Las Redes Neuronales

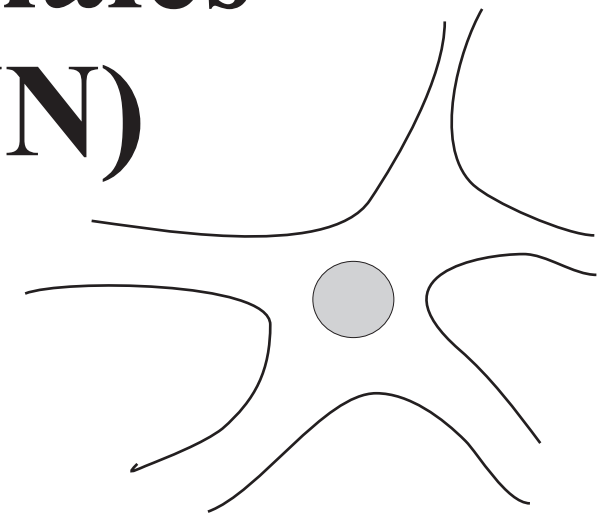
implementación y consideraciones prácticas

Dr. Sergio Ledesma
selo@salamanca.ugto.mx
Facultad de Ingeniería
Universidad de Guanajuato

MICAI

Fifth Mexican
International Conference on
Artificial Intelligence

Las Redes Neuronales Artificiales (ANN)



Parte I:

1. Introducción a las redes neuronales.
2. Usando redes neuronales (mapping).

Parte II:

3. Consideraciones prácticas (genetic algorithms & simulated anneling)

Parte III:

4. Reducción de ruido (redes auto-asociativas).

Parte IV:

5. Clasificación (redes usadas como clasificadores).
6. El Neurocontrolador.

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

1. Introducción a las redes neuronales (1).

Definición:

Las redes neuronales son una implementación muy sencilla de un comportamiento local observado en nuestros cerebros. El cerebro está compuesto de neuronas, las cuales son elementos individuales de procesamiento. La información viaja entre las neuronas, y basado en la estructura y ganancia de los conectores neuronales, la red se comporta de forma diferente.

Tip:

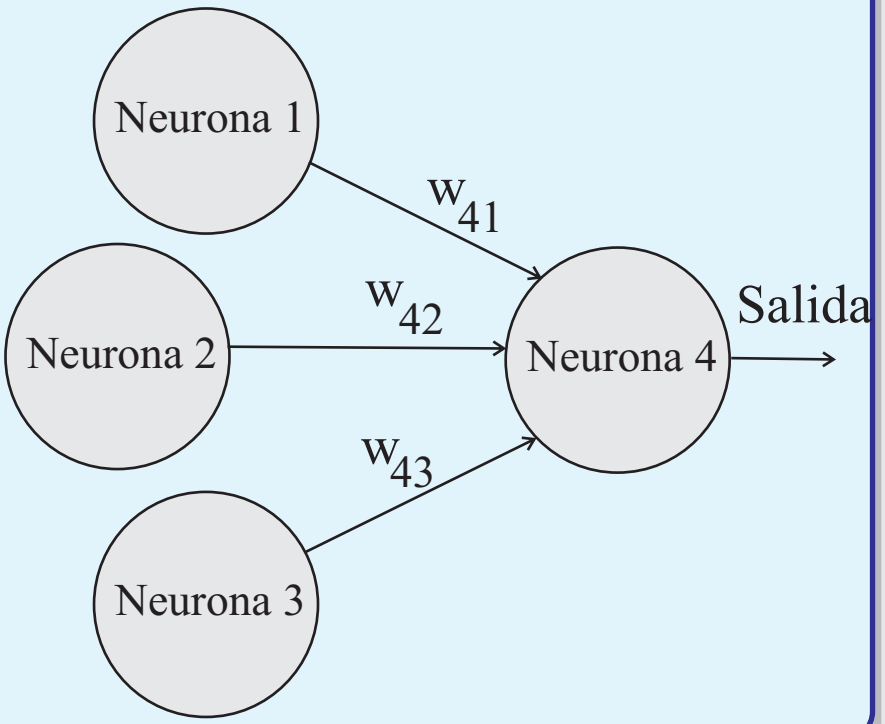
El cerebro humano contiene aproximadamente 100,000 millones de neuronas. Cada neurona está conectada aproximadamente a otras 1000 neuronas, excepto en la corteza cerebral donde la densidad neuronal es mucho mayor.

Tip:

Hoy en día las redes neuronales son entrenadas para resolver problemas que son difíciles para las computadoras convencionales o los seres humanos.

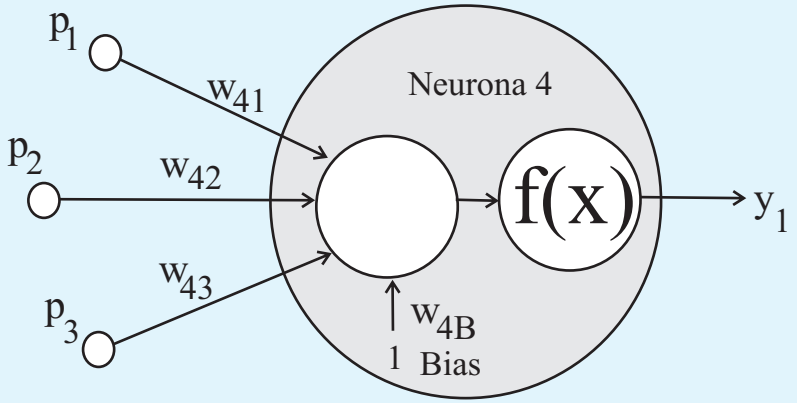
Definición:

Cada neurona está conectada con otra neurona por medio de un peso de ajuste representado por la letra w, el primer subíndice indica la neurona destino, mientras que el segundo subíndice indica el la neurona de origen.



