_{ji}(n) x_i(n)$$

$$y_j(n) = \begin{cases} -1 \text{ si } y_j^{(in)}(n) < 0 \\ 0 \text{ si } y_j^{(in)}(n) = 0 \\ 1 \text{ si } y_j^{(in)}(n) > 0 \end{cases}$$

Paso 4: Si $y_j(n) \neq d_j(n)$, para algún j entre 1 y m_1, entonces

$$w_{ji}(n+1) = w_{ji}(n) + \eta d_j(n) x_i(n);$$

donde $j = 1; \ldots; m_1; i = 0; \ldots; m_0$. En caso contrario $w_{ji}(n+1) = w_{ji}(n)$

Paso 5: Si los pesos sinápticos no cambian para cada patrón de entrenamiento durante la última vez que se realizó el paso 2, entonces parar, sino es así, continuar.

Se ve claramente que en nuestro caso, $\Delta w_{ji}(n) = \eta d_j(n) x_i(n)$ o 0, dependiendo de si hubo error o no. Podemos entender intuitivamente el algoritmo de la siguiente forma. Supongamos que la j-ésima neurona respondió de forma incorrecta, dijo -1 en vez de 1. Esto significa que $y_j^{(in)}(n)$ fue demasiado pequeño, debemos hacer que crezca haciendo que más términos en la sumatoria $\sum_{i=0}^{m_0} w_{ji}(n)x_i(n)$ sean positivos y lo máximo posible. O sea, si la i-ésima entrada, $x_i(n)$ es +1, entonces la i-ésima sinapsis, $w_{ji}(n)$, debiera ser positiva y lo más grande posible también: debemos hacerla crecer. Si por el contrario, $x_i(n)$ es -1, debemos hacer bajar a $w_{ji}(n)$. Eso es lo que se re eja en la forma en que hemos construido el $\Delta w_{ji}(n)$, si $d_j(n)$ es +1, entonces $\Delta w_{ji}(n)$ tiene el mismo signo que $x_i(n)$. En el caso contrario, es todo al revés.

Es bastante evidente que si un problema es linealmente separable, existen infinitos pesos sinápticos que servirán para solucionar el problema. Basta con multiplicar por una constante la ecuación $\sum_{i=0}^{m_0} w_{ji} x_i = 0$ y seguimos teniendo el mismo hiperplano de separación, aunque distintos pesos sinápticos. Además, generalmente, no es un solo hiperplano el que nos podría delimitar bien la frontera, sino que más bien hay infinitos, como se muestra en la figura 18:

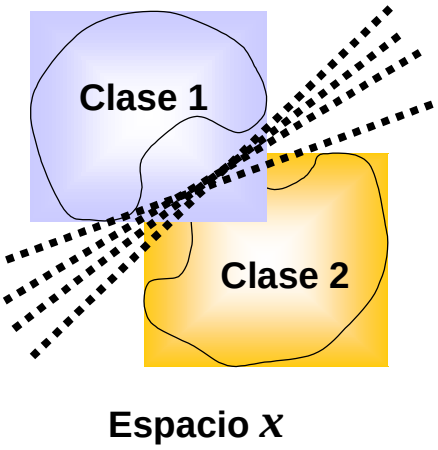


Figura 18: Infinitas Soluciones.

O sea, o no existe ninguna solución, o existen infinitas.

Es posible demostrar que **si existe** solución, entonces el algoritmo perceptrónico convergerá a una de las infinitas soluciones en un número **finito** de pasos.

