



Capítulo 4

Redes- FANN

Redes de Retropropagación (Backpropagation Network)

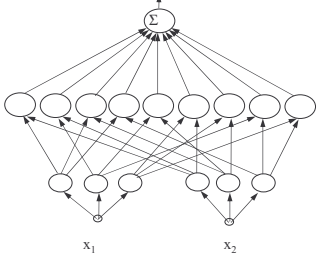
FANN. Son las ANN más simples en términos de diseño

Profesor: Héctor Allende Redes Neuronales Artificiales 1






Redes Feedforward

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





Profesor: Héctor Allende Redes Neuronales Artificiales 2

Estructura de la Red

- Capa de entrada: (sensorial)
 - También llamada capa 0 o sensorial
 - 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.



Profesor: Héctor Allende Redes Neuronales Artificiales 3

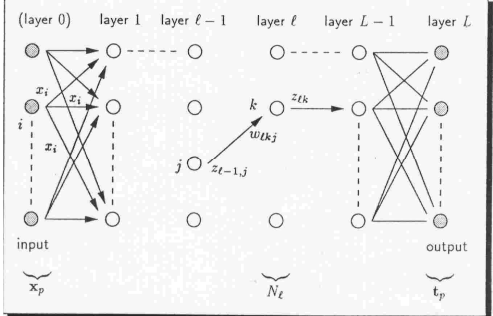
Estructura de la Red

- 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.



Profesor: Héctor Allende Redes Neuronales Artificiales 4

Estructura de la Red



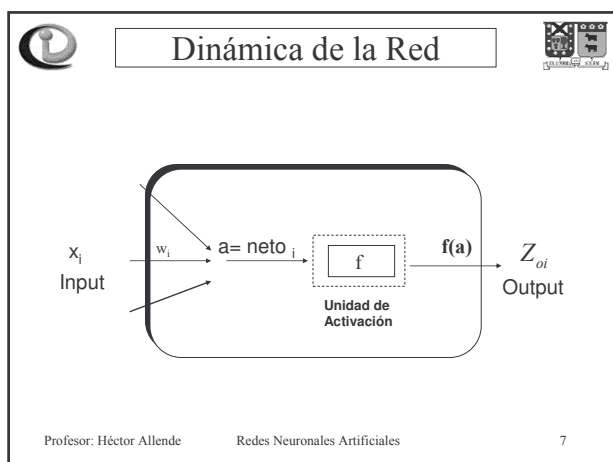
Profesor: Héctor Allende Redes Neuronales Artificiales 5

Notación

- w_{lkj} :peso (parámetro) 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 .
- x_p :vector de entrada de entrenamiento p
- $t_p(x_p)$:respuesta (salida deseada) del vector x_p .
- $Z_{oi} \equiv x_i$:componente i del vector de entrada.
- N_l :número de neuronas de la capa l .
- z_{lk} :salida de la neurona j de la capa l .
- L :número de capas.
- P :número de vectores de entrenamiento.
- $\{(x_p, t_p)\}_{p=1, \dots, P}$:conjunto de aprendizaje

Profesor: Héctor Allende Redes Neuronales Artificiales 6



HISTORIA DE LAS FANN

- Linear separability of the perceptron (1957)
- Manfred Minsky (1969)
- **Rumelhart D.E. and McClelland (1986)**
Parallel Distributed Processing
 (Chapter: Backpropagation)

"Multilayer Perceptron (MLP)"

Profesor: Héctor Allende Redes Neuronales Artificiales 8 © cm

Principle of universal approximation

FANN: $\mathbb{R}^n \rightarrow \mathbb{R}^m$

Theorem: Any continuous function defined on a compact subset of the reals may be realized with a feedforward neural network with one *finite* hidden layer, using non-constant and bounded activation functions, with as much accuracy as desired.

Notice: The above theorem is an *Existence Theorem*. It gives no information on the size of the neural network neither on the activation functions.

