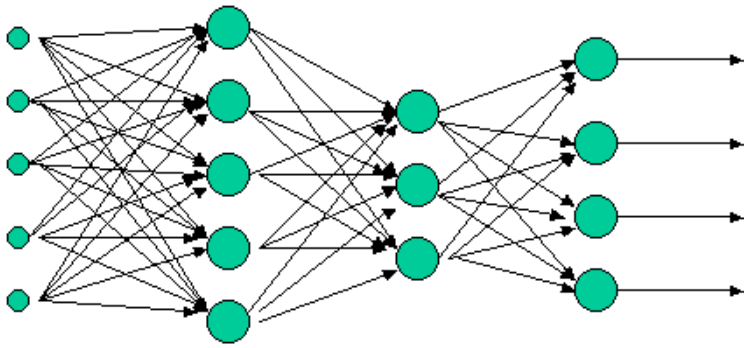
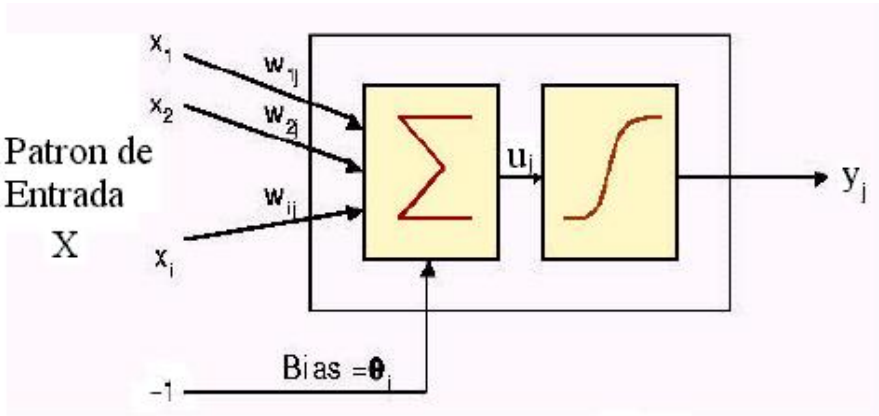
Red Perceptrón Multicapa

- Redes de apenas una capa pueden solo representar funciones linealmente separables.
- Redes de múltiples capas solucionan esa restricción.
- El desarrollo del algoritmo **Back-Propagation** fue uno de los motivos para el resurgimiento del área de redes neuronales.



El gran desafío fue encontrar un algoritmo de aprendizaje para actualizar los pesos de las capas intermediarias

• **Idea Central**: Los errores de las neuronas de la capa de salida (conocidos por el algoritmo de aprendizaje supervisado) son **retro-propagados** para las capas intermedias



Características

- Regla de propagación $u_i(t) = \sum_{j=1}^{N} x_j w_{ij} + \theta_j$
- Función de activación: Función no-lineal diferenciable en todos los puntos;
- Topología: Múltiplas capas;
- Algoritmo de Aprendizaje: Supervisado;
- Valor de Entrada / Salida: Binarias o Continuos.

PROCESO DE APRENDIZJE

- Proceso de aprendizaje mediante la minimización del error cuadrático por el método del gradiente descendiente $\Delta w_{ij} = -\frac{\eta \delta E}{\delta w_{ij}}$
- Cada peso sináptico *i* de la neurona *j* es actualizado proporcionalmente al negativo de la derivada parcial del error de esta neurona con relación al peso.
- Donde el error cuadrático de la neurona *j* es definido por:

$$E_j = \frac{1}{2}(d_j - y_j)^2$$

donde *d* es el valor deseado de la salida para la respectiva neurona
y es la salida de la neurona *j*.

- Se debe minimizar el error de todas las neuronas de la capa de salida para todos los patrones

$$E_{SSE} \rightarrow Sum\ of\ Squared\ \ Errors$$

$$E_{SSE} = \frac{1}{2}\sum_p \sum_j \left(d_{pj} - y_{pj}\right)^2$$

d_{pj} es el valor deseado de salida para el patrón *p* para la neurona *j*

y_{pj} es la salida de la neurona *j* cuando es presentado el patrón *p*

- Calcular Δw_{ij}

$$\Delta w_{ij} = -\eta \frac{\delta E_p}{\delta w_{ij}} = -\eta \frac{\delta E_p}{\delta u_j}\frac{\delta u_j}{\delta w_{ij}}$$

$$e_j = -\frac{\delta E_p}{\delta u_j} \qquad u_j(t) = \sum_{i=1}^{N} x_i w_{ij} + \theta_j$$

$$\Delta w_{ij} = \eta\, e_j\, y_j$$

- Calcular e_j depende de la capa donde se encuentre la neurona

$$e_j = -\frac{\delta E_p}{\delta u_j} \quad \Rightarrow \quad \frac{\delta E_p}{\delta u_j} = \frac{\delta E_p}{\delta y_j}\frac{\delta y_j}{\delta u_j}$$

- Si *j* pertenece a la capa de salida entonces:

$$\frac{\delta E_p}{\delta u_j} = \frac{\delta E_p}{\delta y_j}\frac{\delta y_j}{\delta u_j}$$

$$Si\ \ E_p = \sum_j \frac{1}{2}(d_j - y_j)^2 \ \Rightarrow \frac{\delta E_p}{\delta y_j} = \left[\frac{2*1}{2}\left(d_j - y_j\right)*(-1)\right]$$