Definición:

Una neurona artificial está formada por un sumador y una función de activación, representada por f(x).



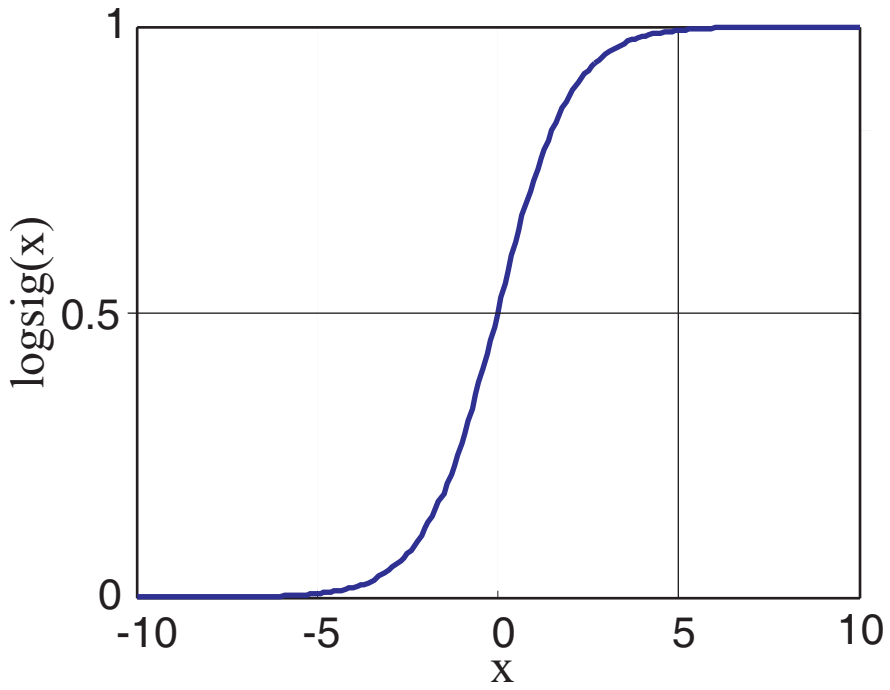
donde la función de activación f(x) debe ser un sigmoide (forma de S), continua, real, de rango acotado y tener una derivada positiva. La funciones más populares, en las redes de varias, son la tangente hiperbólica y la función logística:

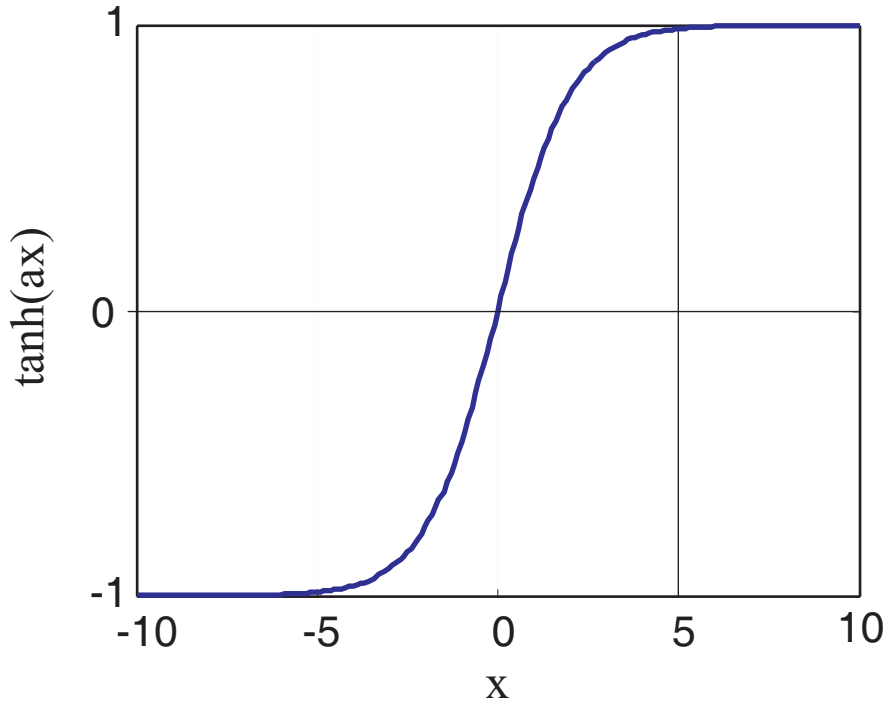
$$f(x) = logsig(x) = \frac{1}{1 + e^{-x}}$$

