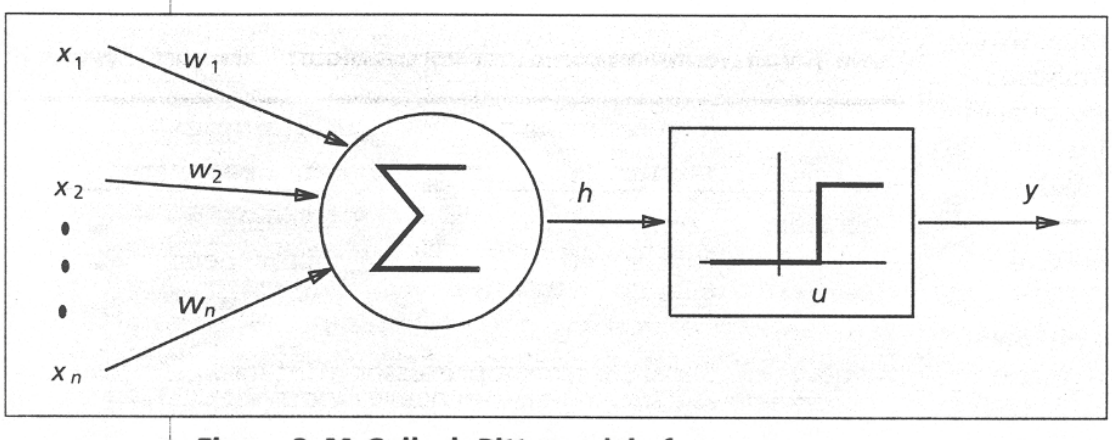
REDES NEURALES

REFERENCIAS

- **1943. McCULLOCH Y PITTS. ("A Logical Calculus of Ideas Immanent in Neurous Activity").**
- **1960. ROSENBLATT. El Perceptron.**
- **1982. HOPFIELD. Enfoque energético. Algoritmo de aprendizaje de propagación hacia atrás para perceptrones multicapa. WERBOS.**
- **RUMELHART Y McCLELLAND. PDP.**

McCULLOCH Y PITTS

Modelo computacional para una neurona artificial: unidad de umbral binario.



Modelo de Neurona Artificial de McCulloch y Pitts

Esta Neurona computa una suma ponderada de sus n señales de entrada, x_j, j= 1, 2,...,n, y genera un resultado de 1 si esta suma supera un cierto umbral u. 0 en otro caso.

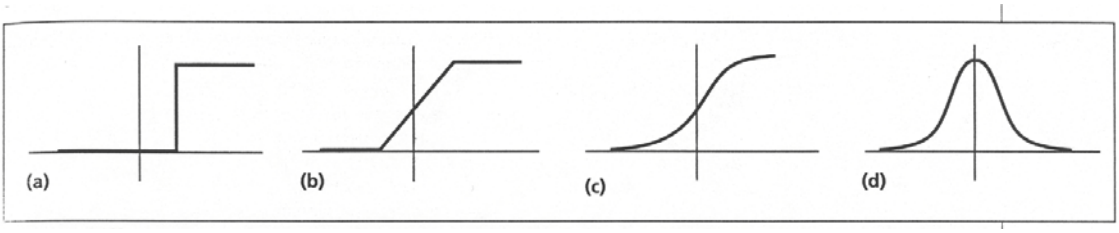
$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right)$$

Donde: θ(.) es una función de paso de unidad en 0, y w_j es el peso de la sinapsis asociado con la j-ava entrada.

Por simplicidad podemos considerar el umbral u como otro peso w_0= -u asociado a la neurona con un input constante x_0=1.

Pesos positivos corresponden a sinapsis excitadoras, mientras que los negativos a inhibidoras.

La Neurona de McCulloch y Pitts se ha generalizado de muchas maneras. En general, usando distintas funciones de activación, sigmoides, gaussianas. etc.