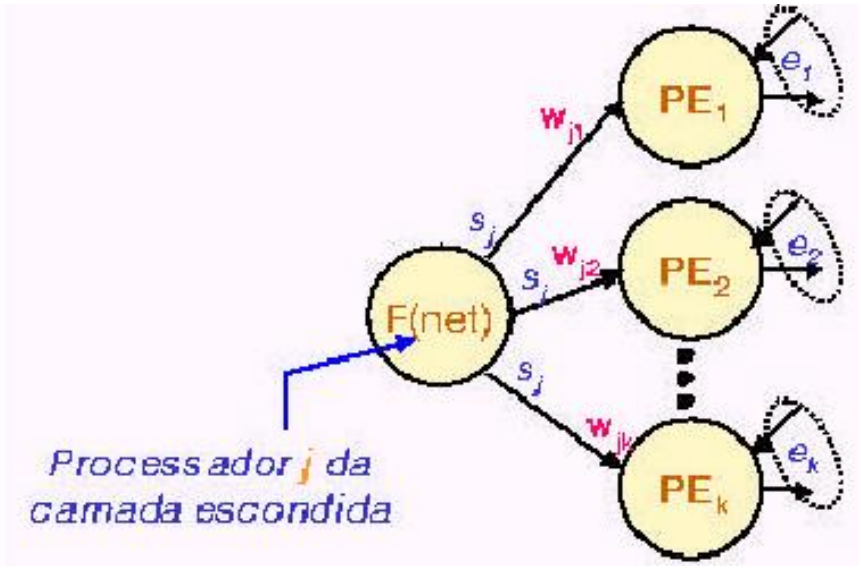
$$Si\ y_j = F(u_j) \ \Rightarrow \ F'(u_j)$$

$$e_j = -\left[-\left(d_j - y_j\right)\right]* F'(u_j) = \left(d_j - y_j\right)F'\left(u_j\right)$$

- Si *j* pertenece a la capa escondida

$$\frac{\delta E}{\delta u_j} = \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta u_j}$$

  - Dado que en el aprendizaje supervisado solo se conoce el error de la capa de salida (e_j)
  - El error de la capa de salida esta en función del potencial interno de la neurona (u_j)
  - u_j depende de los estados de activación de las neuronas de la capa anterior (y_i) y de los pesos de las conexiones de las conexiones (w_{jk}).
  - Por lo tanto y_j de una capa escondida afecta en mayor o menor grado el error de todos, los procesadores de la capa subsiguiente.



- Error de una capa escondida esta dada por

$$e_j = \frac{\delta E}{\delta u_j} = \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta u_j} \quad Si\ y_j = F(u_j) \Rightarrow F'(u_j)$$

$$\frac{\delta E_p}{\delta y_j} = \frac{\delta}{\delta y_j}\left[\frac{1}{2}\sum_k (d_k - y_k)^2\right] = \left[2 * \frac{1}{2}\sum_k (d_k - y_k) * (-1)\frac{\delta y_k}{\delta y_j}\right] = -\sum_k (d_k - y_k)\frac{\delta y_k}{\delta y_j}$$

$$\frac{\delta y_k}{\delta y_j} = \frac{\delta y_k}{\delta u_k}\frac{\delta u_k}{\delta y_j} = F'(u_k)\frac{\partial u_k}{\partial y_j}$$

$$\frac{\partial u_k}{\partial y_j} = \frac{\partial \sum_k w_{jk}x_k + \theta_k}{\partial y_j} = w_{jk}$$

$$-\sum_k (d_k - y_k)\frac{\delta y_k}{\delta y_j} = -\sum_k (d_k - y_k)F'(u_k)w_{jk}$$

$$e_j = -\left[-\sum_k e_k w_{jk}\right] * F'(u_j) = F'(u_j) * \sum_k e_k w_{jk}$$

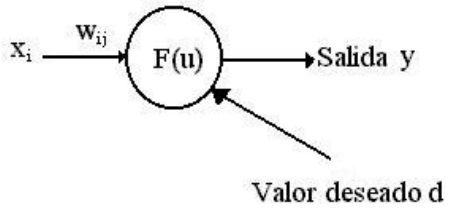
Resumiendo después del calculo de la derivada se tiene que:

$$\Delta w_{ij} = \eta x_i e_j$$

donde x_i es la entrada a la neurona j y e_j es el error de la neurona j.

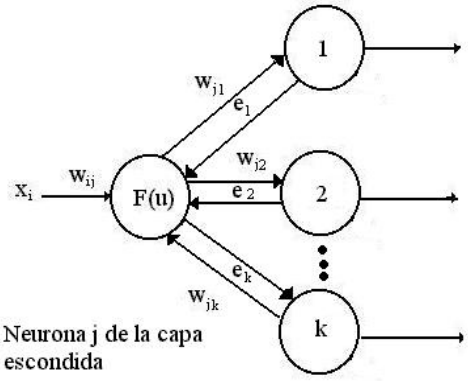
- Si la neurona corresponde a la salida:

$$e_j = (d_j - y_j)\frac{\partial F(u)}{\partial u}$$

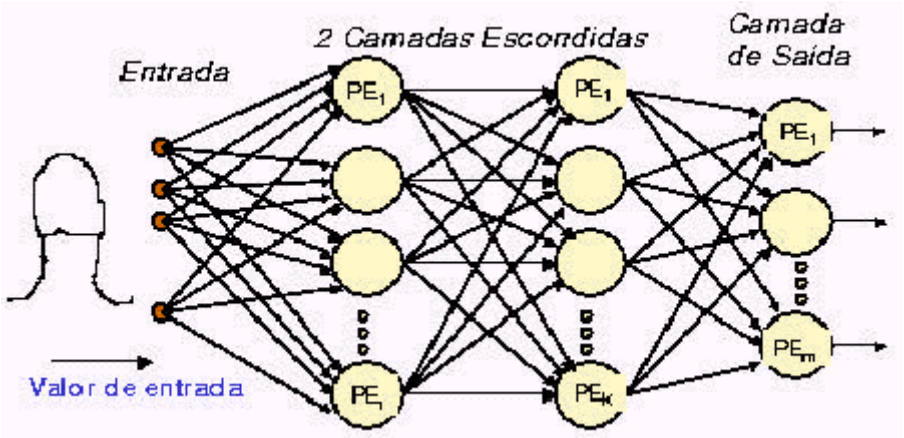


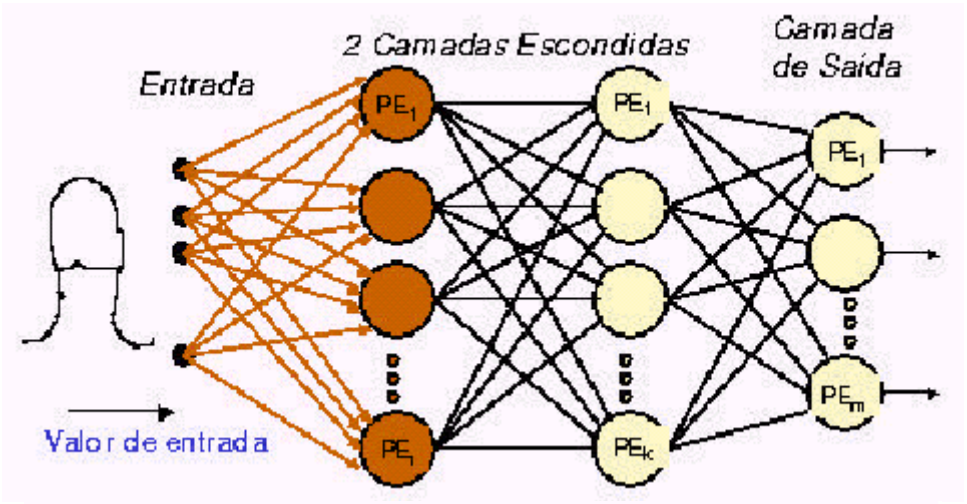
- Si la Neurona pertenece a la capa escondida:

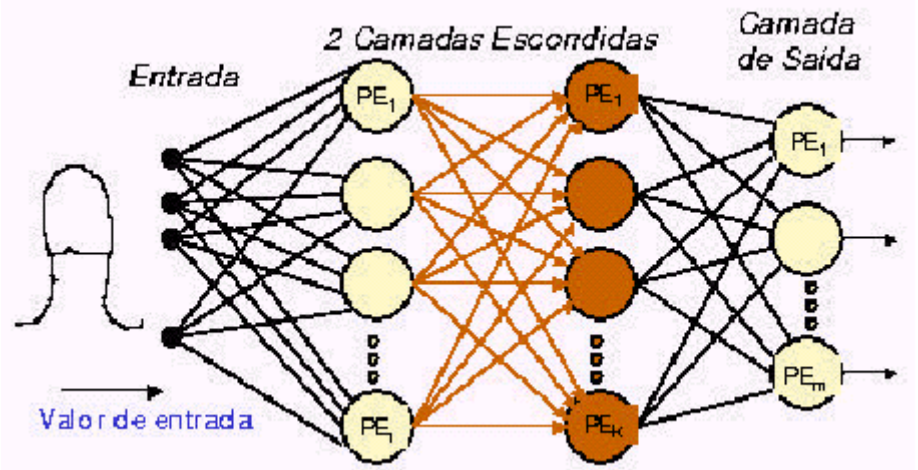
$$e_j = \left(\sum_k e_k w_{jk}\right)\frac{\partial F(u)}{\partial u}$$