$$f(x) = \tanh(ax)$$

1. Introducción a las redes neuronales (3).

Las funciones de activación



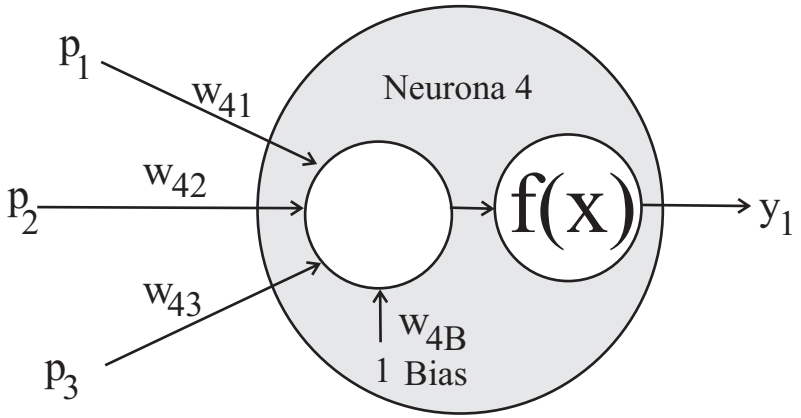


Actividad

Encuentre la salida de la red neuronal mostrada, suponga que f(x) es la función logsig(x).

$\vec{\mathbf{p}} = [7 \quad 9 \quad -10]$

$\vec{\mathbf{w}} = [-2 \quad 4 \quad 5 \quad 11]$



Solución:

$N_1 = (7)(-2)+(9)(4)+(-10)(5)+(1)(11) = -17$

$$y_1 = \frac{1}{1 + e^{17}} = 4.14 \times 10^{-8}$$

Tip:

Durante el entrenamiento de una red todas las neuronas de la red se activarán un número considerable de veces; para reducir el tiempo de entrenamiento es posible una utilizar una tabla de lookup e interpolación para evaluar la función de activación.

La clase Logsig descrita en los archivos Logsig.h y Logsig.cpp reduce el tiempo de ejecución en más de la mitad que cuando se usa la evaluación directa de la función.

Logsig.h

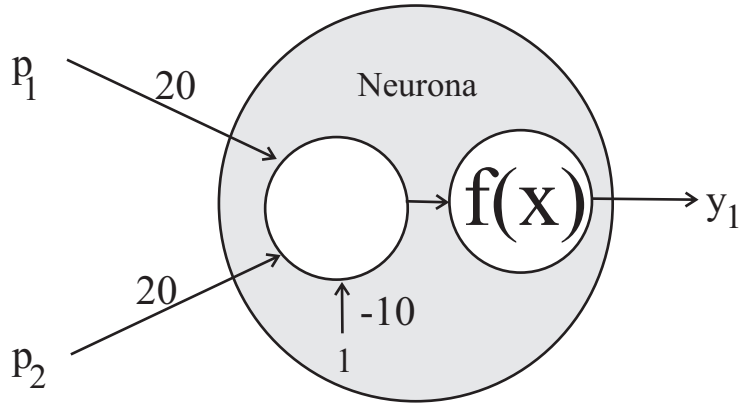
```
#pragma once
#define NN_AF_TABLE_LENGTH 2048
#define NN_AF_TABLE_MAX 14

class Logsig
{
public:
 Logsig();
 ~Logsig();
 double getValue(double x)
 {
  if (x>NN_AF_TABLE_MAX) return 1.0;
  if (x<-NN_AF_TABLE_MAX) return 0.0;
  double xd = fabs(x*factor);
  int i = (int)xd;
  double y = funcion[i]+derivada[i]*(xd-i);
  return (x>=0) ? y : 1.0-y;
 }
private:
 double factor;
 double funcion[NN_AF_TABLE_LENGTH];
 double derivada[NN_AF_TABLE_LENGTH];
};
```

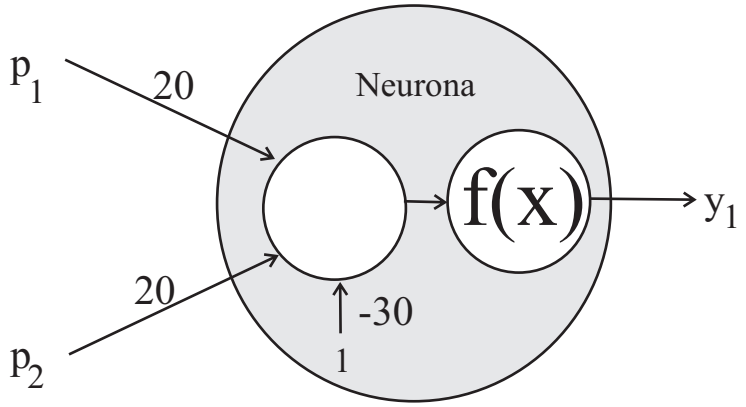
Logsig.cpp

```
#include "Logsig.h"

Logsig::Logsig(void)
{
 factor = (double)(NN_AF_TABLE_LENGTH-1)/(double)NN_AF_TABLE_MAX;
 for(int i=0; i<NN_AF_TABLE_LENGTH; i++)
 {
  funcion[i]=1.0/(1.0+exp(-((double)i)/factor));
  if (i!=0) derivada[i-1]= funcion[i] - funcion[i-1];
 }
}

Logsig::~Logsig(void)
{
}
```

Actividad:

Complete las tablas de salida mostrada, suponga que f(x) = logsig(x). Compruebe sus resultados usando NeuralLab.





