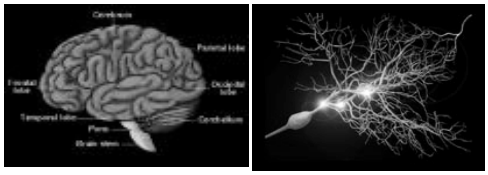
INTRODUCCION A REDES NEURONALES ARTIFICIALES



Héctor Allende
Octubre de 2005

Inteligencia Computacional ? Computational Intelligence

AREA : Soft-Computing

Combinación:

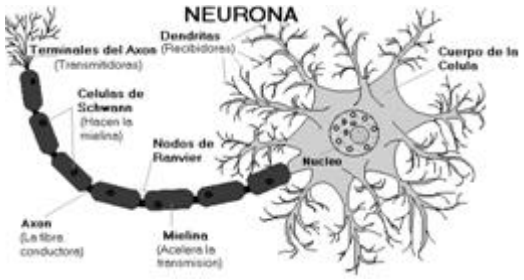
- Ciencias de la Computación
- Neuro-Fisiología
- Teoría del conocimiento y lógica

Construcción Máquinas que aprendan según el Test de Turin

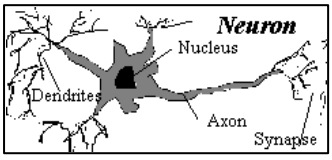
Corteza Cerebral Humana

- Aproximadamente 10^{11} neuronas
- 1000 a 10.000 Synapsis por neurona
- Comunicación tren de impulsos electro-quimicos (mensaje modulado)
- Proceso Cognitivo
  - → tiempo (milisegundos)
  - → Operación **Masiva Paralela**
  - → Secuencial en 100 Etapas

Biológico de neurona Modelo



Célula Nerviosa



Soma:	Información Hereditaria + Plasma + Generación Señales
Dendritas:	Recepción Señales → Impulsos
Axón:	Transmisión de Señales
Sinapsis:	Interfaz Neuronal (Inhibitoria, Excitatoria)

Desarrollo Histórico

- 1943 W.McCulloch, W. Pitts: Modelo ANN (El Perceptrón)
- 1969 Minsky y Papert: El Perceptrón (limitaciones).
- 1982 J. Hopfield: Memoria Asociativa "Redes de Hopfield".
- 1984 T. Kohonen : Redes SOM (Self-Organizing Maps)
- 1986 Rumulhart, Hunton y Williams : redescubren el BPL algoritmo de "back-propagation learning" (Paul Werbor, 1974)
- 1989 K. Hornik, M. Stinchcombe, H. White: Multi-FANN y Aproximación Universal

Red neuronal artificial (ANN)

ANN: Es un sistema dinámico compuesto por redes paralelas y distribuidas de procesadores elementales, con la capacidad de aprender y almacenar "conocimiento"

•Arquitectura

•Interacción

•Función de activación

Aplicaciones de las ANN

- Resolver problemas Complejos (Visión)
- Generalización (Máquinas de Inferencia)
- Establecer Relaciones no evidentes (PR)
- Análisis de sistemas complejos
- Percepción
- Comprensión y Aprendizaje
- Generación de nuevo conocimiento
- Robótica

Aplicaciones de las ANN

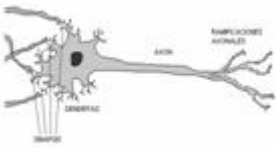
- Ciencias de la Tierra
- Astronomía
- Minería
- Energía
- Economía
- Medicina
- Sociología

Aplicaciones de las ANN

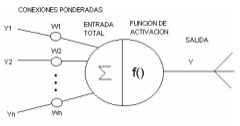
- Clasificación ; Clustering
- Pre-procesamiento de datos
- Reconocimiento de patrones
- Aproximación de funciones
- Predicción de Series de Tiempo
- Optimización Control
- Robótica

ANN y Neuronales Biológicas

Neurona y Conexiones Sinápticas



Procesador Elemental



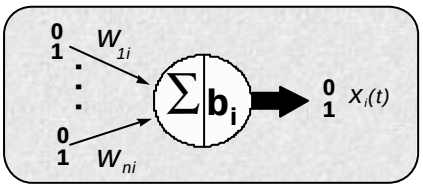
Neuronas: El aprendizaje se produce mediante la variación de la efectividad de las sinapsis, de esta manera cambia la influencia que unas neuronas ejercen sobre otras.