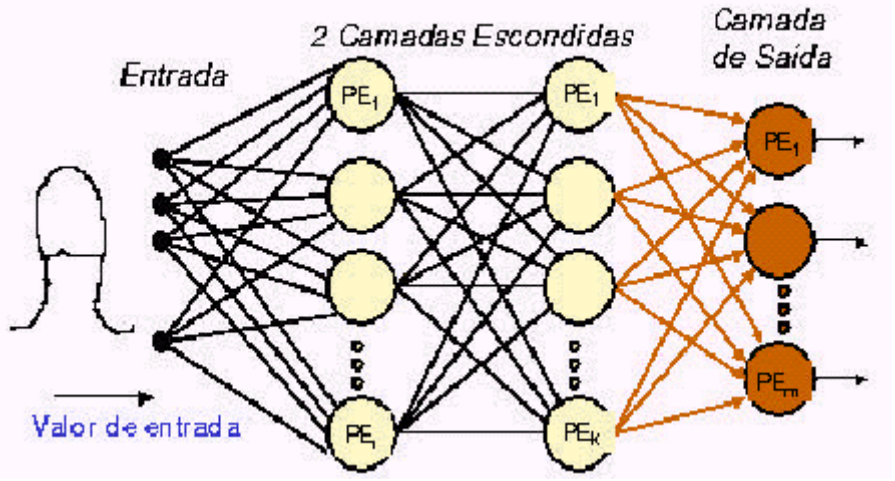


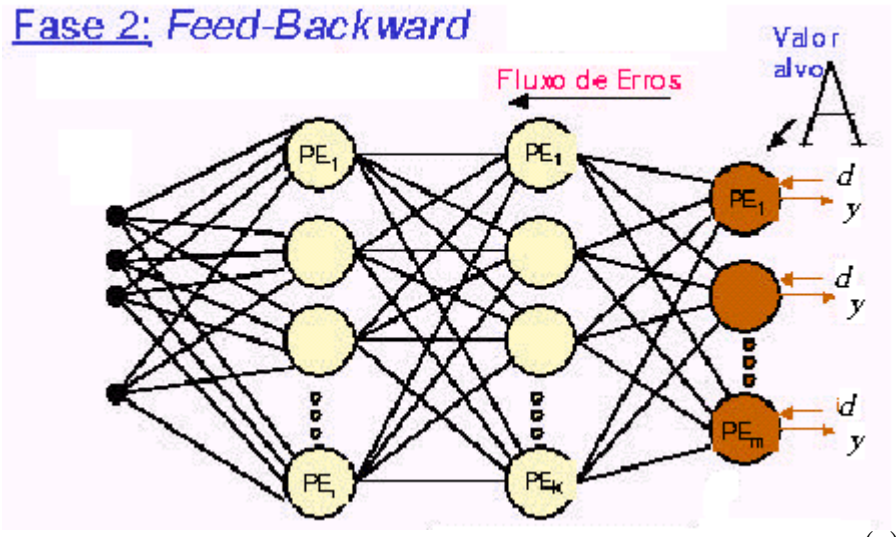
- El entrenamiento tiene dos fases cada una es recorriendo las capas en un solo sentido.
  - Fase forward (prueba)
  - Fase Backward (entrenamiento)
- A partir de la última capa
  - Cada neurona ajusta sus pesos a partir de su error
  - Cada neurona de capas anteriores tienen sus errores definidos por los errores de capas subsecuentes, ponderadas por sus pesos.
  - El entrenamiento intenta reducir el error hasta que sea aceptable