p_1	p_2	y_1
0	0	
0	1	
1	0	
1	1	

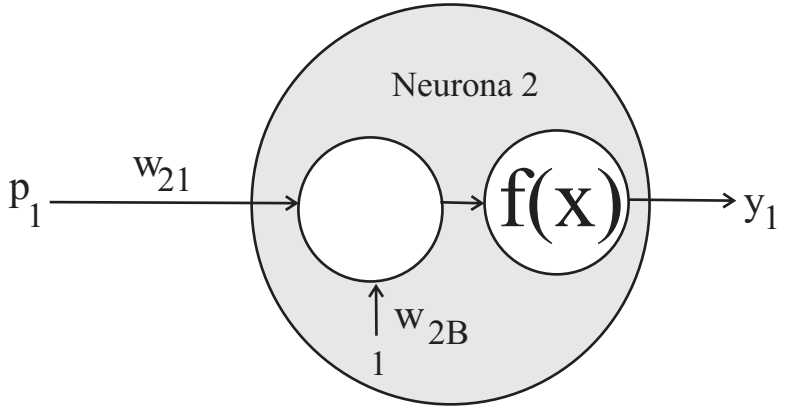
p_1	p_2	y_1
0	0	
0	1	
1	0	
1	1	

Tip:

En las actividades anteriores los pesos de la red fueron asignados manualmente. Sin embargo, una red neuronal es capaz de ajustar sus pesos por sí misma cuando recibe el entrenamiento apropiado.

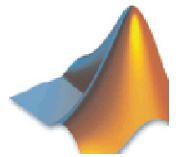
Actividad:

Encuentre el valor de los pesos para modelar una compuerta NOT, suponga que f(x) = logsig(x). Compruebe sus resultados usando NeuralLab.



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
```

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

Actividad:

Crear una red neuronal que aprenda la compuerta lógica AND usando NeuralLab. No se olvide de simular la red. Configure la red para obtener un goal de 0.000000001

Neural Lab

Tip:

Rosenblatt diseñó la primer red neuronal artificial en 1958 (el perceptrón). Desafortunadamente dos compañeros de clases, muy respetados y reconocidos, publicaron un análisis de matemático que explicada las deficiencias