Profesor: Héctor Allende Redes Neuronales Artificiales 9

Función monótona

$a = f(W, X, w_0)$

- Función de activación logística:

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

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

Profesor: Héctor Allende Redes Neuronales Artificiales 10

Función no monótona

$$Q(\mathbf{x}) = \sum_{i=1}^n \left((x_i w_i - w_0)^2 / 2\sigma^2 \right)$$

$$f(Q(\mathbf{x})) = \exp(-Q(\mathbf{x}))$$

Profesor: Héctor Allende Redes Neuronales Artificiales 11 © cm

Principle of universal approximation

En el caso de Funciones de activación **monotonas** revisar

Hornik K., Stinchcombe M., White H., (1989): Multilayer Feedforward Networks are universal approximators. *Neural Networks* 2, 359–366
 Funahashi K.I., (1989): On the approximate realization of continuous mappings by neural networks. *Neural Networks* 2, 183–192

En el caso de Funciones de activación **no monotonas** revisar

White H. (1992): *Artificial Neural Networks. Approximation and Learning Theory*. Basil Blackwell, Oxford.

Profesor: Héctor Allende Redes Neuronales Artificiales 12

Ejecución de la Red

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

- Matriz de pesos:
$$W_l = \begin{pmatrix} w_{l11} & \dots & w_{l1N_{l-1}} \\ \vdots & \ddots & \vdots \\ w_{lN_l1} & \dots & w_{lN_lN_{l-1}} \end{pmatrix}$$
- Vector de salida de la capa anterior:
$$z_{l-1}^T = (z_{l-1,1} \dots z_{l-1,N_{l-1}})$$
- Salida de la capa L $z_l^T = f(a_l^T) = (f(a_{l1}) \dots f(a_{lN_l}))$
donde $a_l = W_l z_{l-1}$

Profesor: Héctor Allende Redes Neuronales Artificiales 13

Proceso de Aprendizaje

- El proceso de aprendizaje de la red FANN es supervisado. (Etapa Entrenamiento)
- El aprendizaje involucra ajustar los pesos de manera que el error sea minimizado (Estimación mínimo cuadrático de parámetros)
- Uso de los Datos Crudos (sin corrección de outliers)

Profesor: Héctor Allende Redes Neuronales Artificiales 14

Proceso de Aprendizaje

- Función de suma de los errores cuadráticos:
$$E(W) \equiv \frac{1}{2} \sum_{q=1}^{N_q} [z_{l_q}(x) - t_q(x)]^2$$

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

Profesor: Héctor Allende Redes Neuronales Artificiales 15

Proceso de Aprendizaje

- Los pesos de la red W se computan iterando el algoritmo BPL.
- N_w :número total de pesos, entonces la función de error:
$$E : \mathcal{R}^{N_w} \rightarrow \mathcal{R}$$

Genera una superficie en el espacio \mathcal{R}^{N_w+1}

- El vector gradiente: $\nabla E = \left\{ \frac{\partial E(W)}{\partial w_{ji}} \right\}$

nos muestra la dirección del máximo cambio en el error cuadrático medio. ECM

Profesor: Héctor Allende Redes Neuronales Artificiales 16

Proceso de Aprendizaje

- Los pesos son ajustados en tiempos discretos (usando la Regla Δ):
$$w_{lji}(t+1) = w_{lji}(t) - \mu \left. \frac{\partial E(W)}{\partial w_{lji}} \right|_{W(t)}$$

$$= w_{lji}(t) - \mu \sum_{p=1}^P \left. \frac{\partial E_p(W)}{\partial w_{lji}} \right|_{W(t)}$$
- donde $\mu > 0$ es la constante de aprendizaje.

