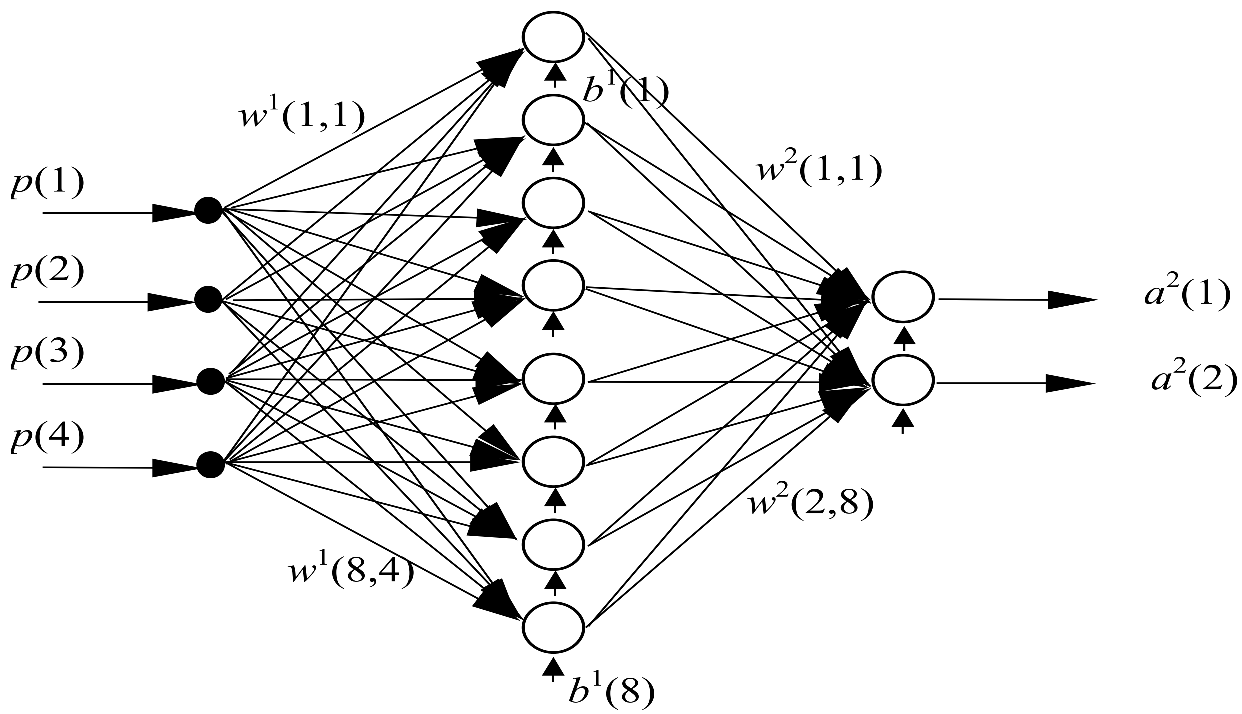
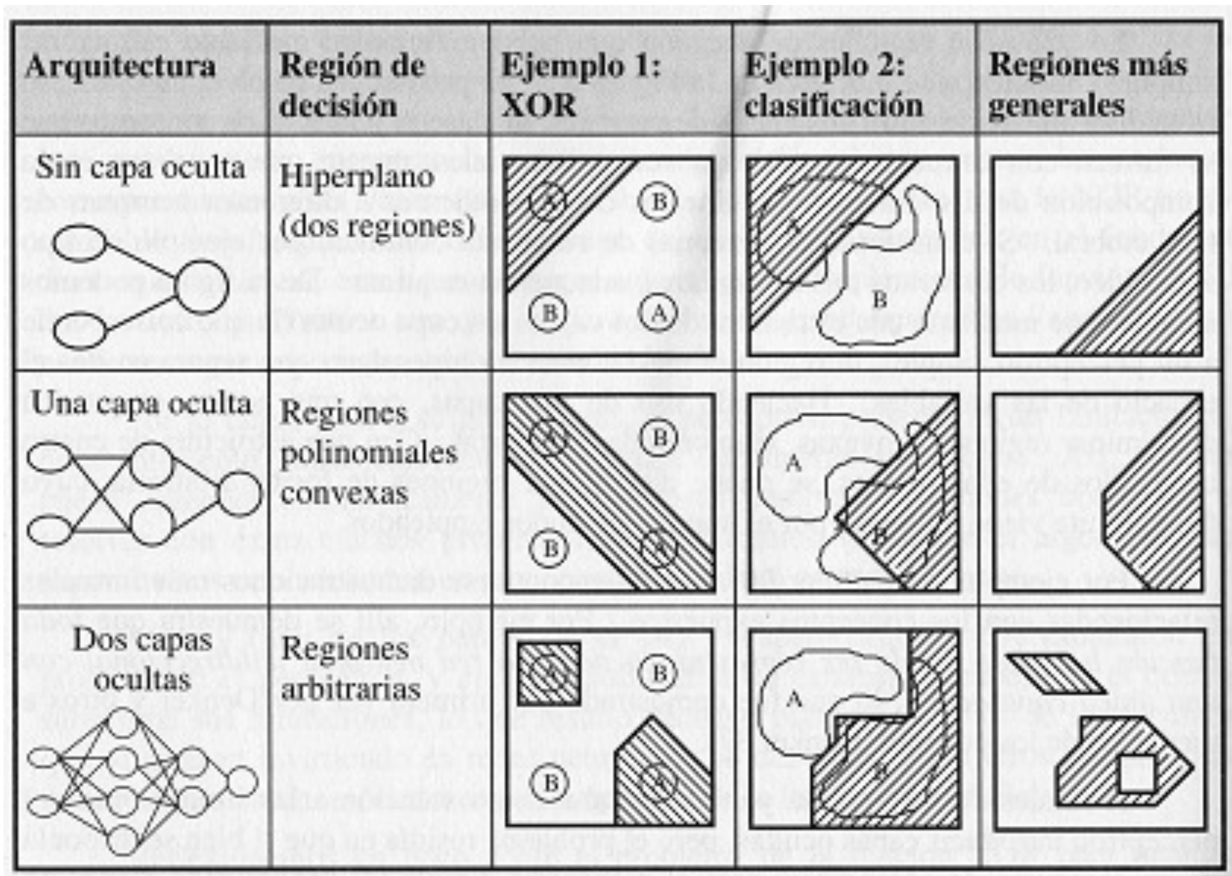
Redes Neuronales Artificiales