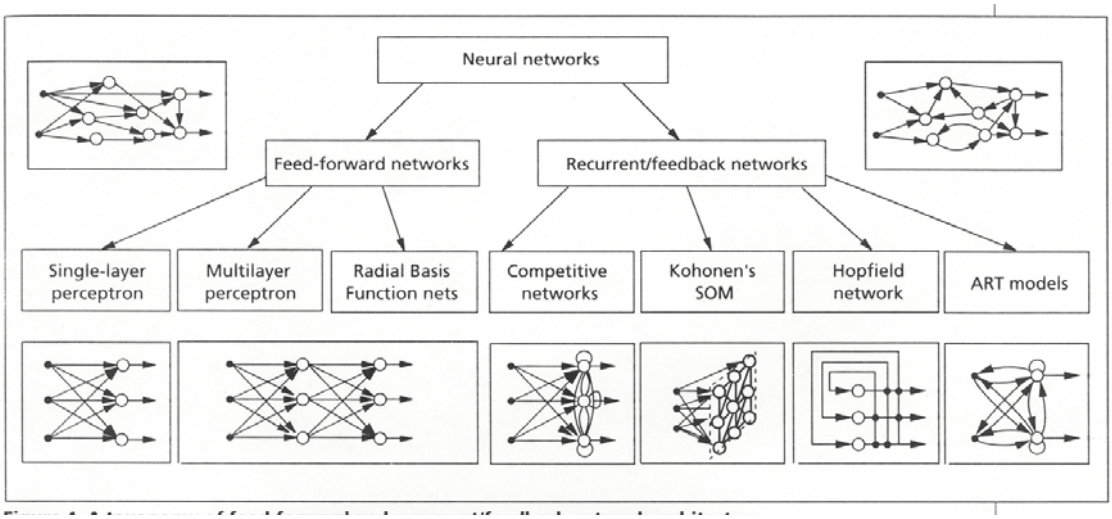
Distintas Funciones de Activación: (a) umbral (b) Lineal (c) sigmoide (d) Gaussiana

ARQUITECTURAS DE RED

Una Red de Neuronas Artificial puede considerarse como un grafo dirigido ponderado en el que neuronas artificiales son nodos y las hojas dirigidas y ponderadas son conexiones entre salidas y entradas de neuronas.

Dependiendo del patrón de conexión pueden dividirse en:

- **<u>Redes de Propagación</u> hacia delante (feedforward): en las que los grafos no tienen bucles.**
- **<u>Recurrentes o de Retroalimentación</u> (feedback), en las cuales los bucles ocurren debido a conexiones de retroalimentación**



Clasificación de los tipos de redes

REDES DE PROPAGACIÓN HACIA DELANTE

- **Son estáticas. Producen sólo un conjunto de valores como resultado, mejor que una secuencia de valores a partir de un input dado.**
- **Sin memoria: en el sentido en que su respuesta a un input es independiente del estado previo de la red.**

PERCEPTRON

Consiste en una neurona con pesos ajustables, w_j= 1,2,...,n y un umbral u.

Dado un vector de entrada $x=(x_1,x_2,...x_n)^t$ el input a la neurona es

$$v = \sum_{j=1}^{n} w_j x_j - u$$

La salida y es +1 si v>0 y 0 en otro caso.

En un problema de clasificación en dos clases, el perceptrón asigna un patrón de entrada a una clase si y=1 y a la otra si y=0.

La ecuación $\sum_{j=1}^{n} w_j x_j - u = 0$ **líneal define el límite de decisión (un hipercubo en el espacio n-dimensional de entrada) que divide por la mitad el espacio.**

ALGORITMO DE APRENDIZAJE DEL PERCEPTRON

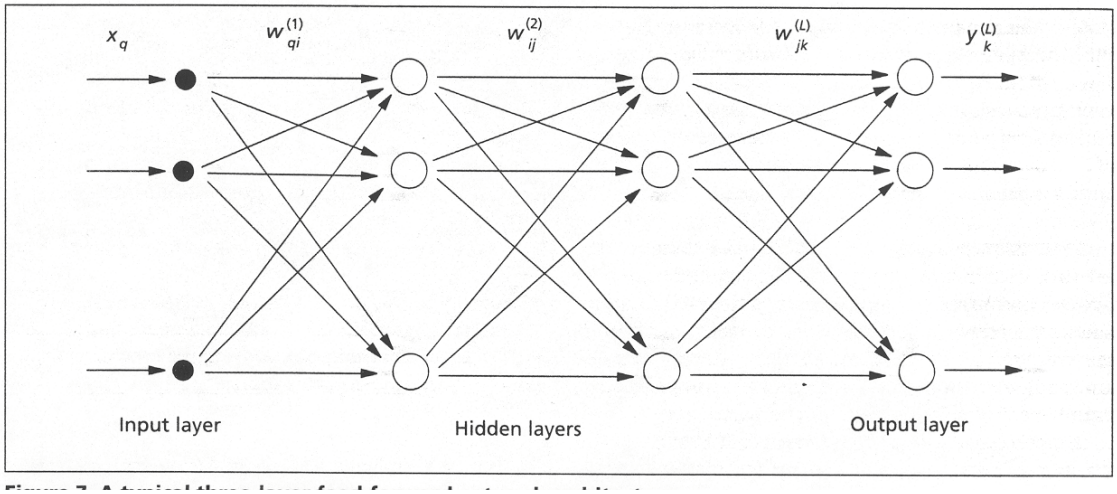
(1) Inicializar los pesos y el umbral a un número pequeño aleatorio.
(2) Presentar un vector patrón $(x_1, x_2,..., x_n)^t$ y evaluar el resultado de la neurona.
(3) Modificar los pesos de acuerdo con:

$$w_j(t+1) = w_j(t) + \eta(d-y)\, x_j$$

donde d es el output deseado, t es el número de iteración y $\eta(0.0 < \eta < 1.0)$ es el aumento (el tamaño del paso)