Experimentos Computacionales. A modo de ejemplo se incluyen dos experimentos (computacionales), ambos de clasificación de letras. Para el primero, usamos las siguientes entradas:

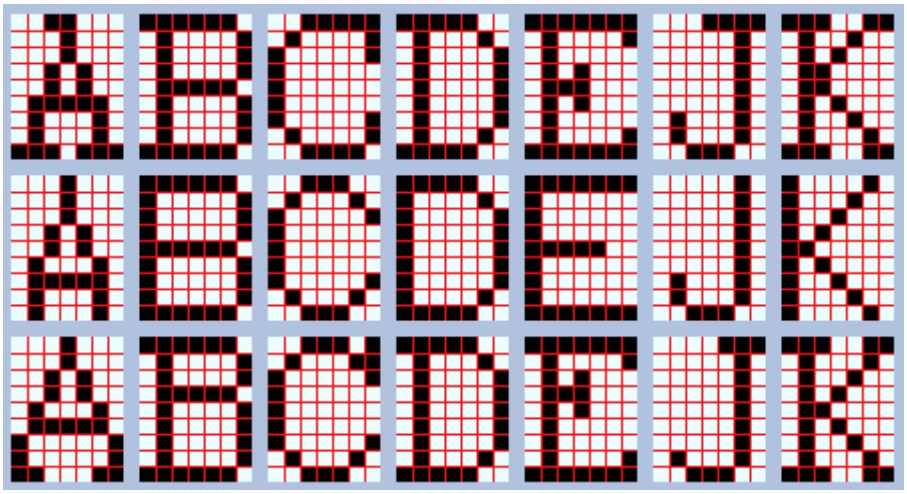
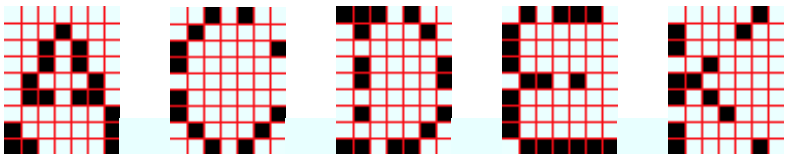


Figura 19: Patrones de entrenamiento para el Experimento 1

Cada imagen es de $7 \times 9 = 63$ píxels, un píxel negro corresponde a un +1 y uno blanco a un -1, se usó $\eta = 1$. Las sinapsis se inicializaron con 0. Para construir el vector x_i de entradas, simplemente ponemos una fila de la imagen después de la otra. Después del entrenamiento, algunos patrones que fueron clasificados correctamente fueron los siguientes:



A C D E K

Aquí observamos el funcionamiento de la red que se ha construido, que a pesar de ser muy simple, tiene **plasticidad** y es capaz de **generalizar**. A pesar de que nunca vio esos patrones con errores durante su entrenamiento, fue capaz de reconocer a qué letra correspondían.

Para ampliar el experimento nos preguntamos: ¿Se podrá realizar con patrones más grandes? y, ¿Qué imagen podemos hacernos de cómo están distribuidas las sinapsis?

Para responder esas preguntas, construimos un perceptrón que sólo clasificara entre X, O e I, pero con entradas de una resolución mucho mayor: $56 \times 72 = 4032$ pixeles. Trabajamos exactamente del mismo modo que con el ejemplo anterior. Los patrones de entrenamiento ahora son los siguientes:

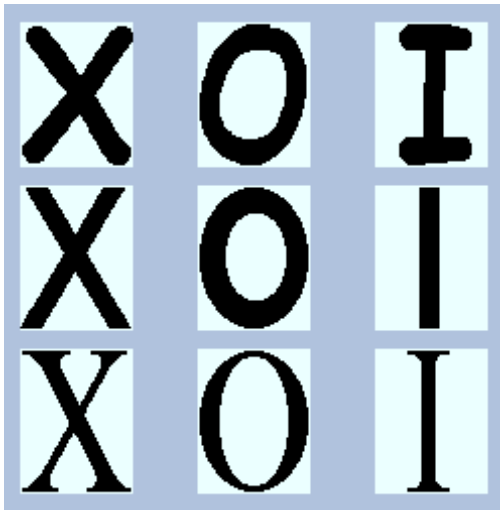


Figura 21: Patrones de entrenamiento, 2

Se necesitaron sólo tres épocas. Algunos patrones que fueron clasificados correctamente, son:

X O I

Nuevamente observamos la plasticidad. Pero, ¿cómo se distribuyen las sinapsis?. Para verlo de un modo gráfico, simplemente reordenamos en la misma forma de la imagen original a las sinapsis, obteniéndose 3 gráficas: Una para las sinapsis que se conectan con la neurona de la X, otra con la de la O y otra con la de la I.