Dr. Misael López Ramírez

REDES NEURONALES MULTICAPA



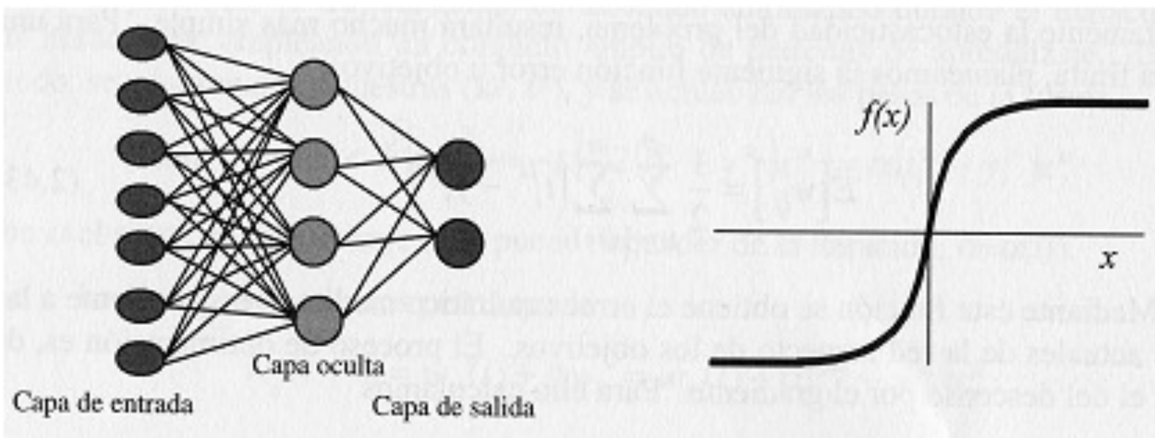
Perceptrón Multicapa por que???