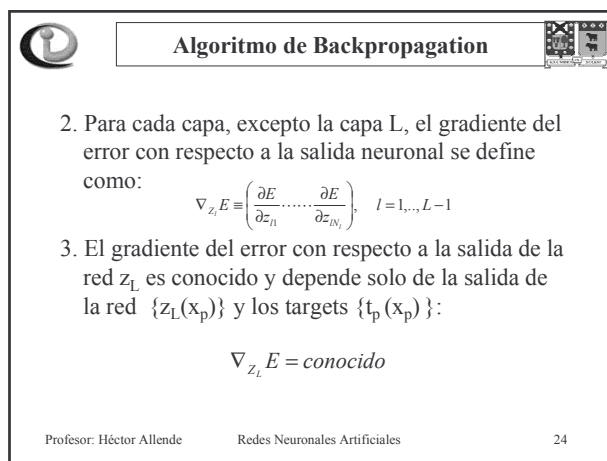
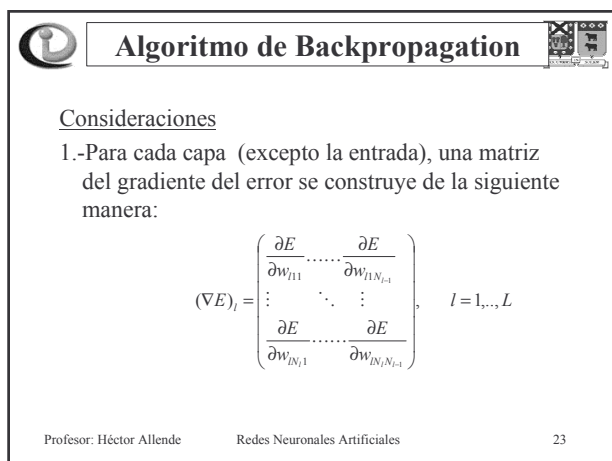
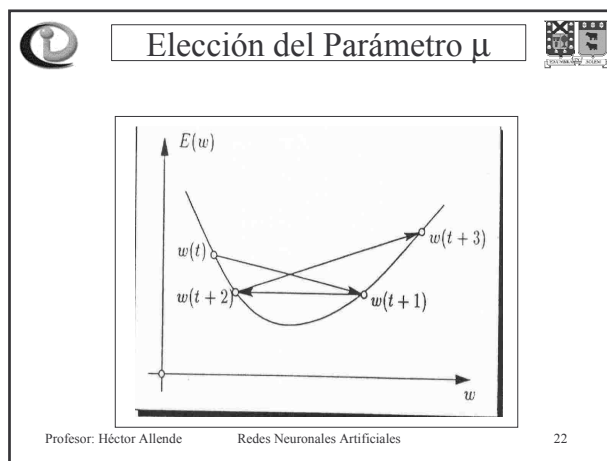
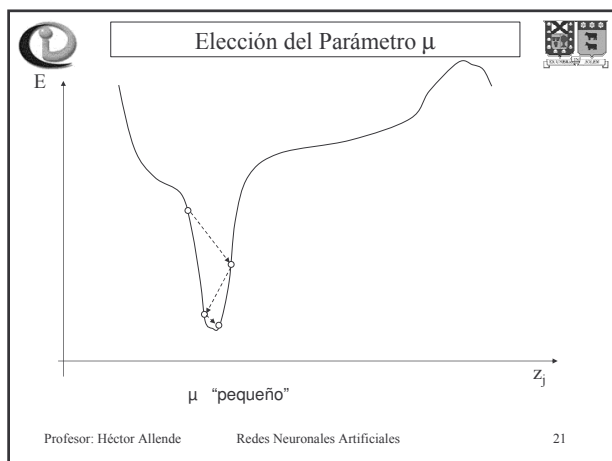
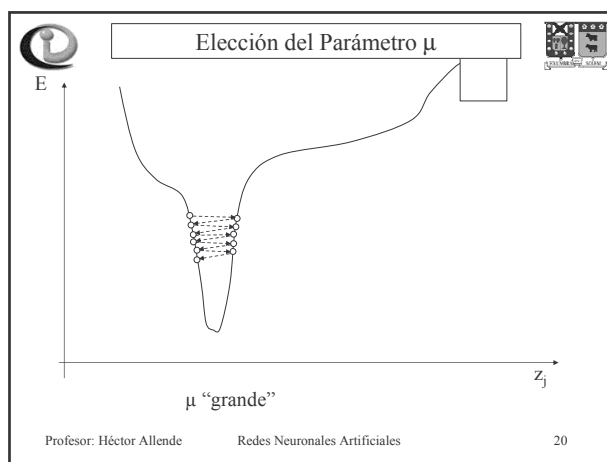
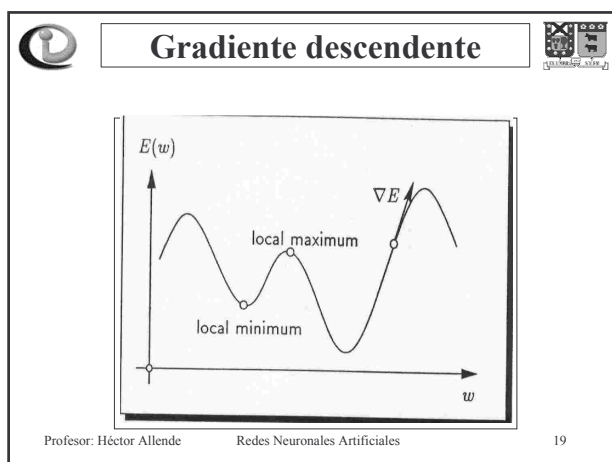
En notación matricial: $W(t+1) = W(t) - \mu \nabla E$