ANN: La regla de aprendizaje usada indica como se ajustan los pesos de las conexiones en función del vector entrada

Analogías

Células Biológicas	**Redes Neuronales Artificiales**
Neuronas	Unidades de proceso
Conexiones Sinápticas	Conexiones Ponderadas
Efectividad de las Sinapsis	Peso de las conexiones
Efecto exitatorio o inhibitorio	Signo del Peso
Estimulo Total	Entrada total Ponderada
Activación → Tasa de disparo	Función de Activación → Salida

Modelo Neuronal: Mc Culloch & Pitts 1943



$$x_i(t) = 1 \left[\sum_{j=1}^{n} w_{ij} x_j(t-1) - b_i \right]$$

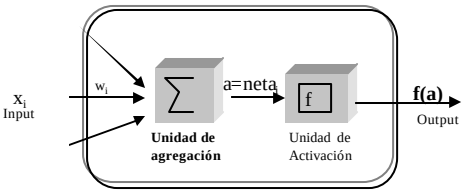
Procesador Elemental.

PE: Es una unidad básica de procesamiento la que posee múltiples entradas y solo una salida.

Cada entrada x_i es ponderada por un factor (peso) w_i y se calcula la suma ponderada de las entradas: $\sum_i w_i x_i = a = neta_i$

Luego es aplicada una transformación mediante la función de activación : salida $= f(a)$

Procesador elemental.



Procesador elemental.

- **ANN Feedforward**: Se construye colocando las neuronas en capas y conectando las salidas de una capa con las entradas de las neuronas de la próxima capa.
- **Capas de una red:**
  - Capa de entrada Zona sensorial (S)
  - Capa de salida Zona de Respuesta (R)
  - Capas ocultas Zona de asociación (A)

ANN: Aprendizaje y Generalización

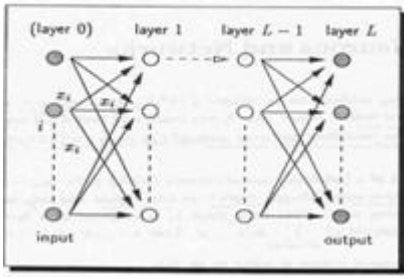
Tipos de Aprendizaje { **Supervisado**, **No - Supervisado**

Tipos de Arquitectura { **FeedForward**, **Single, Multiple**, **Recurrentes**

Tipos de Función de Transición: deterministas, probabilistas

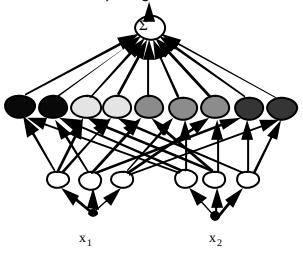
Tipo de Algoritmo de Aprendizaje: BPL, PPL, LM, etc

Feedforward Neural Network



Redes Feedforward

- FANN La capa 0 no realiza procesamiento alguno, solo distribuye las entradas a la capa siguiente



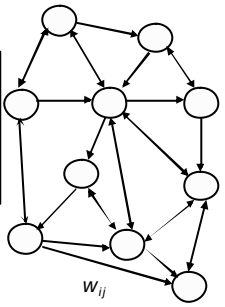
Modelo de Turing

$$x_i(0) \in \{0,1\}$$

$$x_i(t) = \mathbb{1}\left[\sum_{j=1}^{n} w_{ij} x_j(t-1) - b_i\right] \quad \forall\ i=1,\ldots,n$$

$\|w_{ij}\|$ = Matriz de conectividad

(b_i) = vector de umbrales



Neuronas y Redes Simples.

- **ANN Recurrente**: La salida de una neurona es la entrada de neuronas de capas anteriores (feedback).
- Feedback lateral: La salida de una neurona es la entrada de otra neurona en la misma capa.