Arquitectura	Región de decisión	Ejemplo 1: XOR	Ejemplo 2: clasificación	Regiones más generales
Sin capa oculta	Hiperplano (dos regiones)			
Una capa oculta	Regiones polinomiales convexas			
Dos capas ocultas	Regiones arbitrarias			

Multi-Layer Perceptron MLP

- Limitaciones del perceptrón simple
- Se suele entrenar por medio de Back Propagation BP
  - **Rumelhart (1986)**



Arquitectura de MLP

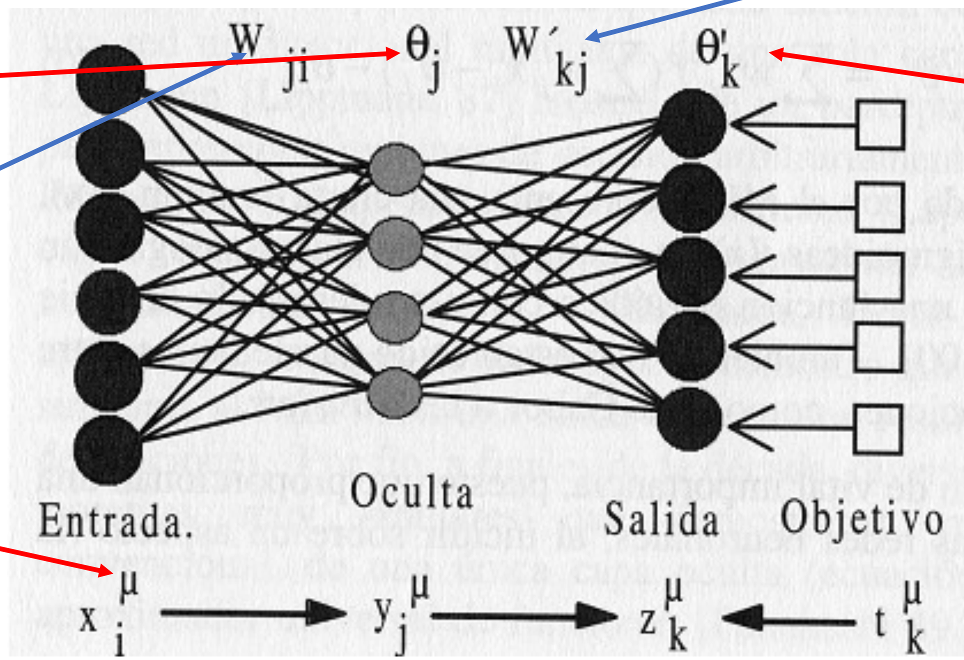
Pesos de salida

Bias o humbral
Capa oculta

Bias o umbral
Capa de salida

Pesos de Capa
Oculta

Iteración ó epocas



$$z_k = \sum_{j=1}^{o} w'_{kj} y_j - \theta'_k = \sum_{j=1}^{o} w'_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i - \theta_j\right) - \theta'_k$$

O=# de neuronas capa salida

n=# de entradas capa de entrada

j=# neuronas

f= sigmoide

Consideraciones en MLP

- La popularidad de la arquitectura MLP se debe al hecho de que un MLP con una única capa oculta puede aproximar cualquier función continua en un intervalo hasta el nivel deseado, cuestión demostrada por Funahaski (1989).