de las redes neuronales en 1969. Debido a esto y a la muerte de Ronsenblatt en 1971, los fondos para investigación en redes neuronales se terminaron. En 1986, se demostró que las redes neuronales podían se usadas para resolver problemas prácticos y fue entonces que éstas comenzaron ha ser importantes.

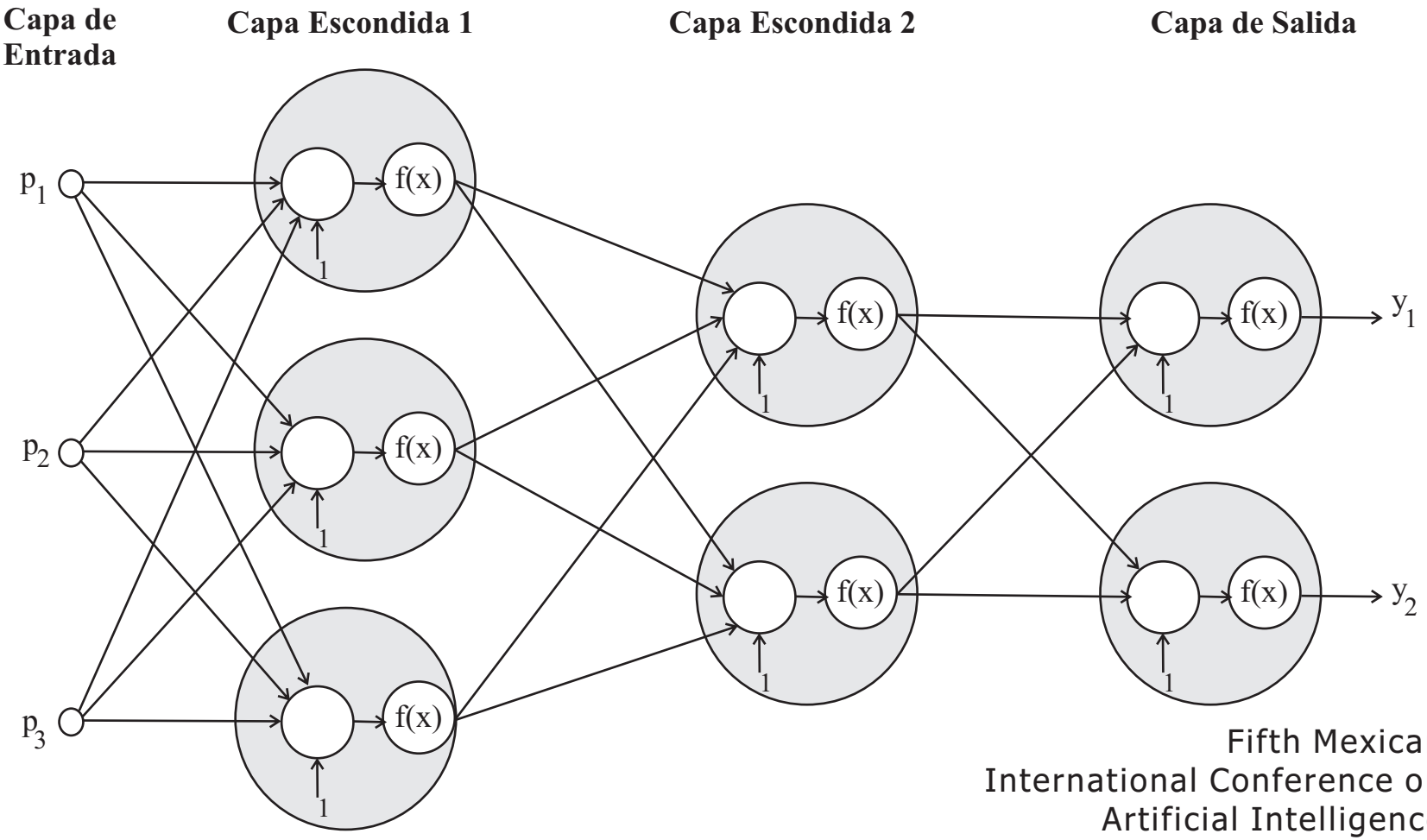
Tip:

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.

Actividad:

La red mostrada tiene tres entrada, dos salidas y dos niveles escondifos. Calcule el número de pesos (variables de ganancia de conexión entre neuronas) que deben ajustarse durante el entrenamiendo de la red.



Fifth Mexican International Conference on Artificial Intelligence
Dr. Sergio Ledesma

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

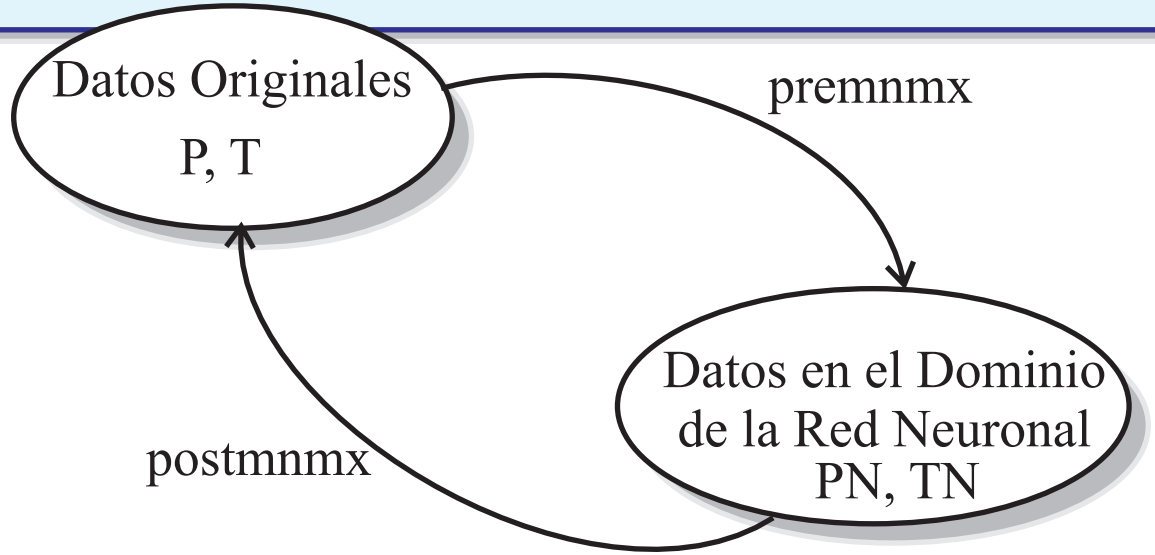
Train case	in 1	in 2	out 1
1	0.0000000	0.0000000	-1.0000000
2	0.0000000	1.0000000	1.0000000
3	1.0000000	0.0000000	1.0000000
4	1.0000000	1.0000000	-1.0000000

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
```