PERCEPTRÓN MULTICAPA

- **Un estandard red L-capa feedforward consiste en una capa de entrada, L-1 capas ocultas y una capa resultante de unidades sucesivamente conectadas (global o localmente) hacia delante sin conexiones entre unidades de la misma capa y sin retroalimentación entre capas.**



Red de propagación hacia delante de tres capas

Denotamos $w_{ij}^{(l)}$ como el peso de la conexión entre la i-ava unidad en la capa L-1 y la j-ava unidad en la capa L.

Sea { $(x^{(1)}, d^{(1)}), (x^{(2)}, d^{(2)}),..., (x^{(p)}, d^{(p)})$} un conjunto de p patrones de entrenamiento (pares input-output), donde $x^{(i)} \in R^n$ es el vector de entrada en el espacio patrón n-dimensional, y $d^{(i)} \in [0,1]^m$, un hipercubo m-dimensional, m es el número de clases. La función de coste error-cuadrado más frecuentemente usada es:

$$E = \frac{1}{2}\sum_{i=1}^{p} \left\| y^{(i)} - d^{(i)} \right\|^2$$

- **Cada unidad computacional emplea o la función umbral o la función sigmoide.**
- **Pueden formar límites de decisión complejos arbitrariamente y representar cualquier función booleana.**

RED DE FUNCIÓN DE BASE RADICAL (RBF)

- **Tiene Dos capas**
- **Es una clase espacial de red multicapa hacia delante**
- **Cada unidad en las capas ocultas emplea una función de base radial, tal como un kernel gaussiana, como función de activación. La función de Base Radial o función núcleo se centra en el punto especificado por el vector de peso asociado con la unidad. Tanto las posiciones como las**

anchuras de estos núcleos deben aprenderse por patrones de entrenamiento.

- **Cada unidad resultante implementa una combinación lineal de estas funciones de base radial.**
- **Hay diversos algoritmos para redes RBF**

  - **Estrategia de aprendizaje en dos pasos o aprendizaje híbrido.**

    (1) Estima posiciones y anchuras centrales usando un algoritmo de agrupamiento no supervisado

    (2) Se usa un algoritmo supervisado de cuadrado medio mínimo (CMS) para determinar los pesos de las conexiones entre capas ocultas y capa de salida.

    (3) Una vez que se obtiene esta solución, un algoritmo supervisado basado en el gradiente se usa para refinar los parámetros de la red.

    Este algoritmo híbrido cnverge más rápidamente que el de propagación hacia atrás, pero las redes RBF exigen muchas capas ocultas lo que ralentiza las ejecuciones cuando ya se ha estabilizado la red.

Al diseñar una red de propagación hacia delante hay que tener en cuenta:

- **Cuántas capas necesitamos para la tarea a realizar.**
- **Cuántas unidades por capa.**
- **Cómo se comportará la red con datos no incluidos en el conjunto de entrenamiento.**
- **Qué tamaño exige el conjunto de entrenamiento para una buena generalización.**

REDES RECURRENTES O DE RETROALIMENTACIÓN

MAPAS AUTOORGANIZATIVOS DE KOHONEN

- **Los Mapas Auto-organizativos (SOM) preservan la topología. En proyecciones que preservan la topología patrones de entrada cercanos activarán unidades de salida cercanas en el mapa.**
- **Un SOM de Kohonen consiste fundamentalmente en un array bidimensional de unidades, cada una conectada a todos lo n nodos de entrada.**
- **Sea w_{ij} el vector n-dimensional asociado con la unidad localizada en (i,j) del array 2D. Cada neurona computa la distancia euclidiana entre el vector de entrada x y el peso almacenado en el vector w_{ij}.**
- **Este SOM es un tipo especial de red de aprendizaje competitivo, que define una vecindad espacial para cada**

unidad de salida. la forma de la vecindad local puede ser cuadrada, rectangular o circular. El tamaño de la red inicial suele fijarse a ½ o a 2/3 del tamaño de la red y reducirla posteriormente de acuerdo con un plan (por ejemplo, una función exponencial decreciente).

ALGORITMO DE APRENDIZAJE SOM

(1) Inicializar los pesos a números pequeños aleatorios; fijar la ratio de aprendizaje y de vecindad inicial.