Entrenamiento Back propagation

- Retroporpagación de los errores:
  - Es un extensión del algoritmo Least Mean Square (LMS).
  - Se utiliza regla de la cadena
  - Se consideran funciones de transferencia diferenciables

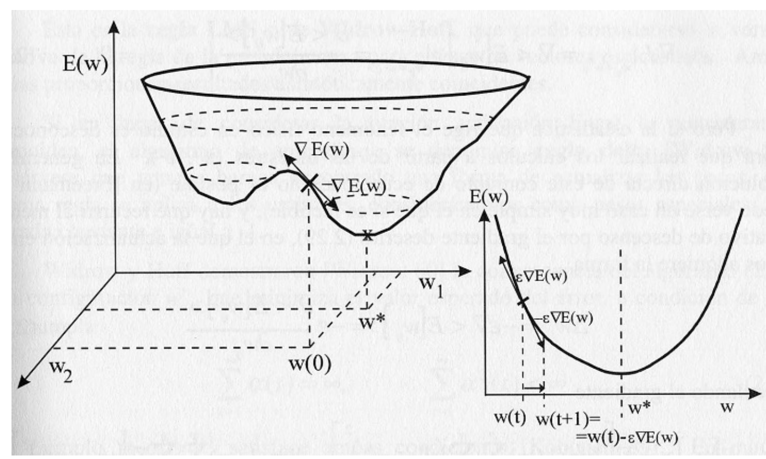
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \varepsilon \nabla E(\mathbf{w})$$

Recordando

$$E(\mathbf{w}) = \frac{1}{2}\sum_{r=1}^{N}\sum_{i=1}^{m}(c_i^r - y_i^r)^2$$

$$\frac{\partial E(w_{ij})}{\partial w_{ij}} = -\left(\frac{1}{2}\right) 2 \sum_{r=1}^{N} (c_i^r - y_i^r)\frac{dy_i^r}{dw_{ij}} = -\sum_{r=1}^{N} (c_i^r - y_i^r)\, x_j^r$$

$$\Delta w_{ij} = -\varepsilon \frac{\partial E(w_{ij})}{\partial w_{ij}} = \varepsilon \sum_{r=1}^{N} (c_i^r - y_i^r)\, x_j^r$$



Entrenamiento Back propagation

K-salidas (m)

r-salidas= observaciones(N)

$$z_k^r = \sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k = \sum_{j=1}^{o} w'_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j\right) - \theta'_k$$

Error Cuadrático Medio $E(\mathbf{w}, \mathbf{w}', \boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{2} \sum_{r=1}^{N} \sum_{k=1}^{m} (c_k^r - z_k^r)^2$

Obteniendo el gradiente con respectó a los pesos de las dos capas

$$\Delta w'_{kj} = -\varepsilon \frac{\partial E}{\partial w'_{kj}}$$

$$\Delta w_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}}$$

Sustituyendo Zk En el error cuadrático Medio

$$\Delta w'_{kj} = \varepsilon \sum_{r=1}^{N} \left(c_k^r - \left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k \right) \right)^2 y_j^r$$

$$\Delta w_{ji} = \varepsilon \sum_{r=1}^{N} \Delta_j^r x_i^r$$

$$\text{con } \Delta_j^r = \left(\sum_{k=1}^{s} \left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k \right) w'_{kj} \right) \frac{\partial f\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j \right)}{\partial \left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j \right)}$$

Entrenamiento Back propagation

$$z_k^r = \sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k = \sum_{j=1}^{o} w'_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j\right) - \theta'_k$$

K-salidas (m)

r-salidas= observaciones(N)

Error Cuadrático Medio $E(\mathbf{w}, \mathbf{w}', \boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{2} \sum_{r=1}^{N} \sum_{k=1}^{m} (c_k^r - z_k^r)^2$

Sustituyendo Zk En el error cuadrático Medio

$$\Delta w'_{kj} = -\varepsilon \frac{\partial E}{\partial w'_{kj}}$$

$$\Delta w_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}}$$

1er paso

$$\Delta w'_{kj} = \varepsilon \sum_{r=1}^{N} \left(c_k^r - \left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k \right) \right) y_j^r$$