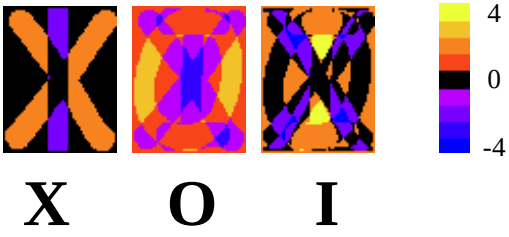


Figura 23: Sinapsis para X,O e I.

Simplemente observando se puede entender cómo funcionan las sinapsis, y qué regiones son más cruciales que otras al reconocer el patrón.

Pero dijimos que las sinapsis no eran únicas. Si empezamos con valores iniciales aleatorios llegamos a otro tipo de conexiones sinápticas, como estas:

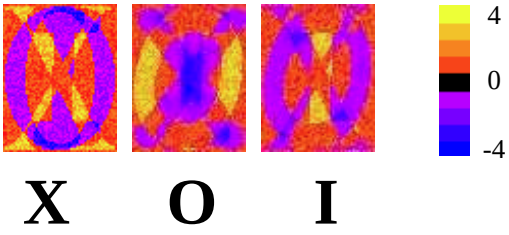


Figura 24: Otras Sinapsis para X, O e I.

Ahora, pasaremos a otra tarea que realizan muy bien las neuroredes: predecir.

4.3.2. Regla Delta, o corrección de error.

Esta es una regla muy popular, en ella se usa una red de una sola capa, igual que la perceptrónica, pero la neurona de salida tiene una función de activación derivable, generalmente la función identidad o la tangente hiperbólica. Para esta regla, usamos un algoritmo más sencillo, simplemente calculamos el error $e_j(n) = d_j(n) - y_j(n)$ correspondiente a cada entrada, y luego corregimos las sinapsis de la red mediante la regla:

$$\Delta w_{ji}(n) = \eta e_j(n) \varphi'_j(y_j^{(in)}(n)) x_i(n) \qquad (10)$$

Si las neuronas de salida tienen a la identidad como función de activación, $\varphi'_j(y_j^{(in)}(n)) = 1$, y entonces,

$$¢ w_{j\,i}(n) = ´ e_j(n) x_i(n)$$

que es la forma más común del algoritmo. Esta regla en realidad es un caso particular muy sencillo del algoritmo de retropropagación de errores.

La convergencia del algoritmo depende fuertemente del valor de ´. Si se elige uno muy pequeño, la convergencia se hará muy lenta, y si se elige muy grande, el proceso se volverá inestable y no convergerá. Existen criterios para determinar cotas superiores para ´, pero suelen ser complicados, nosotros en nuestro experimento simplemente recurrimos al ensayo y error, que suele ser mucho más rápido.

Predictor lineal, o filtro Lineal Adaptativo. Supongamos que tenemos un sistema dinámico al que estamos describiendo por un único parámetro x. Lo único que conocemos de él es su historia, muestreando x cada cierto período T. Lo que queremos hacer es predecir cuál será la respuesta del sistema en el próximo instante. Esto lo hacemos mediante una interacción red - sistema dinámico como la mostrada en la figura 25:

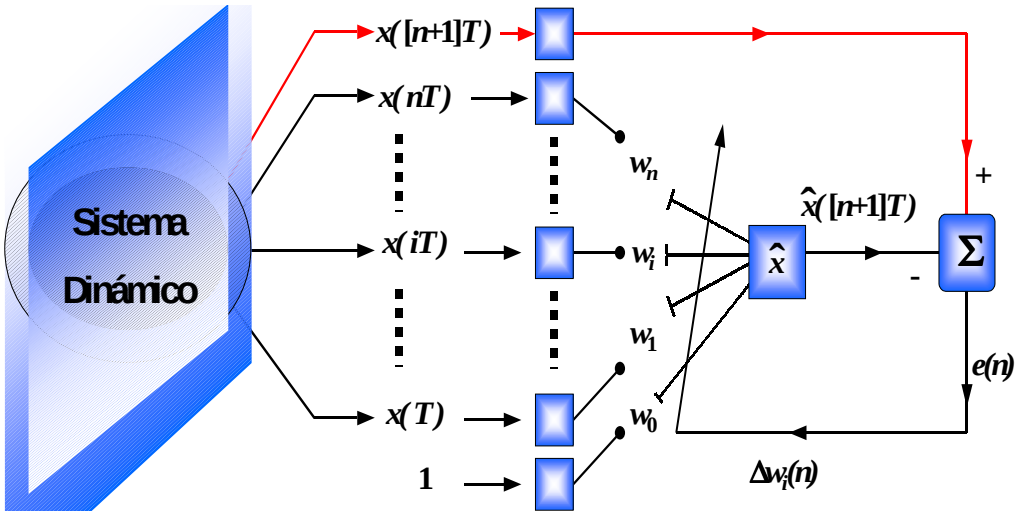


figura 25: Predictor Lineal

Aquí vemos que el papel de profesor es llevado de modo automático por el mismo sistema dinámico. La red conoce todas las entradas desde x(T) hasta x(nT), y debe predecir el valor de x([n + 1]T). El papel de respuesta deseada lo juega x([n + 1]T) y el de entrada el historial del proceso. Es completamente análogo al proceso de aprendizaje con profesor, excepto por que el número de neuronas de entrada debe aumentar constantemente.

Experimento computacional. Usamos un ´ = 0:01 y una neurona de salida con la función identidad. Nuestro sistema dinámico era una señal senoidal con variaciones aleatorias.

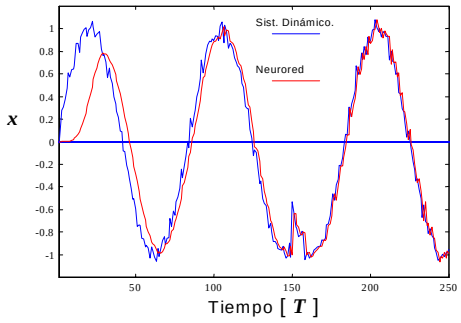


Figura 26: Experimento de Predicción.

Se puede observar fácilmente que a medida que la red va aprendiendo, cada vez es capaz de predecir mejor.

4.4. Aprendizaje para Redes Multicapa.

Ahora, romperemos con nuestras limitaciones anteriores y estudiaremos el caso nolineal. Debemos recordar que en este tipo de redes, las funciones de activación de las capas ocultas son siempre no lineales. Además, veremos de las ecuaciones que necesitamos una función de activación diferenciable en todo su dominio. Además, se encuentra que el algoritmo de aprendizaje es más difícil de visualizar. Nosotros sólo estudiaremos un tipo de aprendizaje, el Aprendizaje Retropropagador de Error.

4.4.1. Estructura y Notación general

La estructura de la red se muestra en la figura 27, la capa de salida es la capa L-ésima, y tiene m_L neuronas. La de entrada es la capa 0, y tiene m_0 neuronas. Decimos que nuestra red tiene L capas, a L se le llama a veces la **profundidad** de la red.