(2) Presentar un patrón x, y evaluar resultados

(3) Seleccionar la unidad (c_i,c_j) con el resultado mínimo:

$$\left\| x - w_{cicj} \right\| = \min_{ij} \left\| x - w_{ij} \right\|$$

(4) Modificar todos los pesos de acuerdo con la siguiente regla de aprendizaje

$$w_{ij}(t+1) = \begin{cases} w_{ij}(t) + \alpha(t)[x(t)-w_{ij}(t)], \text{ si } (i,j) \in N_{cicj}(t) \\ w_{ij}(t) \text{ en otro caso} \end{cases}$$

donde $N_{cicj}(t)$ es la vecindad de la unidad (c_i,c_j) en el tiempo t, y $\alpha(t)$ es la ratio de aprendizaje.

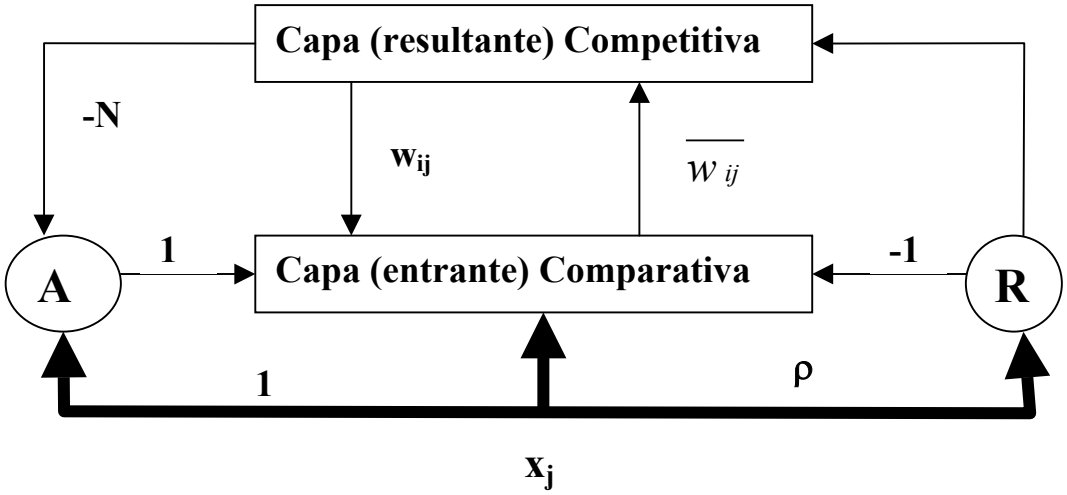
(5) Disminuir el valor de $\alpha(t)$ y reducir la vecindad $N_{cicj}(t)$.

(6) Repetir de 2 a 5 hasta que el cambio en los valores de los pesos sea menor que el umbral preespecificado o se alcance el número máximo de iteraciones.

MODELOS DE LA TEORÍA DE LA RESONANCIA ADAPTATIVA (MODELOS ART)

- **Sobre el dilema estabilidad-plasticidad: ¿Cómo podemos aprender nuevas cosas (plasticidad) asegurándonos la estabilidad de conservar el conocimiento existente? Los modelos ART (Carpenter y Grossberg ART1, ART2, ARTMap) intentan solucionar este problema.**
- **La red tiene un suplemento suficiente de unidades de salida, pero no se usarán hasta que resulte necesario. Una unidad se dice que está *comprometida* (*no-comprometida*) si está (no está) siendo usada.**
- **El algoritmo de aprendizaje modifica los prototipos almacenados de una categoría sólo si el vector de entrada es lo suficientemente similar a ellos. (si son *resonantes*).**
- **La extensión de similaridad es controlada por un parámetro de vigilancia, ρ, con $0 < \rho < 1$, que también determina el número de categorías.**
- **Cuando el vector de entrada no es lo suficientemente similar a algún prototipo existente, se crea una nueva categoría y se asigna a una unidad no-comprometida como prototipo inicial. Si no hay una unidad tal la entrada nueva no produce respuesta en la red.**

ART1



El gráfico ilustra un ART1 simplificado. Consta de dos capas de unidades completamente conectadas. Un vector arriba-abajo w_j está asociado con la unidad j en la capa de entrada y un vector abajo-arriba $\overline{w_{ij}}$ está asociado con la unidad de salida i; $\overline{w_{ij}}$ es la versión normalizada de wi,

$\overline{w_i} = \frac{w}{\varepsilon + \sum_j w_{ji}}$ **donde ε es un número pequeño usado para romper empates en la selección de la unidad ganadora. El vector arriba-abajo almacena los prototipos. El papel de la normalizacióm es prevenir que prototipos de vectores grandes se distancien de prototipos dominantes que tengan vectores pequeños.**

Dado un vector de entrada de n-bit x, el resultado de la unidad auxiliar A viene dado por:

$$A = Sgn_{0/1}(\sum_j x_j - n\sum_i o_i - 0.5)$$

donde Sgn es la función signum que produce +1 si x≥0 y 0 en otro caso.