3er paso

$$\Delta w_{ji} = \varepsilon \sum_{r=1}^{N} \Delta_j^r x_i^r$$

2do paso señales de erro en capa oculta

$$\boxed{\text{con } \Delta_j^r = \left(\sum_{k=1}^{s} \left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k \right) w'_{kj} \right) \frac{\partial f\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j \right)}{\partial \left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j \right)}}$$

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}^r$$

Algoritmo de aprendizaje

1. Establecer aleatoriamente los pesos y umbrales iniciales (para t := 0).
2. Para cada patrón r del conjunto de entrenamiento
   1. Llevar a cabo una fase de ejecución para obtener la respuesta de la red frente al patrón r-ésimo
   2. Calcularla señales de error asociadas a: $\left(c_k^r - \left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k\right)\right)$ y

$$\left(\sum_{k=1}^{s}\left(\sum_{j=1}^{o} w'_{kj} y_j^r - \theta'_k\right) w'_{kj}\right) \frac{\partial f\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j\right)}{\partial\left(\sum_{i=1}^{n} w_{ji} x_i^r - \theta_j\right)}$$

   3. Calcular el incremento parcial de los pesos y umbrales debidos a cada patrón.
3. Calcular el incremento total actual, extendido a todos los patrones, de los pe $\Delta w'_{kj}$ Δw_{ji}

Algoritmo de aprendizaje

4.- Actualizar pesos y umbrales.

5.-Calculat el error toral.

  1.- Hacer t=t+1 volver al paso 2 si todavía el error total no es satisfactorio

Matlab toolbox

Formato para el Training Set en Matlab:

Matlab

P

Training case 1	Training case 2	Training case 3	Training case 4
entrada 1	entrada 1	entrada 1	entrada 1
entrada 2	entrada 2	entrada 2	entrada 2
entrada 3	entrada 3	entrada 3	entrada 3

T

Training case 1	Training case 2	Training case 3	Training case 4
salida 1	salida 1	salida 1	salida 1
salida 2	salida 2	salida 2	salida 2

Perceptron simple en Matlab

Actividad:

Crear una red neuronal que aprenda la compuerta lógica OR usando Matlab. No se olvide de simular la red.

```
>> net = newff([0 1; 0 1], [1], {'logsig'}, 'traingdx');
>> P=[0 0 1 1; 0 1 0 1];
>> T=[0 1 1 1];
>> net.trainParam.epochs=10000;
>> net.trainParam.goal=0.0001;
>> net = train(net, P, T);
TRAINGDX, Epoch 0/10000, MSE 0.14723/0.0001, Gradient 0.129945/1e-006
...
TRAINGDX, Performance goal met.

>> sim(net, P)

ans =  0.0149    0.9906    0.9907    1.0000
```

Perceptron simple en Matlab

Tip:

Pasos para el diseño y uso de una red neuronal:

1. Crear el conjunto de datos de entrenamiento. (Training Set)
2. Crear el conjunto de datos validación. (Validation Set)
3. Crear la red.
4. Entrenar la red (Usar el conjunto de datos de entrenamiento).
5. Validar la red para averiguar si aprendió y generalizó. (Usar el conjunto de datos de validación)
6. Usar la red aplicando datos nuevos, posiblemente diferentes a los de entrenamiento y validación.

MLP en Matlab

Actividad:

Diseñe y entrene una red para aprender la compuerta XOR. Pruebe con 0, una y dos neuronas en el nivel escondido 1 hasta obtener un goal de 0.00001. Use el mínimo número de neuronas. Use Neural Lab y Matlab. Use la función de activación tanh(ax).