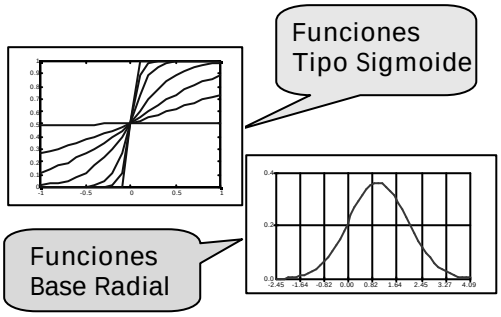
Neuronas y Redes simples.

- **Parámetros de la Red**: Los pesos $\{w_i\}$.
- **Aprendizaje o entrenamiento:** Es el procedimiento mediante el cual los pesos son ajustados.
- **Conjunto de entrenamiento:** Conjunto de datos que consiste vectores de entrada asociado con vectores de salida deseada:$\{(x_i, y_i)\}$.

Neuronas como funciones

- Las neuronas transforman una entrada no acotada x(t) en el tiempo t en una señal de salida acotada f(x(t)).
- La función de activación o función de señal: f
- **Velocidad de la señal**: $\dot{f} = \frac{df}{da}\frac{da}{dt} = f'\dot{a}$

Funciones de Activación



Funciones de activación

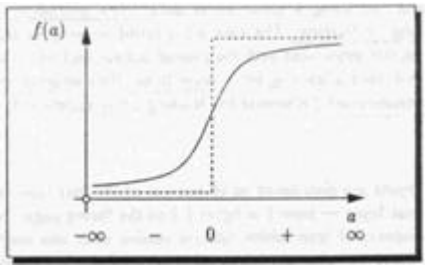
- **Función de activación logística:**

$$f(a) = \frac{1}{1 + e^{-ca}}$$

- Es monótamente creciente para c >0
- Es derivable

$$f' \equiv \frac{df}{da} = cf(1 - f) > 0$$

Funciones de activación



Funciones de activación

- **Tangente hiperbólica**:

$$f(a) = \tanh(ca) = \frac{e^{ca} - e^{-ca}}{e^{ca} + e^{-ca}}$$

donde c>0.

$$f' = c(1 - f^2) > 0$$

Preguntas Abiertas

- Tamaño de las muestras
- Cuántas Neuronas
- Cuantas Capas
- Tipo de Arquitectura
- Tipo de Aprendizaje
- Algoritmos de Aprendizaje
- ¿Cuándo usar ANN

Modelador

Buenas Referencias

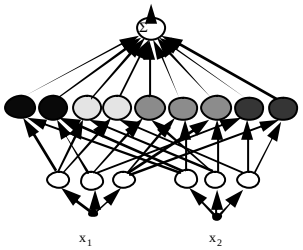
- **C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.**
- FAQ NN: "ftp://ftp.sas.com/pub/neural/FAQ.html"
- **B.D. Ripley: Pattern Recognition and Neural Network Cambridge University Press, 1996**.
- J. Hertz, A. Krogh and R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, 1991

Test de Turing:

"Un computador merece ser llamado inteligente si puede hacer pensar a un ser humano que es otro ser humano"


Alan Turing

Introducción ANN Feedforward



PARTE 2
Backpropagation Learning

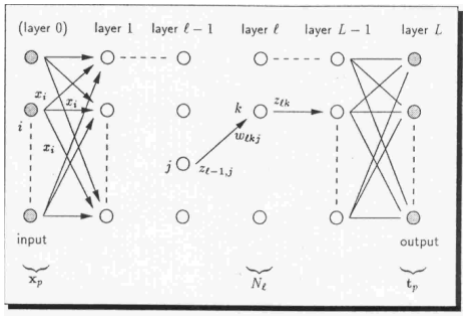
Estructura de la Red

- Capa de entrada: **sensorial**
  - También llamada capa sensorial (capa 0)
  - No existe procesamiento.
  - Su función es distribuir la entrada a la próxima capa del vector de entrada **x**.
- Capas Oculta: **asociativa**
  - Son las capas que están ubicadas entre la capa de entrada y salida.

Estructura de la Red

- Capa de salida: **respuesta**
  - Esta capa proporciona la salida de los datos procesados.
  - Entrega un vector de salida **y**
- Red Feedforward:
  - Cada neurona recibe como entrada las salidas de todas las neuronas de la capa anterior.