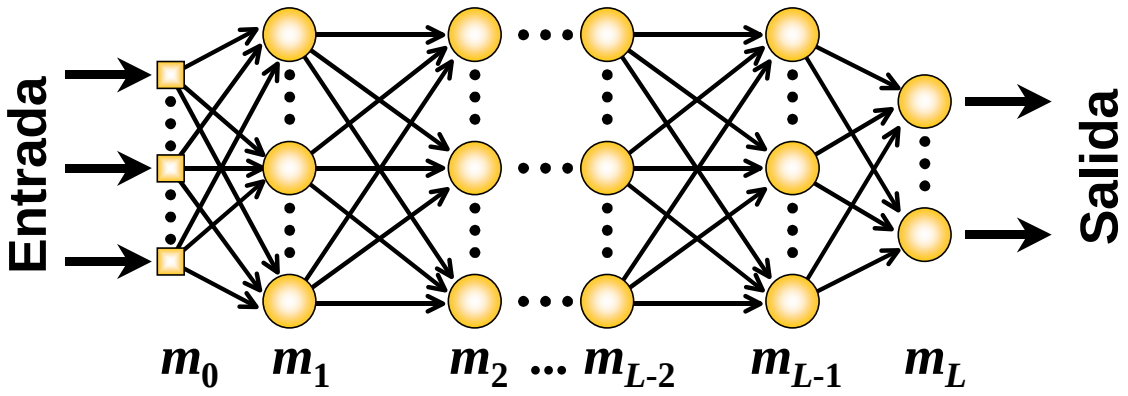


Figura 27: Red Multicapa.

Supondremos que cada capa tiene sus neuronas de inclinación, que por lo general no dibujaremos en los diagramas. En general, las neuronas de cada capa están completamente conectadas con las de la siguiente.

En el funcionamiento de nuestra red, nos encontraremos con dos tipos de señales: Señales de Función y Señales de error.

- **Señales de Función:** Es el estímulo que entra en la capa 0, y pasa hacia adelante, capa por capa del modo tradicional, hasta la última capa, L, en donde se genera la señal de salida.

- **Señales de Error:** Luego de la etapa hacia adelante, viene la retropropagación del error, hacia atrás. Cuando corregimos las sinapsis, corregimos las de la capa L primero. Luego, observando las sinapsis de la capa L, corregimos las de la capa $L - 1$, y así sucesivamente hasta la primera capa. A esto se le llama señal de error, vamos desde las últimas capas hasta las primeras corrigiendo sinapsis. Esto es lo que se ilustra en la figura 28:

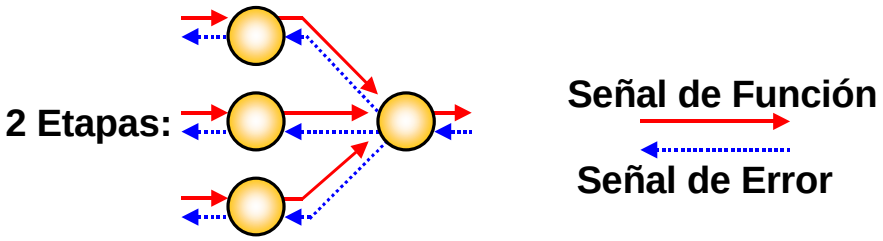


Figura 28: Etapas hacia adelante y hacia atrás.

4.4.2. Definiciones.

Error: Supongamos que la capa de salida está constituida por las neuronas z_k. Entonces, el error cometido al presentarse el n-ésimo par de entrenamiento es:

$$e_k(n) = d_k(n) - z_k(n). \qquad (11)$$

"Energía" de error: La "energía" de error al presentarse el n-ésimo par de entrenamiento es:

$$\varepsilon(n) = \frac{1}{2}\sum_{k=1}^{N_L} e_k^2(n) \qquad (12)$$

Esta **no** es una energía física, en la jerga de las neuroredes sólo se le llama así por su forma análoga a la energía cinética.

Muchos físicos han trabajado en este campo, y han empleado términos de la Física.

Energía promedio de error. Es el promedio de la energía de error durante una época completa de presentación de patrones.

$$\varepsilon_{pro} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n) \qquad (13)$$

donde $\varepsilon(n)$ y ε_{pro} son funciones de todas las sinapsis de la red. El objetivo del proceso de aprendizaje será minimizar ε_{pro}. Sea w_{ji} una sinapsis cualquiera de la red. Es fácil ver que $\varepsilon_{av}(w_{ji})$ y $\varepsilon(n)(w_{ji})$ constituyen **superficies** de error. La idea del algoritmo será la del descenso paso a paso. Vamos a hacer una primera aproximación para aclarar conceptos.