Definición:

Las redes neuronales pueden ser más eficientes si pre procesamiento de datos es aplicado a los datos de entrada y a los de salida deseados. Por ejemplo ciertas redes prefieren valores en el rango de ideal de 0 a 1 (real de 0.1 a 0.9) mientras otras redes prefieren datos en el rango de ideal -1 a 1 (real de -0.9 a 0.9), esto se conoce como normalización de rango.

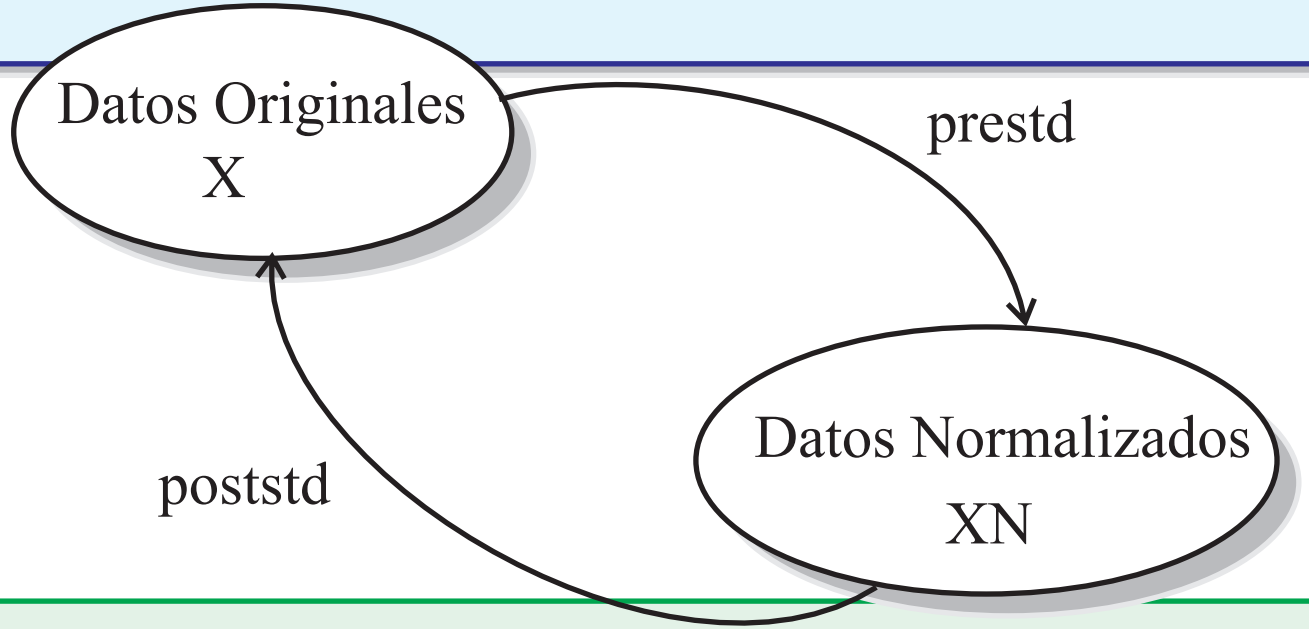


Tip:

Una vez que la red está entrenada, todos los datos de entrada deben ser ajustados al rango de la red. A su vez, los datos de salida de la red deben ser transformados al rango original de los datos.

Definición:

En algunos casos es conveniente normalizar los datos antes de entrenar una red neuronal. El procedimiento de normalización consiste en transformar los datos de tal forma que tengan una media de 0 y una varianza de 1. Una vez terminado el entrenamiento se de usar la desnormalización para regresar al dominio original de los datos.



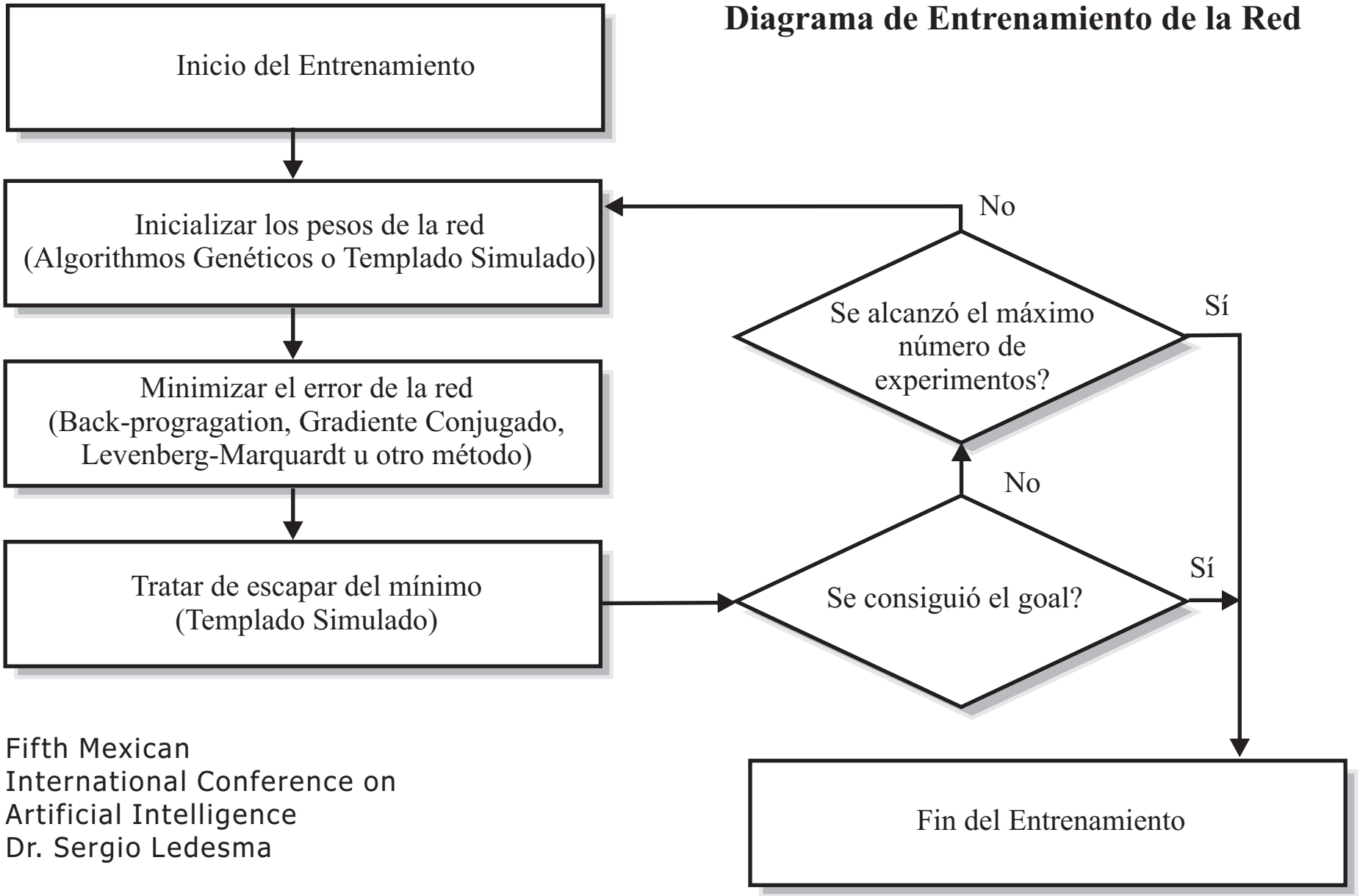
Tip:

Una vez que la red está entrenada, todos los datos de entrada deben ser ajustados al rango de la red. A su vez, los datos de salida de la red deben ser transformados al rango original de los datos.