Y la salida de una unidad de entrada viene dado por:

$$V_j = Sgn_{0/1}(x_j + \sum_j x_{ji} o_i + A - 1.5) = \begin{cases} x_j & \text{si ningún } O_j \text{ se activa} \\ x_j \wedge \sum_i w_{ij} O_i & \text{, en otro caso} \end{cases}$$

Se genera una señal R de reinicialización sólo cuando la similaridad es menor que el nivel de vigilancia.

ALGORITMO DE APRENDIZAJE PARA ART1

(1) Inicializar $w_{ij} = 1$, para todo i,j. Activar toda las unidades de salida.

(2) Presentar un nuevo patrón x

(3) Encontrar la unidad ganadora i* entre las unidades de salida activadas $\overline{w}_{i*} \cdot x \geq \overline{w}_j \cdot x$, para todo i.

(4) Realizar el test de vigilancia $r = \dfrac{\overline{w}_{j*} \cdot x_i}{\sum_j x_j}$

si r ≥ ρ (resonancia), ir al paso 5. En otro caso, desactivar la unidad i* e ir al paso 3 (hasta que todas las unidades queden desactivadas).

(5) Modificar los pesos del vector ganador w_{j*i}, activar todas las unidades e ir al paso 2

$$\Delta w_{ij*} = \eta (V_j - w_{ji*})$$

(6) Si todas las unidades de salida están desactivadas, seleccionar una de las unidades de salida no comprometidas y fijar el peso del vector a x. Si no existe, la capacidad de la red se ha alcanzado y se rechaza el patrón de entrada.

RED DE HOPFIELD

Dos versiones:

- **Binaria**
- **Continuamente valorada.**

Sea v_i el estado o salida de la i-esima unidad. Para redes binarias v_i es o +1 0 –1.

Para redes contínuas v_i puede ser cualquier valor entre 0 y 1.

Sea w_{ij} el peso de las sinapsis en la conexión de las unidades i,j.

En las redes de Hopfield, $w_{ij} = w_{ji}$, para todo i,j, y $w_{ii} = 0$, para todo i.

El comportamiento dinámico para la red binaria es:

$$V_i = Sgn\left(\sum_j w_{ij} v_j - \theta_i\right)$$

La modificación dinámica puede realizarse sincrónica o asincrónicamente:

- **Sincrónicamente: Todas las unidades se modifican simultáneamente en cada paso de tiempo. Un reloj central sincroniza el proceso.**
- **Asincrónicamente: se selecciona una unidad en cada momento y se modifica su estado.**

La función energía para la red binaria es un estado $v=(v_1,v_2,..,v_n)$ viene dada por:

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j$$

La propiedad central de la función energía es que como el estado de la red evoluciona de acuerdo con la dinámica de red, la energía de la red siempre decrece y eventualmente alcanza un punto mínimo local (atractor) donde la red permanece con energía constante.

Si un conjunto de patrones se almacena en estos atractores puede usarse como memoria asociativa.

TABLA DE MODELO-APRENDIZAJE-TAREAS

Paradigma	Regla	Arquitectura	Algoritmo de Aprendizaje	Tarea
Supervisado	Corrección de errores	perceptrón simple o multicapa	- aprendizaje del perceptrón - Propagación hacia atrás - Adaline y Madaline	Clasificación de Patrones. Predicción de la aproximación de funciones y control
	Boltzmann	Recurrente	aprendizaje de Boltzmann	Clasificación de Patrones
	Hebbian	Multicapa de propagación hacia delante	Análisis Lineal discriminante	Análisis de Datos Clasificación de Patrones
	Competitivo	Competitivo	Cuantización del vector de aprendizaje	Categorización en clases Compresión de datos
		Red ART	ARTMap	Clasificación de Patrones Categorización en clases
No Supervisado	Corrección de errores	Multicapa de propagación hacia delante	Proyección de Sammon	Análisis de datos
	Hebbian	De propagación hacia delante o competitivo	Análisis de componente principal	Análisis de datos Compresión de datos
		Red de Hopfield	Aprendizaje de memoria asociativa	Memoria asociativa
	Competitivo	Competitivo	Cuantizacion de vector	Categorización Compresión de datos
		SOM de Kohonen	SOM de Kohonen	Categorización Análisis de datos
		Red ART	ART1, ART2	Categorización
Híbrido	Corrección de errores y Competitivo	Red RBF	Algoritmode aprendizaje RBF	Clasificación de patrones Predicción de la función de aproximación Control

APRENDIZAJE EN REDES DE NEURONAS

El proceso de aprendizaje de una red de neuronas puede verse como:

- **El problema de modificar la arquitectura de la red y los pesos de las conexiones, de tal manera que la red pueda realizar eficientemente una tarea específica.**
- **La red debe usualmente aprender los pesos de las conexiones a partir de patrones de entrenamiento disponibles.**
- **Se realiza mediante la modificación iterativa de los pesos en la red.**

Para comprender o diseñar un proceso de aprendizaje es necesario:

(1) Un paradigma de aprendizaje: la información disponible a la red.