El gradiente de ε_{av} señala su dirección de crecimiento. Evidentemente, viene dado por:

$$\nabla_i \varepsilon_{pro}(w_{ji}) = \frac{\partial \varepsilon_{pro}}{\partial w_{ji}}; \quad (14)$$

Si queremos minimizar ε_{pro}, deberíamos dirigirnos en **contra** del gradiente, como vemos en la siguiente relación:

$$w_{ji}(p+1) = w_{ji}(p) - \eta \frac{\partial \varepsilon_{pro}}{\partial w_{ji}(p)} \quad (15)$$

En donde p simplemente señala que estamos en el p-ésimo paso. Lo que estamos haciendo es "esquiar" o "resbalarnos" sobre la superficie de error, tratando de llegar al mínimo global de la superficie. Sin embargo, haciendo esto, corremos el peligro de quedar atrapados en un minímo **local** de la superficie, y nunca alcanzar el mínimo global, como se ilustra en la figura 29.

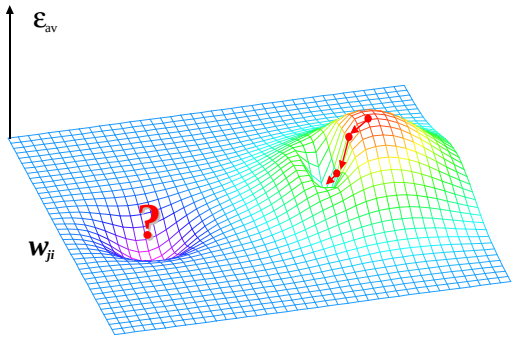


Figura 29: Peligro de caer en mínimo local.

Se puede intentar evitar esto tratando de minimizar las $\varepsilon(n)$ en vez de ε_{pro}, pero de un modo bien especial, como se explica en la siguiente sección.

4.4.3. Idea del Algoritmo.

Intentaremos minimizar ε_{av} minimizando las $\varepsilon(n)$. Es decir, tendremos que:

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}; \quad (16)$$

Cada patrón que se presenta tiene una superficie de error $\varepsilon(n)$ **diferente**. Lo que hacemos es presentar el n-ésimo par de entrenamiento y corregir **todas** las sinapsis de la red. Es decir tenemos la n-ésima superficie y nos "resbalamos" un paso. Luego, presentamos el $(n+1)$-ésimo par de entrenamiento y corregimos nuevamente todas las sinapsis de la red. O sea, **cambiamos** de superficie y nos "resbalamos" otro paso. Este constante cambio de superficie hace muy difícil quedar atrapado en un mínimo local. Una buena imagen mental sería estar esquiando en una montaña, ¡que está temblando alocadamente!. Evidentemente, tarde o temprano llegaremos al valle más profundo que exista. Este poceso se ilustra en la figura 30.

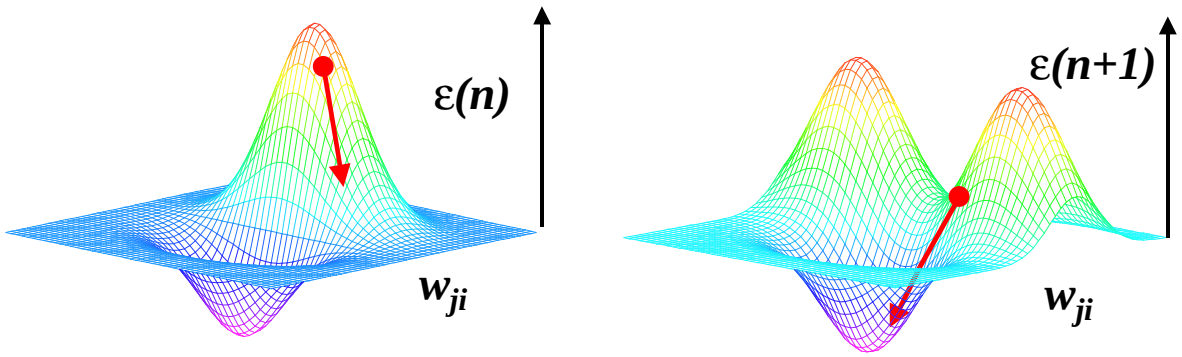


Figura 30: Esquivando Minímos Locales.