Tip:

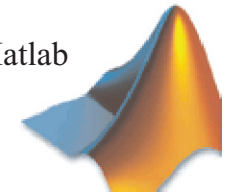
Debido a que durante el entrenamiento se usan procedimientos aleatorios, es recomendable realizar múltiples experimentos desde "cero" para ver si en alguno de ellos se consigue un menor error al evaluar el conjunto de datos de entrenamiento.

Diagrama de Entrenamiento de la Red



2. Usando Redes Neuronales (4).

Formato para el Training Set en 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

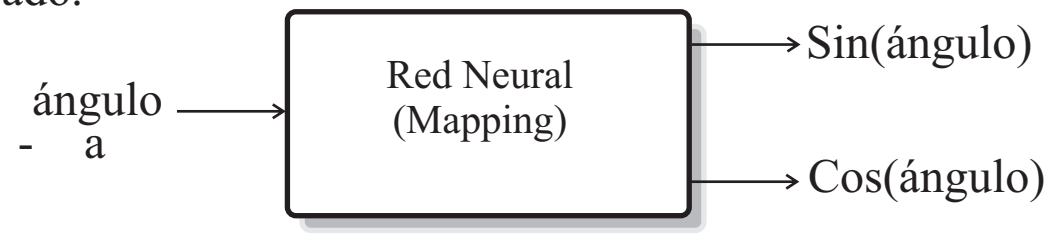
Formato para el Training Set en Neural Lab:

Training Set

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

Actividad:

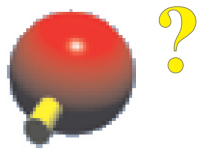
Se creara una red neuronal para aprender las funciones seno y coseno. Escriba el archivo Trigom.m mostrado.




```
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);
```

```
'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');
```