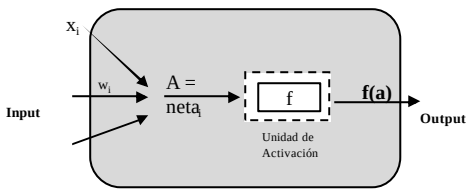
Estructura de la Red



Notación

- $\mathbf{w_{lkj}}$ es el peso por el cual la salida de la neurona j de la capa l-1 contribuye a la entrada de la neurona k de la capa l.
- $\mathbf{x_p}$ es la entrada de entrenamiento p
- $\mathbf{t_p(x_p)}$ es el destino (salida deseada) del vector $\mathbf{x_p}$.
- $\mathbf{z_{oi}} = \mathbf{x_i}$ es el componente i del vector de entrada.
- $\mathbf{N_l}$ número de neuronas de la capa l.
- $\mathbf{z_{lk}}$ es la salida de la neurona j de la capa l.
- **L** es el número de capas.
- **P** es el número de vectores de entrenamiento.
- $\mathbf{\{(x_p, t_p)\}_{p=1,..,P}}$ es el conjunto de aprendizaje

Dinámica de la Red



Función de salida

- Función de activación logística:

$$f(a) = \frac{1}{1+\exp(-ca)}; \quad f: \Re \to (0,1), c > 0, c \text{ constante}$$

$$\frac{df}{da} = \frac{c\exp(-ca)}{[1+\exp(-ca)]^2} = cf(a)[1-f(a)]$$

Ejecución de la Red

$$z_{lk} = f\left(\sum_{j=1}^{N_{l-1}} w_{lkj} z_{l-1,j}\right)$$

– Matriz de pesos:

$$W_l = \begin{pmatrix} w_{l11} & \cdots & w_{l1N_{l-1}} \\ \vdots & \ddots & \vdots \\ w_{lN_l1} & \cdots & w_{lN_lN_{l-1}} \end{pmatrix}$$

– Vector de salida de la capa anterior:

$$z_{l-1}^T = (z_{l-1,1} \quad \cdots \quad z_{l-1,N_{l-1}})$$

– Salida de la capa l

$$z_l^T = f(a_l^T) = (f(a_{l1}) \quad \cdots \quad f(a_{lN_l}))$$

donde $a_l = W_l z_{l-1}$

Aprendizaje de la Red

- El proceso de aprendizaje de la red es supervisado. (Etapa Entrenamiento)
- El aprendizaje involucra ajustar los pesos de manera que el error sea minimizado.
- Uso de los Datos Crudos

Aprendizaje de la Red

- Función de suma de los errores cuadráticos:

$$E(W) \equiv \frac{1}{2}\sum_{q=1}^{N_L}[z_{L_q}(x) - t_q(x)]^2$$

donde z_{L_q} es la salida de la neurona q de la capa de salida

- Observaciones:
  - Suma total de la suma de los errores cuadráticos:

$$E_{tot}(W) \equiv \sum_{p=1}^{P} E(W)$$

Aprendizaje de la Red

- Los pesos de la red W se obtienen paso a paso.
- N_w es el número total de pesos, entonces la función de error: $E: \Re^{N_w} \to \Re$

  es una superficie en el espacio $\Re^{N_w+1}$
- El vector gradiente: $\vec{\nabla} E = \left\{ \frac{\partial E(W)}{\partial w_{lji}} \right\}$

  muestra la dirección del máximo error cuadrático medio. ECM

Aprendizaje de la Red

- **Los pesos son ajustados en tiempos discretos (usando la Regla D):**

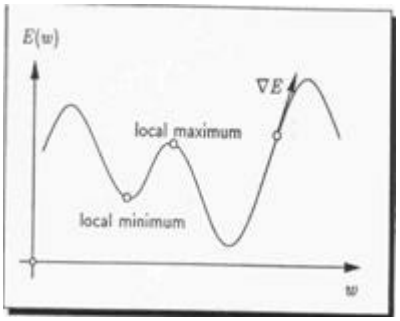
$$w_{lji}(t+1) = w_{lji}(t) - m\left.\frac{\partial E(W)}{\partial w_{lji}}\right|_{W(t)}$$

$$= w_{lji}(t) - m\sum_{p=1}^{P}\left.\frac{\partial E_p(W)}{\partial w_{lji}}\right|_{W(t)}$$

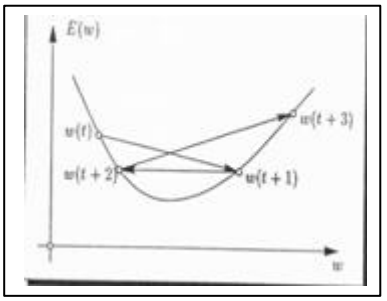
donde $m > 0$ es la constante de aprendizaje.