Lo que estamos suponiendo implícitamente es que el promedio de los cambios individuales en las sinapsis es un **estimador** del cambio que debiera ocurrir si minimizaramos directamente ε_{pro}.

Además, el orden de presentación de los pares de entrenamiento se **randomiza** de época en época. Esto hace que la trayectoria seguida sobre la superficie sea completamente estocástica. Supongamos que no randomizáramos el conjunto de entrenamiento. Entonces tendríamos que época tras época estaríamos repitiendo el **mismo** procedimiento, llamémoslo F. Entonces estaríamos **iterando**:

$$w_{ji}((n+1)\text{-Época}) = F(w_{ji}(n\text{-Época})) \quad (17)$$

Desde Teoría de Caos, sabemos que procesos como estos pueden converger a estados metaestables, como ciclos límites. Para eliminar esta posibilidad se intenta randomizar el conjunto de entrenamiento, mediante:

$$w_{ji}(1\text{-Época}) = F(w_{ji}(0\text{-Época}))$$
$$w_{ji}(2\text{-Época}) = G(w_{ji}(1\text{-Época}))$$
$$w_{ji}(3\text{-Época}) = H(w_{ji}(2\text{-Época})), \text{ etc.}$$

4.4.4. Algoritmo de Retropropagación de Error.

Consideremos la red de la figura 31. No dibujamos las sinapsis, sólo las señalamos con una echa. La red puede seguir teniendo más capas ocultas hacia atrás. Se puede demostrar que:

$$\Delta w_{kj}(n) = -\frac{\partial \mathcal{E}(n)}{\partial w_{kj}} = -\delta_k(n) y_j(n), \qquad k = 1,\ldots,m_L; \quad j = 0,\ldots,m_{L-1} \tag{18}$$
$$\Delta v_{ji}(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_{ji}} = -\delta_j(n) x_i(n), \qquad j = 1,\ldots,m_{L-1}; \quad i = 0,\ldots,m_{L-2}$$

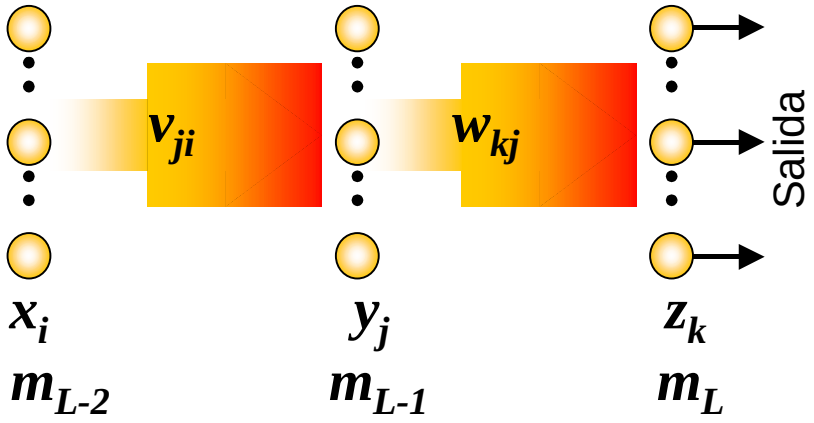


Figura 31: Red Multicapas.

O sea, cada cambio en las sinapsis es directamente proporcional a las señales enviadas por la capa que se encuentra antes. $\Delta w_{kj} \propto y_j$; $\Delta v_{ji} \propto x_i$, y así para todas las capas de la red.