Training Set

Train case	in 1	in 2	out 1
1	0.0000000	0.0000000	-1.0000000
2	0.0000000	1.0000000	1.0000000
3	1.0000000	0.0000000	1.0000000
4	1.0000000	1.0000000	-1.0000000

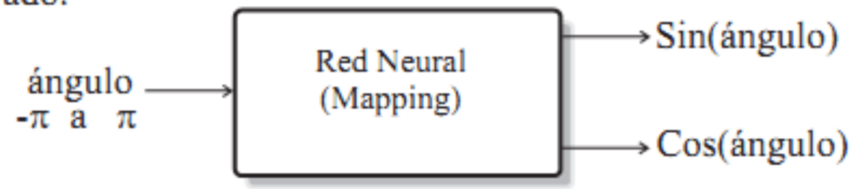
p_1	p_2	y_1
0	0	0
0	1	1
1	0	1
1	1	0

```
net = newff([0 1; 0 1], [3, 1], {'tansig', 'tansig'}, 'trainlm');
P=[0 0 1 1; 0 1 0 1];
T = [-1 1 1 -1];
net.trainParam.epochs =10000;
net.trainParam.goal = 0.00001;
net = train(net, P, T);

sim(net, P)
-0.9988    0.9997    0.9990   -0.9939
```

Actividad:

Se creara una red neuronal para aprender las funciones seno y coseno. Escriba el archivo Trigom.m mostrado.




```
clear;
P=[-pi: 0.01 : pi];
T= [sin(P); cos(P)];
size(P)
size(T)
'Valores minimos y maximos de P'
minmax(P)
'Valores minimos y maximos de T'
minmax(T)
[PN, minp, maxp, TN, mint, maxt] = premnmx(P, T);
'Valores minimos y maximos de PN'
minmax(PN)
'Valores minimos y maximos de TN'
minmax(TN)
net = newff([-1 1], [6, 2], {'tansig', 'tansig'}, 'trainlm');
net.trainParam.goal=0.0001;
net.trainParam.epochs=1500;
net=train(net, PN, TN);
```

feedforwardnet

```
'Valores de entrada para simular la red'
X=[-pi: 0.005 : pi];
XN= tramnmx(X, minp, maxp);
YN = sim(net, XN);

'Valores de salida producidos por la red'
Y = postmnmx(YN, mint, maxt);
plot(Y');
```

mapminmax

Normalizacion
Post-entrenamiento

Normalizacion
Pre-entrenamiento
-1 a 1

Tips:

1. Sobre-entrenamiento: Un síntoma de sobre entramiento es que la red trabaja muy bien con el Training Set pero produce malos resultados con el Validation Set.

2. El conjunto de datos de validación y de entrenamiento debe representar en forma apropiada el experimento.

3. Bajo ninguna circunstancia, se puede usar el Validation Set para entrenamiento.

4. El Training Set debe contener todos los distintos tipos de lecciones (training cases) posibles en el problema real.

5. El Training Set no debe ser más grande que lo necesario.

6. Las redes más grandes requieren Training Sets más grandes.

7. El Training Set no de contener desviaciones creadas por factores humanos.

8. El Training Set puede obtenerse de una colección muy grande de datos y un generador de número aleatorios, para seleccionar sólo algunos datos del conjunto original.

9. El Training Set debe ser escalado apropiadamente para acoplarse a las funciones de activación de las neuronas.

Tips:

Número de Neuronas y Niveles Escondidos

Reglas generales:

1. Usar en lo posible sólo un nivel escondido.
2. Usar el menor número de neuronas escondidas.
3. Entrenar hasta que se termine la paciencia.

Niveles escondidos:

1. Usar más de un nivel escondido es muy pocas veces beneficial.

2. Más niveles escondidos inestabilizan el entrenamiento y producen más falsos mínimos (es difícil escapar de estos.)

3. Usar dos niveles escondidos solamente cuando la función a aprender presenta discontinuidades.

4. Nunca usar más de dos niveles escondidos.

Se recomienda comenzar con un nivel escondido. Si un número grande de neuronas escondidas en este nivel no resuelven el problema satisfactoriamente, entonces se puede intentar incrementar el número de neuronas en el segundo nivel y posiblemente reducir el número total de neuronas.