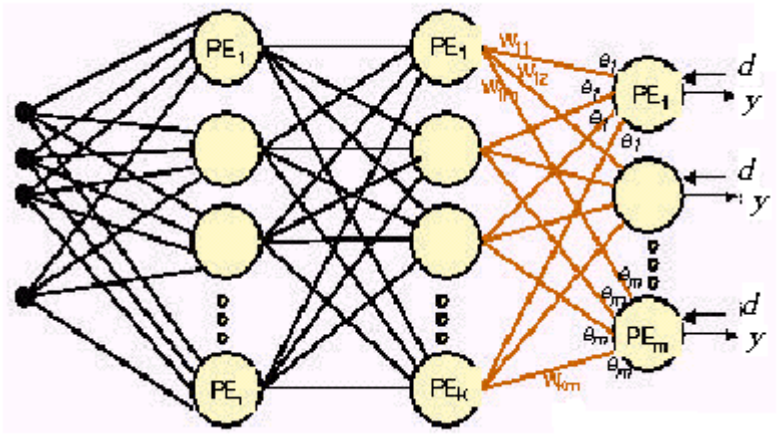

Entrada
2 Camadas Escondidas
Camada de Saída
PE1
PEn
PEk
PEm
Valor de entrada


Entrada
2 Camadas Escondidas
Camada de Saída
PE1
PEn
PEk
PEm
Valor de entrada

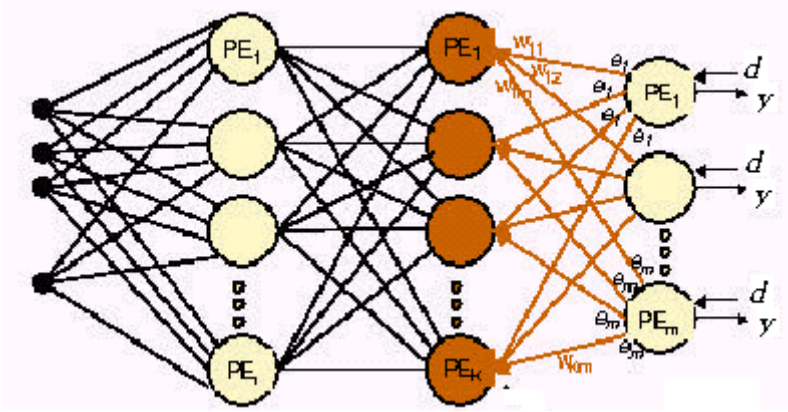

Entrada
2 Camadas Escondidas
Camada de Saída
PE1
PEn
PEk
PEm
Valor de entrada



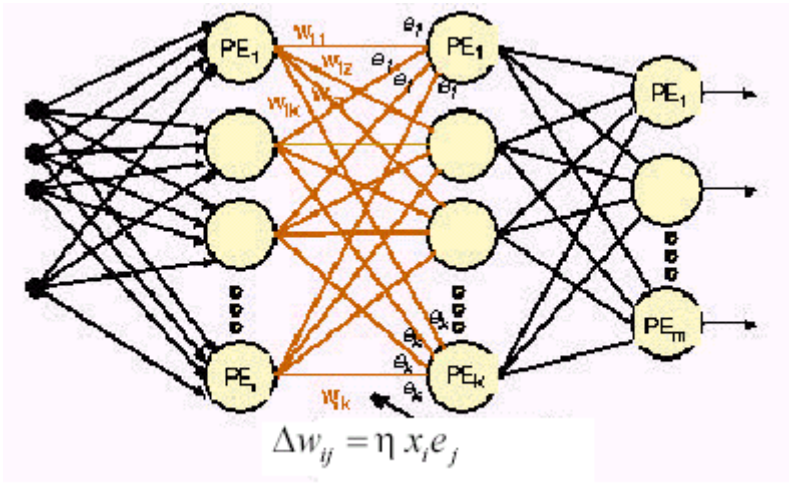
$$e_j = (d_j - y_j)\frac{\partial F(u)}{\partial u}$$



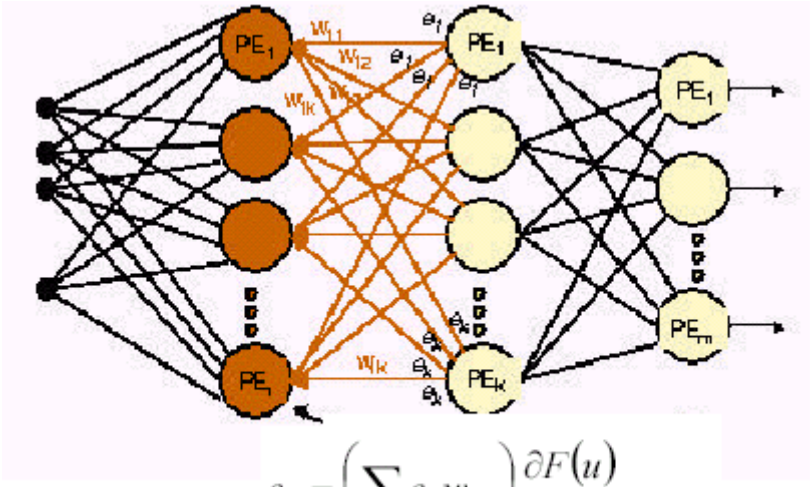
$$\Delta w_{ij} = \eta\, e_j\, y_j$$



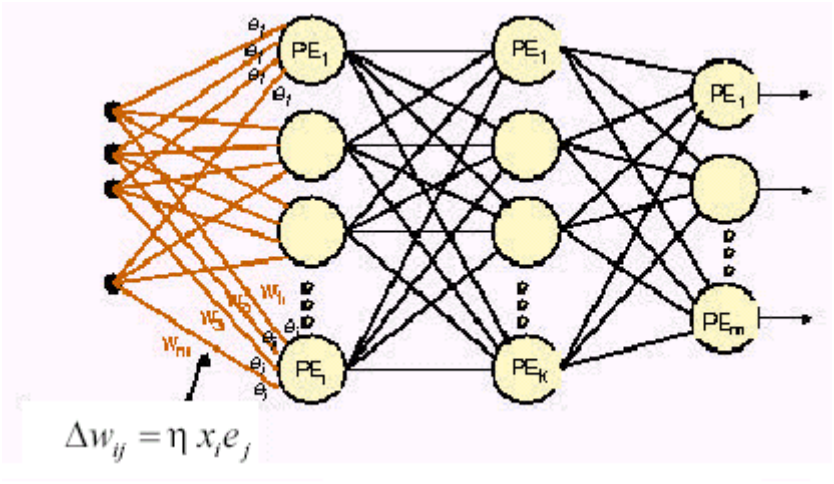
$$e_j = \left(\sum_k e_k w_{jk}\right)\frac{\partial F(u)}{\partial u}$$



$$\Delta w_{ij} = \eta\, x_i e_j$$



$$e_j = \left(\sum_k e_k w_{jk} \right) \frac{\partial F(u)}{\partial u}$$



$$\Delta w_{ij} = \eta\, x_i e_j$$

Algoritmo

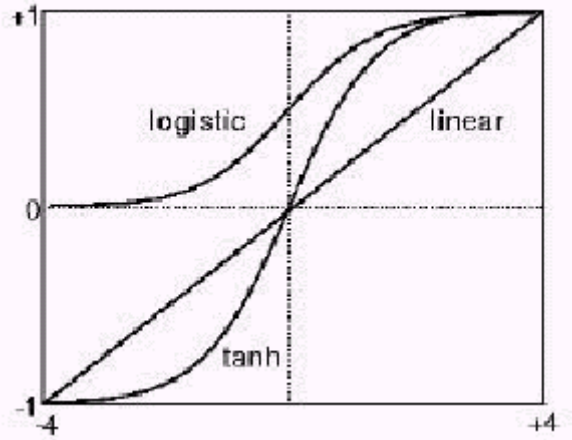
Inicialización de pesos con valores aleatorios ($|w_{ij}| < 0.1$)