Si la capa es de salida o si es oculta, el **gradiente local** $\delta_j = -\frac{\partial \mathcal{E}(n)}{\partial y_j^{(in)}}$ se calcula de formas diferentes.

Capa de Salida:

$$\delta_k^{(L)}(n) = e_k^{(L)}(n) \varphi_k'(z_k^{(in)}), \qquad k = 1,\ldots,m_L \tag{19}$$

Añadimos el superíndice L (aunque con la notación que estamos usando no es estrictamente necesario) para recalcar que nos referimos a la capa de salida. Usamos φ_k', la $'$ significa derivada con respecto al argumento, y el subíndice k se refiere a que cada neurona, en general, ¡pudiera tener una función de activación distinta!.

Este es el mismo resultado que teníamos para el filtro lineal adaptativo de una sola capa. Así que ahora sabemos de donde venía esa ecuación. Simplemente estábamos tratando de minimizar ε_{av} de la forma ya descrita.

Capas Ocultas:

$$\delta_j^{(L-1)}(n) = \varphi_j'(y_j^{(in)}(n)) \sum_{k=1}^{m_L} \delta_k^{(L)}(n) w_{kj}^{(L)}(n); \tag{20}$$

donde $j = 1,\ldots,m_{L-1}$. Aquí vemos que el $\delta_j^{(L-1)}$ depende de los $\delta_k^{(L)}$ de la capa de más adelante. Con las otras capas ocultas, se hace exactamente lo mismo, siempre la corrección depende, de la misma forma, de lo que haya sucedido en las capas de más adelante. Es a lo que nos referíamos con que la señal de error va hacia atrás.

A este método, en donde evitamos los mínimos locales, se le llama **Método Secuencial**. Algunas veces, pese al riesgo de caer en mínimos locales, se minimiza directamente ε_{av}, llamándose a esto **Método Grupal**, que no es muy usado. Aunque las características estocásticas del método secuencial hacen que evite caer a un mínimo local, hacen que sea muy difícil establecer teóricamente la convergencia. En cambio, con el método grupal, la convergencia a un mínimo **local** está garantizada con un mínimo de condiciones. Pese a ello, el método secuencial es altamente popular, pues es muy simple de implementar computacionalmente y además, efectivamente funciona muy bien en una inmensa mayoría de casos difíciles.

5. Conclusiones

Este trabajo ha pretendido realizar una pequeña introducción a algunas características de neuroredes conocidas. De hecho, aún los estudios más avanzados que existen hoy día sobre el tema están muy alejados de entender el funcionamiento del cerebro, que fue su motivación inicial. El tema en realidad es **muy** vasto. Sin embargo, pese a que hemos visto una parte ínfima del total, hemos podido apreciar algunas cualidades de este mecanismo de procesamiento de información.

En primer debemos lugar destacar que es posible modelar el funcionamiento de una neurona en forma extremadamente simple, y sin embargo, posee una gran capacidad, vemos la sencillez y la complejidad unidas de un modo maravilloso. Por ejemplo, de describió la posibilidad de procesar cantidades increíbles de información en forma paralela, de un modo sencillo y natural.

Al poder establecerse las funciones lógicas mediante la combinación de neuronas vemos también la posibilidad de poder construir computadoras con ellas, por lo menos en principio.

Otra características fundamentales que no podemos olvidar son la Robustez y la Capacidad de Aprendizaje. Las neuronas son capaces de imitar y predecir el comportamiento de sistemas dinámicos sin usar ningún modelo explícito, y capaces de reconocer patrones, aunque éstos tengan errores.

Además de todo eso, son muy interesantes para la Física, tanto para procesar información como en sí mismas. En cuanto a esto último, se han descubierto interesantes áreas que relacionan las neuroredes con la Teoría de la Información, el Caos, la Mecánica Estadística.

6. Bibliografía

[1] Laurene Fausett, *Fundamentals of Neural Networks* (Prentice-Hall, New Yersey, USA, 1994).

[2] Simon Haykin, *Neural Networks* (Prentice-Hall, New Yersey, USA, 1999).