Un número excesivo de neuronas producirá el aprendizaje de efectos particulares (over fitting) que no son generales entre todas las muestras.

Descripción	Comandos Básicos para Redes Neuronales en Matlab.
	Usted puede crear un archivo *.m para ejecutar una secuencia de comandos
Crear la red	net = newff([min1 max1; min2 max2; ...], [3, ..., 4], {'logsig',...}, 'traingdx');

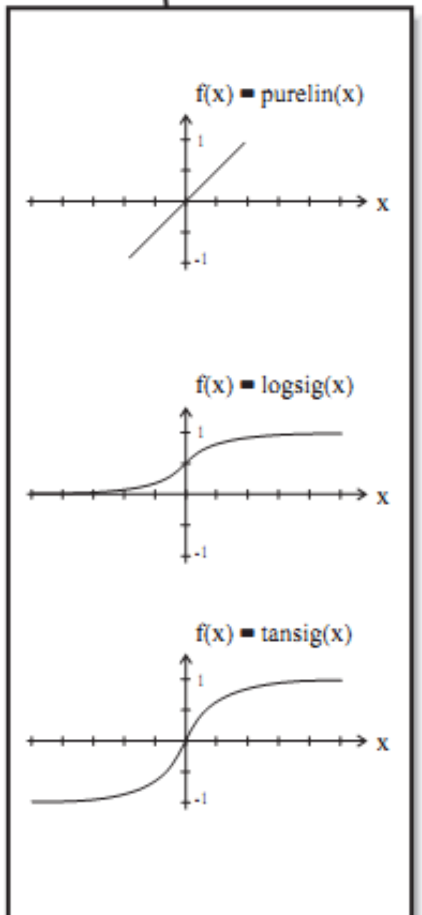
$\underbrace{\text{min1 max1; min2 max2; ...}}_{\text{rango de entradas}}$

Numero de Neuronas En Nivel Escondido 1

Numero de Neuronas de Salidas

- network: custom neural net.
- newc: competitive layer
- newcf: cascade-forward b.
- newelm: Elman
- newff: feed-forward backpr.
- newfftd: feed-forw. in-delay
- newgrnn: gen. regress.
- newhop: Hopfield recurrent
- newlin: linear layer
- newlind: design linear layer
- newlvq: learning vec. quan.
- newp: perceptron.
- newpnn: probab. neural net
- newrb: radial basis network
- newrbe: design an e. r. basis
- newsom: self-org. map

- trainbfg: BFGS quasi-Newton backpr.
- trainbr: Bayesian regularization
- traincgb: Powell-Beale conj. grad. backpr.
- traincgf: Fletcher-Powell conj. grad. backpr.
- traincgp: Polak-Ribiere conj. grad. backpr.
- traingd: Gradient descent backprop.
- traingda: Gradient descent adaptive
- traingdm: Gradient descent momentum
- traingdx: Gradient descent mom-adaptive
- trainlm: Levenberg-Marquardt backpr.
- trainoss: One-step secant backpr.
- trainrp: Resilient backpropagation.
- trainscg: Scaled conjugate gradient backpr.
- trainb: Batch training weight bias learning
- trainc: Cyclical order inc. training learning f.
- trainr: Random order inc. training learning f.



Obtener ayuda	help nnet
Abrir la interface gráfica	nntool
Limpiar	clear
Inicialización de la red	initzero, midpoint, randnc, randnr, rands
Guardando la red	usar la nntool
Retrayendo la red	usar la nntool
Ajustando la red	net.IW para Input Weights net.LW para Layer Weights net.b para la bias
Entrenar la red	train(net, P, T)
Usar la red	sim(net, p)

El Objeto Network

- net.numInputs
- net.numLayers
- net.inputs
- net.layers
- net.outputs
- net.targets
- net.inputWeights
- net.layerWeights
- net.trainFcn
- net.performFcn

- net.trainParam.epochs
- net.trainParam.goal
- net.trainParam.show