Matricialmente: $W(t+1) = W(t) - m\nabla E$

Elección del Parámetro μ



Elección del Parámetro μ



El algoritmo de Backpropagation

Previos

1.- Para cada capa (excepto la de entrada), una matriz del gradiente del error se construiría de la siguiente manera:

$$(\nabla E)_l = \begin{pmatrix} \frac{\partial E}{\partial w_{l11}} \cdots \cdots \frac{\partial E}{\partial w_{l1N_{l-1}}} \\ \vdots \qquad \ddots \qquad \vdots \\ \frac{\partial E}{\partial w_{lN_l1}} \cdots \cdots \frac{\partial E}{\partial w_{lN_lN_{l-1}}} \end{pmatrix}, \qquad l = 1,.., L$$

El algoritmo de Backpropagation

2. Para cada capa, excepto la capa L, el gradiente del error con respecto a la salida neuronal se define como: $\nabla_{z_l} E \equiv \left(\frac{\partial E}{\partial z_{l1}} \cdots\cdots \frac{\partial E}{\partial z_{lN_l}} \right), \quad l = 1,.., L-1$

3. El gradiente del error con respecto a la salida de la red z_L es conocido y depende solo de la salida de la red $\{z_L(x_p)\}$ y los targets $\{t_p(x_p)\}$:

$$\nabla_{z_L} E = conocido$$

El algoritmo de Backpropagation

- Entonces considerando la función de error E y la función de activación f y su derivada f ' se calcula el gradiente del error recursivamente

$$\nabla_{z_l} E = W_{l+1}^T [\nabla_{z_{l+1}} E \otimes f'(a_{l+1})] \qquad \text{desde L-1 a 1.}$$

$$(\nabla E)_l = [\nabla_{z_l} E \otimes f'(a_l)] z^T{}_{l-1} \qquad \text{para las capas l=1..L}$$

donde $z_0 \equiv x$

Corolario

- Si la función de activación es la función logística

$$\nabla_{z_L} E = Z_L(x) - t$$

$$\nabla_{z_l} E = c W_{l+1}^T [\nabla_{z_{l+1}} E \otimes Z_{l+1} \otimes (1 - Z_{l+1})],$$

$$(\nabla E)_l = c[\nabla_{z_l} E \otimes Z_l \otimes (1 - Z_l)] Z^T{}_{l-1}$$

donde $z_0 \equiv x$

Criterios de inicialización y parada

- Pesos son inicializados con valores aleatorios U (-1;1) y el proceso de ajuste continúa iterativamente.
- La parada del proceso se realiza por medio de uno de los siguientes criterios:

1.- Elegir un número de pasos fijos.

2.- El proceso de aprendizaje continua hasta que la cantidad: $\Delta w_{lji} = w_{lji(\text{tiempo t+1})} - w_{lji(\text{tiempo t})}$

sea menor que algún valor específico.

3.- El algoritmo se detiene cuando el error total alcanza un mínimo en el conjunto de prueba.

El Algoritmo

- El algoritmo en una aproximación de tiempo discreto.
- La funciones de error y de activación y la condición de parada se asume que son elegidos y fijos.

Procedimiento de ejecución de la Red

1.- La capa de entrada es inicilizada, es decir, la salida de la capa de igual a la entrada x : $z_0 \equiv x$

Para la capas, desde 1 hasta L, hacer: $z_l = f(W_l z_{l-1})$

2.- La salida final de la red es la salida de la última capa es decir , $y \equiv z_L$

El Algoritmo

Procedimiento de Aprendizaje de la red:

1.- Inicializar los pesos con valores aleatorios pequeños. U(-1; 1)

2.- Para el conjunto de entrenamiento (x_p,t_p),

(a) Correr la red para encontrar la activación para todas las neuronas a_l y luego sus derivadas $f'(a_l)$. La salida de la red $y_p \equiv z_L(x_p)=f(a_l)$ es usada en el próximo paso.