(2) Reglas de aprendizaje que gobiernan el proceso de modificación de pesos.

(3) Un algoritmo de aprendizaje.

PARADIGMAS DE APRENDIZAJE

- **SUPERVISADO: A la red se le ofrece una respuesta correcta para cada patrón de entrada. Los pesos se ajustan para aproximar la respuesta de la red a la respuesta correcta conocida.**
- **NO-SUPERVISADO: Se explora la estructura subyacente o correlaciones entre patrones en los datos, y se organizan estos patrones en ctegorías a partir de las correlaciones encontradas.**
- **HÍBRIDO: Combina los dos anteriores. Parte de los pesos se determinan mediante un proceso supervisado, mientras que el resto se obtienen mediante un aprendizaje no-supervisado.**

La teoría debe evaluar:

- **La Capacidad: Cuántos patrones pueden almacenarse y qué funciones y límites de decisión puede formar la red.**
- **La Complejidad de muestra: Cuántos patrones de entrenamiento son necesarios para entrenar a la red y que ésta garantice una generalización válida.**
- **La Complejidad Computacional: El tiempo requerido para que un algoritmo de aprendizaje estime una solución a partir de los patrones de entrenamiento.**

REGLAS DE APRENDIZAJE

REGLAS DE CORRECCIÓN DE ERRORES

Para aprendizaje Supervisado.

Sea y el resultado generado por la red, sea d el resultado esperado. El principio básico de las reglas de error-corrección es la señal de error (d-y)

Algoritmo de propagación hacia atrás

(1) Inicializar los peso a valores aleatorios pequeños.

(2) Elegir aleatoriamente un patrón de entrada $x^{(\mu)}$

(3) Propagar la señal hacia delante a través de la red

(4) Computar δ_i^L en la capa resultante $(o_i = y_i^L)$

$$\delta_i^L = g'(h_i^L)\left[d_i^u - y_i^L\right]$$

donde h_i^L representa el input a la red en la unidad i-ava en la l-ava capa y g' es la derivada de la función de activación g.

(5) Computar las deltas para las capas precedentes propagando los errores hacia atrás.

$$\delta_i^L = g'(h_i^|)\sum_j w_{ij}^{|+1}\delta_j^{|+1}$$

para |= (L – 1),...,1.

(6) Modificar los pesos usando:

$$\Delta w_{ji}^{|} = \eta \delta_{i}^{|} y_{j}^{|-1}$$

(7) Ir al paso 2 y repetir para el siguiente patrón hasta que el error en la capa de salida esté por debajo del umbral preespecificado o se alcance un número máximo de iteraciones.

APRENDIZAJE DE BOLTZMANN

- **Las máquinas de Boltzmann son redes recurrentes simétricas que consisten en unidades binarias (+1 para 'on' y –1 para 'off'). Por simetría entendemos que los pesos en la conexión entre la unidad i y la unidad j es igual al peso en la conexión entre la unidad j y la unidad i ($w_{ij} = w_{ji}$)**
- **Un subconjunto de neuronas visibles interactúan con el entorno, el resto, las ocultas, no lo hacen. cada neurona es una unidad estocástica que genera un resultado (o estado) de acuerdo con la distribución de Boltzmann de la mecánica estadística.**
- **Una máquina Boltzmann opera de dos modos:**
  - **Restringido: en donde las neuronas visibles están restringidas a estados específicos determinados por el entorno.**
  - **Libres: Todas las neuronas (visibles y ocultas) operan libremente.**

- **El objetivo del aprendizaje de Boltzmann es ajustar los pesos de las conexiones de tal forma que las unidades visibles satisfagan una distribución de probabilidad particular deseada. De acuerdo con esto, el cambio en los pesos es dado por:**

$$\Delta w_{ji} = \eta \left(\overline{\rho}_{ij} - \rho_{ij} \right)$$

donde η es la ratio de aprendizaje $\overline{\rho}_{ij}$ y ρ_{ij} son las correlaciones entre los estados de la unidad i y j. Cuando la red opera en modo restringido o en modo libre respectivamente. Los valores de ρ se estiman habitualmente mediante el método de Monte Carlo, que es muy lento.

REGLA DE HEBB

Se funda en la observación biológica de que si las neuronas de ambos lados de la sinapsis se activan sincrónica y repetidamente, la fuerza de la sinapsis se incrementa selectivamente.

Matemáticamente, la regla de Hebb se expresa:

$w_{ij}(t + 1) = w_{ij}(t) + \eta y_i(t)\, x_i(t)$

donde x_i e y_i son los valores resultados de la neurona i y j que están conectados en la sinapsis w_{ij} y η es la ratio de aprendizaje.