Entrenamiento

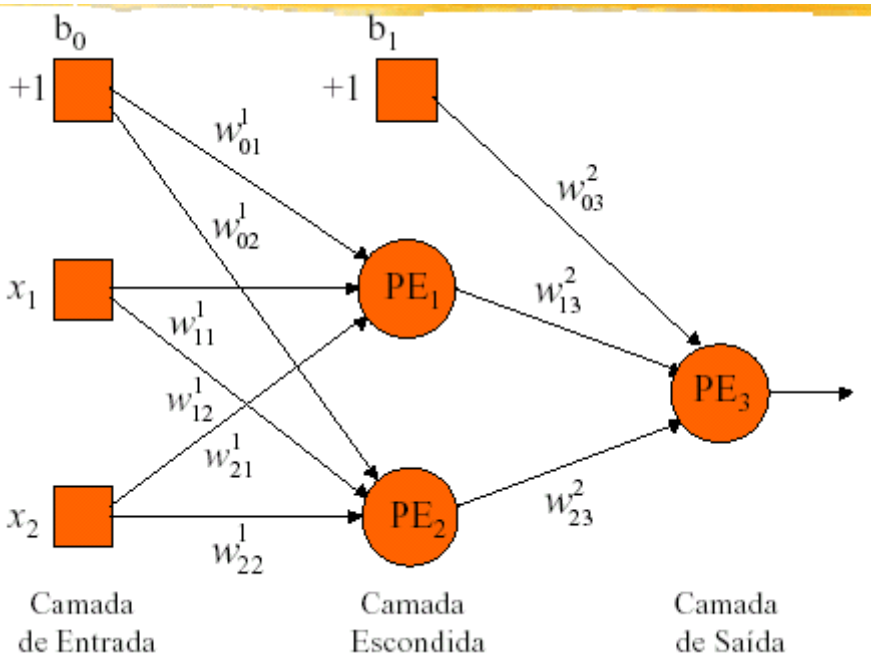
Repetir hasta que el error de cada neurona de salida esta deseada (cierta tolerancia) para todos los patrones de un conjunto de entrenamiento

1) Presentar un patrón de entrada x_i y su respectivo valor de salida deseada
2) Calcule las salidas de las neuronas comenzando en la primera capa escondida hasta la capa de salida.
3) Calcule el error para cada neurona de la capa de salida. Si el error es menor que la tolerancia para todos las neuronas, retorne para 1
4) Actualiza los pesos de cada procesador comenzando por la capa de salida hasta la capa de entrada.
5) Vuelva al paso 1.

Función de activación.



Ejemplo



Entrada $x_1 = 1$ y $x_2 = 0$ salida deseada $d_3 = 1$
Pesos iniciales $w_{ij} = 0$ Tasa de aprendizaje $\eta = 0.5$

Función de activación $F(u_i) = \dfrac{1}{1+\exp(-u_i)}$

Derivada de la función de activación $F'(u_i) = \dfrac{\exp(u_i)}{[1+\exp(-u_i)]^2}$

La actualización de los pesos están dados por:

$$w_{ij} = w_{ij} + \eta x_i e_j$$

$$e_j = (d_j - y_j)F'(u_j)$$

$$e_j = F'(u_j)\sum e_k w_{jk}$$

$b_0 = 1$, $b_1 = 1$

$u_1 = 1\times0 + 1\times0 + 0\times0 = 0$ $\quad$ $y_1 = 0.5$

$u_2 = 1\times0 + 1\times0 + 0\times0 = 0$ $\quad$ $y_2 = 0.5$

$u_3 = 1\times0 + 0.5\times0 + 0.5\times0 = 0$ $\quad$ $y_3 = 0.5$

Entonces, $d_3 - y_3 = 1 - 0.5 = 0.5$ $\quad$ $e_3 = 0.5 \times 0.25 = 0.125$

$$w_{03}^3 = 0 + 0.5\times1\times0.125 = 0.0625$$

$$w_{13}^2 = 0 + 0.5\times0.5\times0.125 = 0.0313$$

$$w_{23}^2 = 0 + 0.5\times0.5\times0.125 = 0.0313$$

$e_1 = 0.125 \times 0.0313 = 0.0039$

$e_2 = 0.125 \times 0.0313 = 0.0039$

$$w_{01}^1 = 0 + 0.5\times1\times0.0039\times0.25 = 0.0004875$$

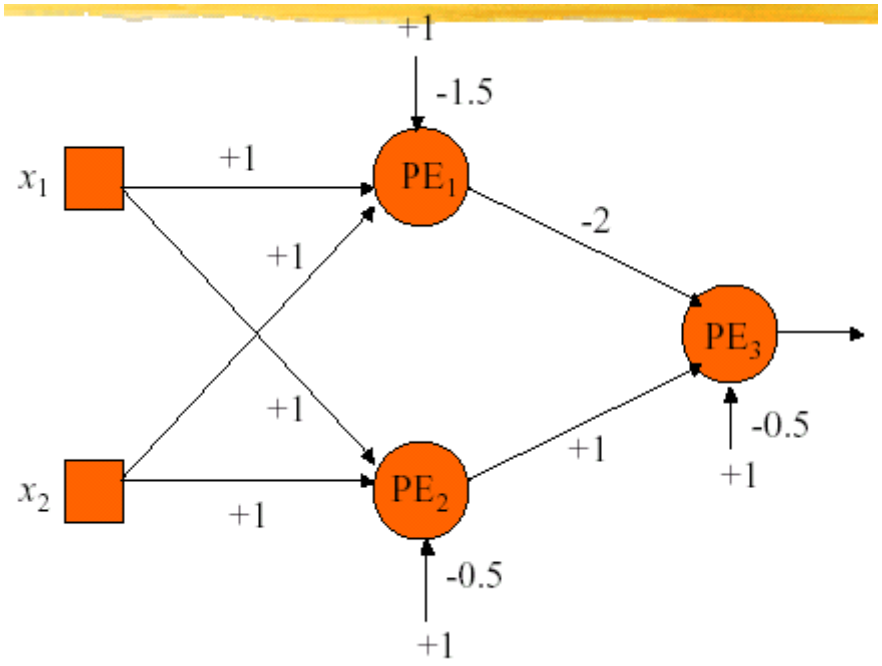
$$w_{02}^1 = 0 + 0.5\times1\times0.0039\times0.25 = 0.0004875$$

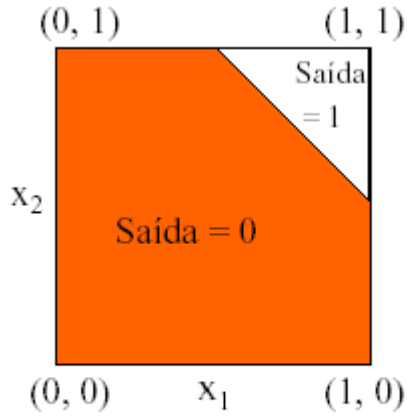
$$w_{11}^1 = 0 + 0.5\times1\times0.0039\times0.25 = 0.0004875$$

$$w_{12}^1 = 0 + 0.5\times1\times0.0039\times0.25 = 0.0004875$$

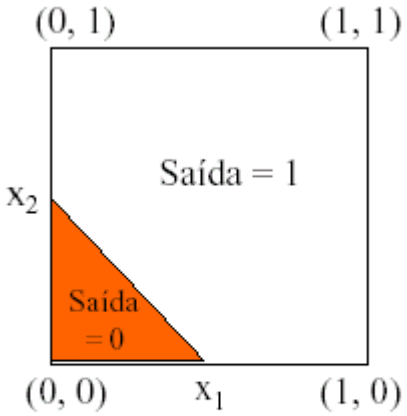
$$w_{21}^1 = 0 + 0.5\times0\times0.0039\times0.25 = 0$$

$$w_{22}^1 = 0 + 0.5\times0\times0.0039\times0.25 = 0$$

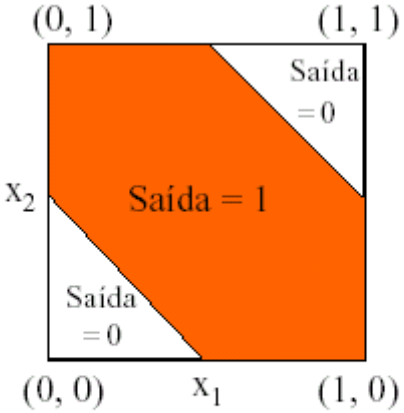




Borda de decisão construída pelo 1ª neurônio escondido



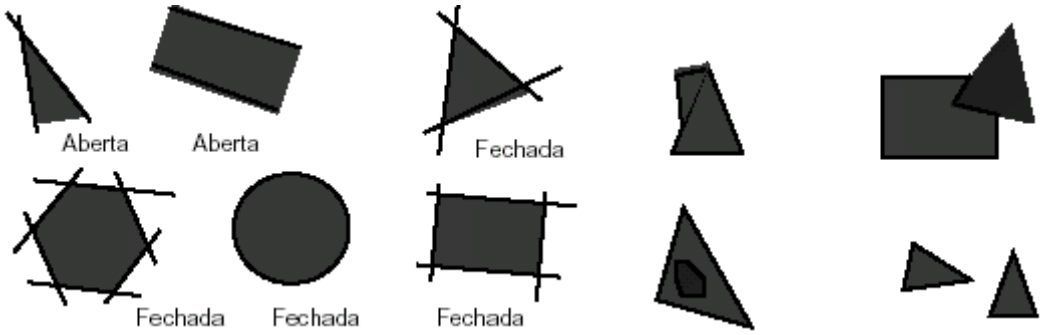
Borda de decisão construída pelo 2ª neurônio escondido



Borda de decisão construída pela rede completa

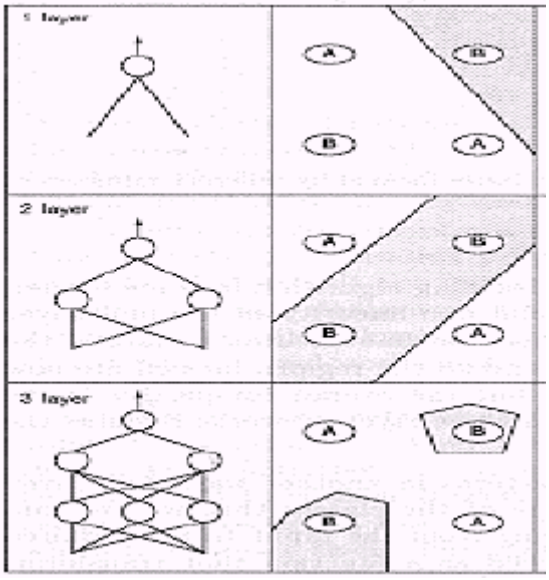
MLP como Clasificadores

- La función implementada por cada neurona es la combinación de las funciones implementadas por las neuronas de la capa anterior.
    - Capa 1: líneas rectas en el espacio de decisión
    - Capa 2: regiones convexas. Numero de lados = número de unidades en la capa anterior
    - Capa 3: combinación de figuras convexas, produciendo formatos abstractos, número de figuras convexas = número de unidades de la capa anterior.