El Algoritmo

(b) Usando (y_p,t_p), calcular para la capa L $\nabla_{z_L} E$

(c) Calcular el gradiente del error, para $\nabla_{z_l} E$ usando b-c calcular $(\nabla E)_l$

(d) Actualizar los pesos W de acuerdo a l regla delta

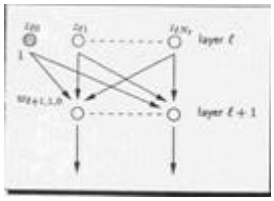
(e) Chequear la condición de parada y parar si se cumple la condición.

SESGO (BIAS)

- Activación Neuronal:Algunos problemas no se pueden resolver con la BN, sin introducir un nuevo parámetro llamado sesgo w_{lk0}

$$z_{lk} = f\left(w_{lk0} + \sum_{j=1}^{N_{l-1}} w_{lkj} z_{l-1,j}\right)$$

Bias



Sesgo (BIAS)

- Salida Neuronal: $\tilde{z}_l^T = (1 \;\; z_{l1} \;\; \cdots \;\; z_{lN_l})$
- Matrices de Pesos:

$$\tilde{W}_l = \begin{pmatrix} w_{l10} & w_{l11} \cdots w_{l1N_{l-1}} \\ \vdots & \vdots \;\; \ddots \;\; \vdots \\ w_{lN_l0} & w_{lN_l1} \cdots w_{lN_lN_{l-1}} \end{pmatrix}$$

$$\Rightarrow z_l = f(a_l) = f(\tilde{W}_l \tilde{z}_{l-1})$$

Sesgo (BIAS)

- Matriz del gradiente del error:

$$(\nabla E)_l = \begin{pmatrix} \frac{\partial E}{w_{l10}} & \frac{\partial E}{w_{l11}} & \cdots & \frac{\partial E}{w_{l1N_{l-1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{w_{lN_l 0}} & \frac{\partial E}{w_{lN_l 1}} & \cdots & \frac{\partial E}{w_{lN_l N_{l-1}}} \end{pmatrix}$$

Backpropagation (+bias)

Si el gradiente del error con respecto a la salida neuronal $\nabla_{z_L} E$ es conocido, y depende sólo de la salida de la red $\{z_L(x_P)\}$ y del target $\{t_p\}$

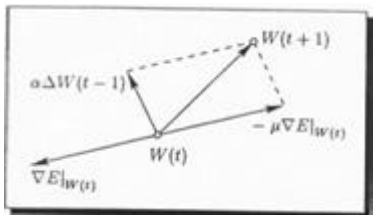
Entonces el gradiente del error $\nabla_{z_l} E$ se calcula recursivamente

$\nabla_{z_l} E = W_{l+1}{}^T [\nabla_{z_{l+1}} E \otimes f'(a_{l+1})]$ para L-1 hasta 1

$(\nabla E)_l = [\nabla_{z_L} E \otimes f'(a_l)] \tilde{z}_{l-1}{}^T$ para las capas l hasta L

donde $z_0 \equiv x$

Algoritmo (Momentum)



Algoritmo (Momentum)

- El algoritmo BPL carece de robustez
- Un procedimiento alternativo que toma en cuenta los atractores en el proceso de aprendizaje es el algoritmo de momentum:

$$\Delta W(t) = W(t+1) - W(t) = -m\nabla E\big|_{W(t)} + a\Delta W(t-1)$$

donde $\alpha \in [0,1)$ es el parámetro de momentum.

- El procedimiento de aprendizaje y ejecución es equivalente al BPL clásico.