REGLAS DE APRENDIZAJE COMPETITIVO

- **Aquí las unidades de salida compiten entre sí para su activación. De tal manera que sólo una unidad de salida esta activa en un momento dado *(winner-take-all)* .**
- **El aprendizaje competitivo a menudo agrupa o categoriza los datos de entrada. Patrones similares son agrupados por la red y representados por una única unidad. El agrupamiento se hace automáticamente mediante correlación de datos.**
- **La red más simple de aprendizaje competitivo consiste en una capa de unidades de salida. Cada unidad i está conectada a todas las unidades de entrada mediante ponderaciones. Cada unidad de salida se conecta también a todas las demás mediante ponderación inhibitoria, pero tiene una autoretroalimentación con un peso excitatorio.**
- **Como resultado de la competición, solo la unidad i con el mayor (o menor) input llega a ser la ganadora.**

$$w_{i^*} \cdot x \geq w_i \cdot x \quad \forall \, i \; o \; \left\| w_{i^*} - x \right\| \leq \left\| w_i - x \right\|$$

Una regla de aprendizaje competitivo puede expresarse:

$$\Delta w_{ij} = \begin{cases} \eta(x_j^u - w_{ij}^*), & i = i^* \\ 0, & i \neq i^* \end{cases}$$

Mediante esta regla sólo se modifican los pesos de la unidad ganadora. Con esta regla la red no dejará de aprender hasta que la ratio de aprendizaje ηsea 0