Regiones Convexas Combinación de Regiones Convexas

MLP como clasificadores



Unidades intermedias

- Detectores de características
- Dado un numero grande o suficiente de unidades intermedias es posible formar representaciones internas para cualquier división de los patrones de entradas.

Número de capas intermedias necesarias

- 1 Capa suficiente para aproximar cualquier función continua o boleana.
- 2 Capas suficiente para aproximar cualquier función
- 3 o mas capas pueden facilitar el entrenamiento de las red, pero hay que tener cuidado cada vez que el error es propagado a la capa anterior, se torna menos útil.

Número de neuronas por capa.

- En general no es conocido
- Usar en función del número de entradas y salidas
- Depende de:
    - Número de ejemplos de entrenamiento
    - Cantidad de ruido
    - Complejidad de la función a ser aprendida

- Puede crecer exponencialmente con el número de entradas.
- Una solución adecuada es aquella donde el número de unidades intermedias crece apenas en forma polinomial con el número de entradas.

Clasificación correcta de patrones no usados en el entrenamiento o con ruido.

- La correcta clasificación de los patrones sucede a través de la detección de características relevantes del patrón de entrada durante el entrenamiento
- Patrones desconocidos son atribuidos a clases cuyos patrones presentan características semejantes
- Es garantizado para tolerancia a fallas

Dificultades durante el entrenamiento

- Mínimos locales: Solución estable que no provee una salida correcta
  - Emplear coeficiente de aprendizaje decreciente
  - Adicionar nodos intermediarios
  - Usar momento
  - Adicionar ruido
- Backpropagation es muy lento en superficies complejas
- Overfitting
  - Después de un cierto momento la red empeora en vez de mejorar
  - Memoriza patrones de entrenamiento, incluyendo sus particularidades (empeora la generalización)
  - Alternativas de solución:
    - Terminar el entrenamiento mas antes
    - Reducir pesos

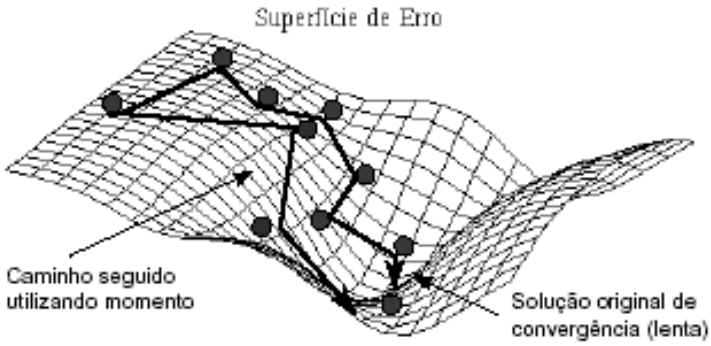
Actualización de los pesos

- Ciclo
  - Presentar todos los ejemplos de entrenamiento durante el aprendizaje
  - Los ejemplos deben ser presentados de forma aleatoria

- El método de actualización va a depender principalmente de la aplicación

Variaciones del BackPropagation

- Momento
  - $\Delta w_{ij}(t + 1) = \eta\, x_i y_j (1 - y_j) d_j + \alpha\, (w_{ij}(t) - w_{ij}(t - 1))$
  - Aumente la velocidad de aprendizaje evitando el peligro de inestabilidad
  - Puede acelerar el entrenamiento en regiones planas de la superficie de error



Consejos para mejorar el funcionamiento

- Utilizar una RNA usando Backpropagation puede ser tomado mas como arte que como ciencia
    - Envuelve muchos factores que son principalmente dependientes del tipo de problema a solucionar
    - Resultado de la experiencia del especialista
- Usar una función Tangente Hiperbolica
    - Aprendizaje mas rápido
- Respuesta deseada debe estar $[-a + \varepsilon, a - \varepsilon]$
    - a = Máximo de la función de activación
- Inicialización de los pesos y θ uniformemente distribuidos dentro de un intervalo pequeño
    - Reduce la probabilidad que las neuronas se saturen
    - Cuidado intervalos pequeños pueden tornar el entrenamiento muy lento
- La tasa de aprendizaje no debe ser preferencialmente la misma para todas las neuronas
    - Generalmente las ultimas capas tienen mayor gradiente que las capas iniciales sendo la tasa de aprendizaje mayor para las ultimas capas
    - Neuronas con muchas entradas deben tener tasa de aprendizaje menores
- Usar en lo posible padrones que envuelvan bases de datos grandes y redundantes

Aplicaciones

- Generar sonido a partir de textos
- Reconocimiento de códigos postales
- Filtraje de ruido de electrocardiogramas
- Previsión en series de tiempo
- Analices de crédito
- Procesamiento de imágenes