Profesor: Héctor Allende Redes Neuronales Artificiales 17

Gradiente descendente


$z_j(t+1) = z_j(t) - \Gamma \frac{\partial E}{\partial z_j}$

Profesor: Héctor Allende Redes Neuronales Artificiales 18





Algoritmo de Backpropagation



- Entonces considerando la función de error E y la función de activación f y su respectiva derivada f'


El gradiente del error puede ser expresado recursivamente de acuerdo a las expresiones:

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


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

donde $z_0 \equiv x$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 25



Corolario



- Si la función de activación es la función logística


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

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


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

donde $z_0 \equiv x$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 26




Criterios de Inicialización y Parada




- Pesos w son inicializados con valores aleatorios pequeños (-1;1) y el proceso de ajuste continúa iterativamente.
- La parada del proceso de aprendizaje puede ser llevado a cabo por medio de uno de los siguientes criterios:
 - Elegir un número de pasos fijos.
 - El proceso de aprendizaje continua hasta que la cantidad:

$$\Delta w_{ji} = w_{ji}(\text{tiempo}+1) - w_{ji}(\text{tiempo}t)$$
 este por debajo algún valor específico.
 - El proceso se detiene, cuando el error total alcanza un mínimo en el conjunto de testeo.

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 27



Algoritmo BPL




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


Procedimiento de ejecución de la Red

- La capa de entrada es inicializada, es decir, la salida de la capa es igual a la entrada $x : z_0 \equiv x$
- Para todas las capas, desde 1 hasta L, hacer: $z_l = f(W_l z_{l-1})$
- La salida final de la red es la salida de la última capa es decir, $y \equiv z_L$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 28




Algoritmo BPL




Procedimiento de Aprendizaje de la red:

- Inicializar los pesos con valores aleatorios pequeños. $U(-1; 1)$
- Para todo conjunto de entrenamiento (x_p, t_p) , tan grande como la condición de parada lo permita:
 - Correr la red para encontrar la activación para todas las neuronas a_i y luego sus derivadas $f'(a_i)$. La salida de la red $y_p \equiv z_L(x_p) = f(a_i)$ es usada en el próximo paso.

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 29



Algoritmo BPL



- Usando (y_p, t_p) , calcular para la capa L $\nabla_{z_L} E$
- Calcular el gradiente del error, para $\nabla_{z_l} E$ usando b-c calcular $(\nabla E)_l$
- Actualizar los pesos W de acuerdo a la regla delta.
- Chequear la condición de parada y parar si se cumple la condición.

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 30

BIAS

- Activación Neuronal: muchos problemas no se pueden resolver con la BPL, sin introducir un nuevo parámetro llamado sesgo w_{ik0}

$$z_{ik} = f\left(w_{ik0} + \sum_{j=1}^{N_{l-1}} w_{ikj} z_{l-1,j}\right)$$

Bias

Profesor: Héctor Allende Redes Neuronales Artificiales 31

Sesgo (BIAS)

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

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

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

Profesor: Héctor Allende Redes Neuronales Artificiales 32

Sesgo (BIAS)