APLICACIONES

OCR (Reconocedor Óptico de Caracteres) mediante redes de Neuronas

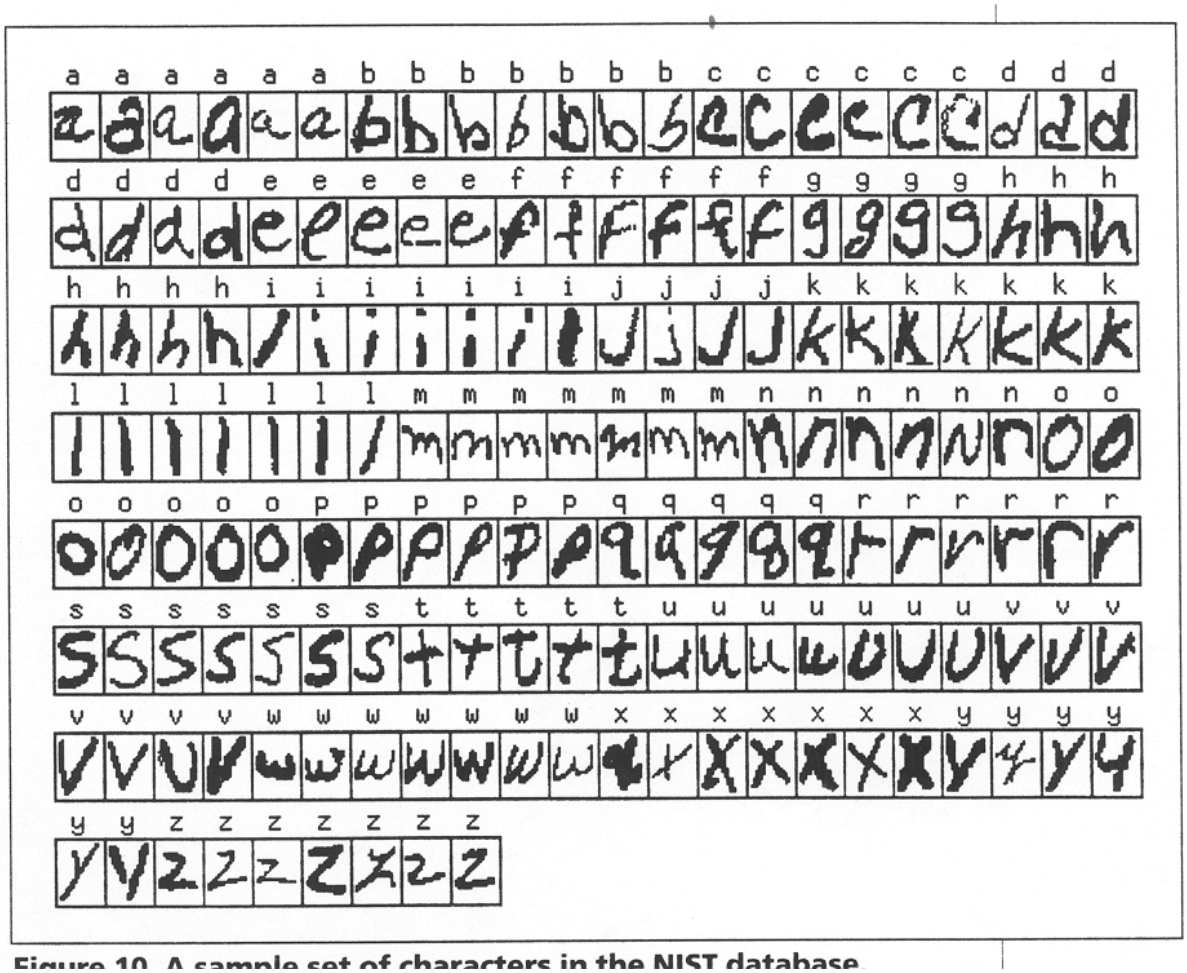
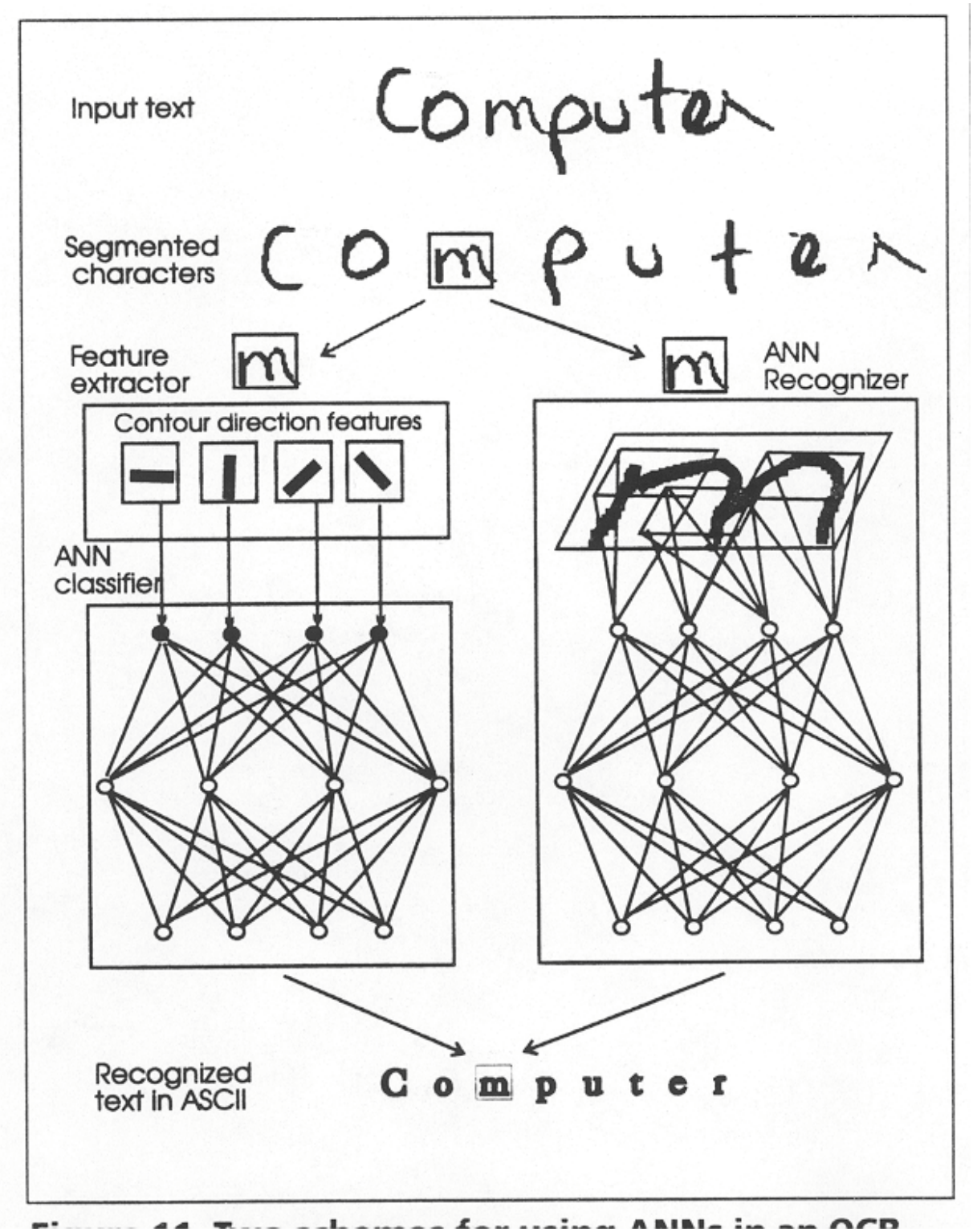


Figure 10. A sample set of characters in the NIST database


Input text
Computer
Segmented characters
Feature extractor
ANN Recognizer
Contour direction features
ANN classifier
Recognized text in ASCII
Computer