Algoritmo (Momentum)

- Otra mejora del algoritmo momentum es la eliminación de puntos planos, i.e. Si la superficie de error es muy plana, entonces $\nabla E \approx \tilde{0}$ y, por lo tanto, $\Delta W \approx \tilde{0}$

Para evitar el problema el calculo del gradiente es llevado de la siguiente manera:

$\nabla_{z_l} E = W_{l+1}{}^T [\nabla_{z_{l+1}} E \otimes f'(a_{l+1})]$ desde L-1 hasta 1

$(\nabla E)_{l,pseudo} = [\nabla_{z_l} E \otimes \left| f'(a_l) + c_f \hat{1} \right|] z^T{}_{l-1}$ desde l=1,..,L

Algoritmo: Momentum

Eliminación de puntos planos:

- c_f es la constante de eliminación de puntos planos.
- Los términos correspondientes de los pesos del gradiente del error cercanos a la capa de entrada son más pequeños que aquellos ubicados en la capa de salida. Por lo tanto un efecto de c_f es la aceleración de la adaptación de los pesos en capas cercanas a la entrada.

Algoritmo: Backpropagation Adaptivo

- Ideas del algoritmo:
  - Si la pendiente de la superficie de error es suave, entonces un parámetro de aprendizaje grande puede ser usado para acelerar el aprendizaje en las áreas planas.
  - Si la pendiente de la superficie de error es abrupta, entonces un pequeño parámetro de aprendizaje debe ser usado para no saltar el mínimo.

Algoritmo: Backpropagation Adaptivo

- Se asignan valores de aprendizaje individual a cada peso basado en el comportamiento previo. Entonces la constante de aprendizaje μ se convierte en una matriz. $[w_{lji}]$
- La razón de aprendizaje aumenta si el gradiente en la mantiene su dirección en los últimos dos pasos, en caso contrario lo disminuye:

$$m_{lji}(t) = \begin{cases} I_{m_{lji}}(t-1) & \text{si } \Delta w_{lji}(t)\Delta w_{lji}(t-1) \geq 0 \\ D_{m_{lji}}(t-1) & \text{si } \Delta w_{lji}(t)\Delta w_{lji}(t-1) < 0 \end{cases}$$

donde $I \geq 1$ es el factor de aumento y $D \in (0,1)$ es el factor de disminución.

Otras Mejoras del Algoritmo BPL

- SuperSAB (Super Self-Adapting Backpropagation):
  - Es una combinación entre momentum y backpropagation adaptivo.
  - Usa backpropagation adaptivo para los términos w_{lij} que continúan el movimiento en la misma dirección y momentum para las otras.

Algoritmo (Momentum)

- El algoritmo BPL carece de robustez
- Un procedimiento que toma en cuenta los atractores del proceso de aprendizaje es el algoritmo de momentum:

$$\Delta W(t) = W(t+1) - W(t) = -m\nabla E\Big|_{W(t)} + a\Delta W(t-1)$$

donde $\alpha \in [0,1)$ es el parámetro de momentum.

- El procedimiento de aprendizaje y ejecución es equivalente al BPL clásico la forma antes descrita.

Mejoras del Algoritmo (Super SAB)

- **Si** $\Delta w_{lji}(t)\Delta w_{lji}(t-1) \geq 0$ **entonces**:

$$m_{lji}(t) = Im_{lji}(t-1)$$

$$\Delta w_{lji}(t+1) = -m_{lji}(t)\frac{\partial E}{\partial w_{lji}}\Big|_{W(t)}$$

- **Si** $\Delta w_{lji}(t)\Delta w_{lji}(t-1) < 0$ **entonces:**

$$m_{lji}(t) = Dm_{lji}(t-1)$$

$$\Delta w_{lji}(t+1) = -m_{lji}(t)\frac{\partial E}{\partial w_{lji}}\Big|_{W(t)} - a\Delta w_{lji}(t)$$

Mejoras del Algoritmo(Super SAB)

- **En notación matricial:**

$$m(t) = \left\{(I-D)sign\left[sign(\Delta W(t)\bullet\Delta W(t-1)) + \tilde{1}\right] + D\tilde{1}\right\}\bullet m(t-1)$$

$$\Delta W(t+1) = -m(t)\bullet\nabla E - a\Delta W(t)\bullet\left\{\tilde{1} - sign\left[sign(\Delta W(t)\bullet\Delta W(t-1)) + \tilde{1}\right]\right\}$$

Algoritmo: Backpropagation Adaptivo

- **En forma matricial:**

$$m(t) = \left\{(I - D)sign\ \left\lfloor sign(\Delta W(t) \bullet \Delta W(t-1)) + \tilde{1}\right\rfloor + D\tilde{1}\right\} \bullet m(t-1)$$