- Matriz del gradiente del error:

$$(\nabla E)_l = \begin{pmatrix} \frac{\partial E}{\partial w_{l10}} & \frac{\partial E}{\partial w_{l11}} & \dots & \frac{\partial E}{\partial w_{l1N_{l-1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{lN_l0}} & \frac{\partial E}{\partial w_{lN_l1}} & \dots & \frac{\partial E}{\partial w_{lN_lN_{l-1}}} \end{pmatrix}$$

Profesor: Héctor Allende Redes Neuronales Artificiales 33

BPL con bias

- Teorema:** Si el gradiente del error con respecto a la salida neuronal $\nabla_{z_L} E$ es conocida, y depende sólo de la salida de la red $\{z_L(x_p)\}$ y del target $\{t_p\}$:
 $\nabla_{z_L} E = \text{conocido}$
entonces el gradiente del error puede ser calculado recursivamente de acuerdo a las siguientes expresiones:
 $\nabla_{z_l} E = W_{l+1}^T [\nabla_{z_{l+1}} E \otimes f'(a_{l+1})]$ 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 $z_0 \equiv x$

Profesor: Héctor Allende Redes Neuronales Artificiales 34

Algoritmo: Momentum

- El algoritmo BPL carece de robustez
- Un procedimiento que toma en cuenta las atracciones en el proceso de aprendizaje es el algoritmo de momentum:

$$\Delta W(t) = W(t+1) - W(t) = -\mu \nabla E|_{W(t)} + \alpha \Delta W(t-1)$$


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

- El procedimiento de aprendizaje y ejecución es equivalente a la forma antes descrita.


Profesor: Héctor Allende Redes Neuronales Artificiales 35

Algoritmo: Momentum

Profesor: Héctor Allende Redes Neuronales Artificiales 36



Algoritmo: Momentum




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

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


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

$$(\nabla E)_{l, pseudo} = [\nabla_{z_l} E \odot [f'(a_l) + c_f \hat{1}]] z_{l-1}^T \quad \text{para las capas } l=1, \dots, L$$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 37




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.

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 38




Algoritmo Momentum (Modificado)




- Adaptación de los pesos con 2 pasos:

$$\Delta W(t) = -\mu \nabla E|_{W(t)} - \alpha \mu \nabla E|_{W(t-1)} + \alpha^2 \Delta W(t-2)$$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 39




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.

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 40



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. $[\mu_{ji}]$
- La razón de aprendizaje aumenta si el gradiente mantiene su dirección en los últimos dos pasos, en caso contrario lo disminuye:


$$\mu_{ji}(t) = \begin{cases} I \mu_{ji}(t-1) & \text{si } \Delta w_{ji}(t) \Delta w_{ji}(t-1) \geq 0 \\ D \mu_{ji}(t-1) & \text{si } \Delta w_{ji}(t) \Delta w_{ji}(t-1) < 0 \end{cases}$$

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

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 41



Algoritmo: Backpropagation Adaptivo



- En forma matricial:

$$\{(I - D) \text{sign}[\text{sign}(\Delta W(t) \bullet \Delta W(t-1)) + \tilde{1}] + D \tilde{1}\} \bullet \mu(t-1)$$

Profesor: Héctor Allende
 Redes Neuronales Artificiales
 42

Algoritmo Super-SAB

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

Profesor: Héctor Allende Redes Neuronales Artificiales 43

Algoritmo Super-SAB

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

$$\mu_{ji}(t) = I\mu_{ji}(t-1)$$

$$\Delta w_{ji}(t+1) = -\mu_{ji}(t) \left. \frac{\partial E}{\partial w_{ji}} \right|_{W(t)}$$
- Si $\Delta w_{ji}(t)\Delta w_{ji}(t-1) < 0$ entonces:

$$\mu_{ji}(t) = D\mu_{ji}(t-1)$$

$$\Delta w_{ji}(t+1) = -\mu_{ji}(t) \left. \frac{\partial E}{\partial w_{ji}} \right|_{W(t)} - \alpha \Delta w_{ji}(t)$$

Profesor: Héctor Allende Redes Neuronales Artificiales 44

Algoritmo Super-SAB

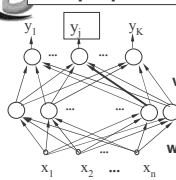
- En notación matricial:

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

$$\Delta W(t+1) = -\mu(t) \bullet \nabla E - \alpha \Delta W(t) \bullet \{ \tilde{1} - \text{sign}[\text{sign}(\Delta W(t) \bullet \Delta W(t-1)) + \tilde{1}] \}$$

Profesor: Héctor Allende Redes Neuronales Artificiales 45

"R-prop": Un algoritmo eficiente de Gradiente



$$v_{ij}(t+1) = v_{ij}(t) - \Gamma \left. \frac{\partial E}{\partial v_{ij}} \right|_{W(t)}$$

$$v_{ij}(t+1) = v_{ij}(t) - \left[\Gamma_{ij} \left. \frac{\partial E}{\partial v_{ij}} \right|_{W(t)} \right]$$

Let $\Omega = \frac{\partial E^{(t-1)}}{\partial v_{ij}} \cdot \frac{\partial E^{(t)}}{\partial v_{ij}}$ for each v_{ij} do the following:

```

begin   if  $\Omega > 0$  then  $\Gamma_{ij}^{(t)} = \min(\eta^+ \Gamma_{ij}^{(t-1)}, \Gamma_{\max})$ 
        else if  $\Omega < 0$  then  $\Gamma_{ij}^{(t)} = \max(\eta^- \Gamma_{ij}^{(t-1)}, \Gamma_{\min})$ 
        else  $\Gamma_{ij}^{(t)} = \Gamma_{ij}^{(t-1)}$ ;
end
```

(1) // $0 < \eta^- < 1 < \eta^+$

Profesor: Héctor Allende Redes Neuronales Artificiales 46

Algoritmo Super-SAB

if $\Omega \geq 0$ then $\Delta v_{ij}^{(t)} = -\text{sign} \left(\left. \frac{\partial E^{(t)}}{\partial v_{ij}} \right|_{W(t)} \right) \Gamma_{ij}^{(t)}$ (2)

where $\text{sign}(x) = 1, 0$ or -1 if $x > 0, x = 0$ or $x < 0$, respectively

if $\Omega < 0$ then $\Delta v_{ij}^{(t)} = -\Delta v_{ij}^{(t-1)}$ and $\frac{\partial E^{(t)}}{\partial v_{ij}} = 0$ (3)

($\frac{\partial E^{(t)}}{\partial v_{ij}}$ is set to 0 to avoid an up-date of the learning rate in the next iteration, after eq. (1))

Finally: $v_{ij}^{(t+1)} = v_{ij}^{(t)} + \Delta v_{ij}^{(t)}$



Similarly for all w_{ij}

Profesor: Héctor Allende Redes Neuronales Artificiales 47

Otros métodos de Entrenamiento FANN

- Métodos de segundo Orden (Levenberg – Marquardt)
- Filtros de Kalman extendidos
- Métodos de Kernel
- Métodos Evolutivos en FANN
- Aprendizaje constructivo Incremental



Profesor: Héctor Allende Redes Neuronales Artificiales 48

Campos de Aplicación

- Clasificación y Reconocimiento de Entidades (Pattern)
- Aproximación de funciones
- Pronóstico (Forecast)
- Modelado (Learning a model for a system from performance data)
- usw

Profesor: Héctor Allende Redes Neuronales Artificiales 49

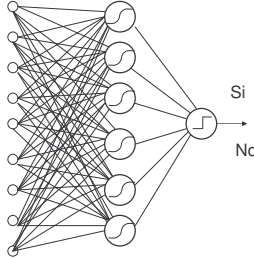



Real Example (C. Serrano, B. Martin del Brío)

The Bank crisis in Spain 1977-85

Inputs: (relevant financial data)

- Activo circulante / Activo total
- Activo circulante – Caja / A.T.
- Activo circulante / Deudas
- Reservas / Deudas
- Beneficio neto / A.T.
- Beneficio neto / Fondos propios
- Beneficio neto / Deudas
- Coste de ventas / Ventas
- Cash flow / Deudas



Profesor: Héctor Allende Redes Neuronales Artificiales 50