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

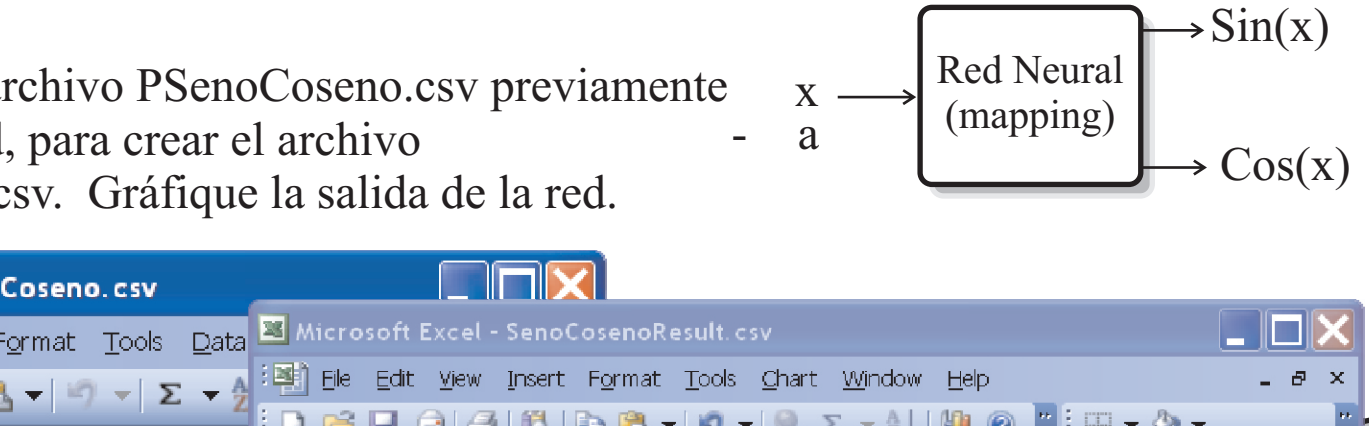


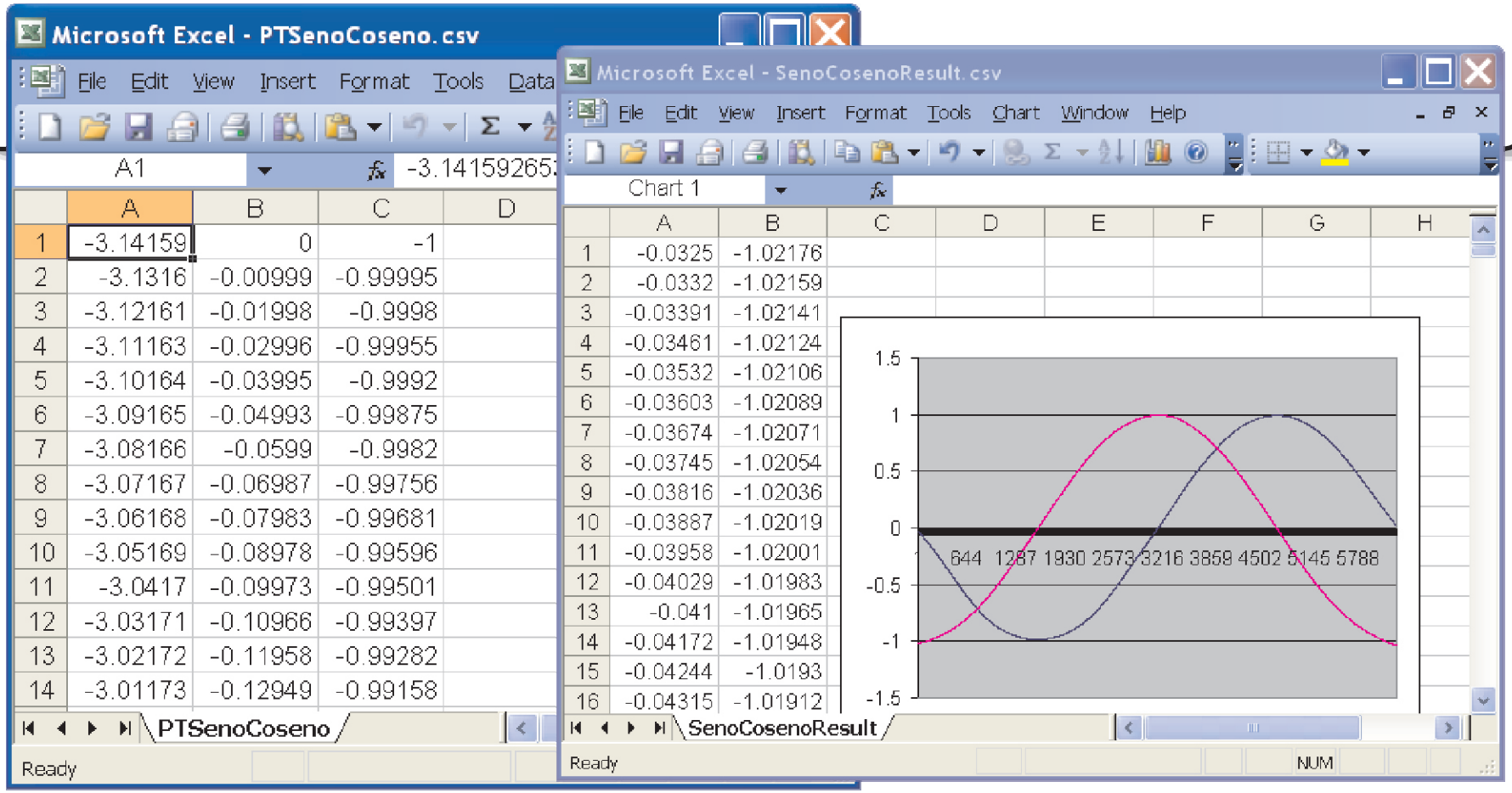
Neural Lab

Actividad:

Parte 1. Crear una red neuronal para aprender las funciones seno y coseno usando Neural Lab. Use una configuración 1 : 6S : 2S escalando la salida de [0.1 0.9] a [-1 1].

Parte 2. Usando el archivo PSenoCoseno.csv previamente creado simule la red, para crear el archivo SenoCosenoResult.csv. Gráfique la salida de la red.





	A	B	C
1	-3.14159	0	-1
2	-3.1316	-0.00999	-0.99995
3	-3.12161	-0.01998	-0.9998
4	-3.11163	-0.02996	-0.99955
5	-3.10164	-0.03995	-0.9992
6	-3.09165	-0.04993	-0.99875
7	-3.08166	-0.0599	-0.9982
8	-3.07167	-0.06987	-0.99756
9	-3.06168	-0.07983	-0.99681
10	-3.05169	-0.08978	-0.99596
11	-3.0417	-0.09973	-0.99501
12	-3.03171	-0.10966	-0.99397
13	-3.02172	-0.11958	-0.99282
14	-3.01173	-0.12949	-0.99158

	A	B
1	-0.0325	-1.02176
2	-0.0332	-1.02159
3	-0.03391	-1.02141
4	-0.03461	-1.02124
5	-0.03532	-1.02106
6	-0.03603	-1.02089
7	-0.03674	-1.02071
8	-0.03745	-1.02054
9	-0.03816	-1.02036
10	-0.03887	-1.02019
11	-0.03958	-1.02001
12	-0.04029	-1.01983
13	-0.041	-1.01965
14	-0.04172	-1.01948
15	-0.04244	-1.0193
16	-0.04315	-1.01912

3. Consideraciones Prácticas para diseñar el Training Set(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.

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.

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.)

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

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

4. Nunca usar más de dos niveles escondidos.

3. Consideraciones Prácticas para elegir el método de entrenamiento (3).

Gradiente conjugado o Levenberg-Marquardt?

Tip:

El método de Levenberg-Marquardt para entrenamiento de redes neuronales usualmente es mejor que otros métodos de entrenamiento. Sin embargo, éste método requiere más memoria que otros métodos. En el caso de redes que tienen un gran número de salidas el método del gradiente conjugado puede ser más rápido que el método de Levenberg-Marquardt.

Definición:

El templado simulado (simulated annealing) es un método de optimización que imita el proceso de templado. Templado (annealing) es el proceso de calentar y después enfriar una sustancia en forma controlada. Las propiedades de un sólido dependen de la razón de enfriamiento después que el sólido se ha calendado más allá de su punto se fusión. El resultado deseado es una estructura cristalina fuerte. Si la sustancia se enfría rápidamente el resultado es una estructura defectuosa quebradiza.

Tip:

Para implementar el templado simulado es necesario generar números aleatorios. Desafortunadamente, los generadores aleatorios proporcionados por los compiladores producen secuencias periódicas y presentan correlación serial que impiden la implementación apropiada del templado simulado. Generadores de números aleatorios más sofisticados son requeridos, (vea http://www.library.cornell.edu/nr/)

Tip:

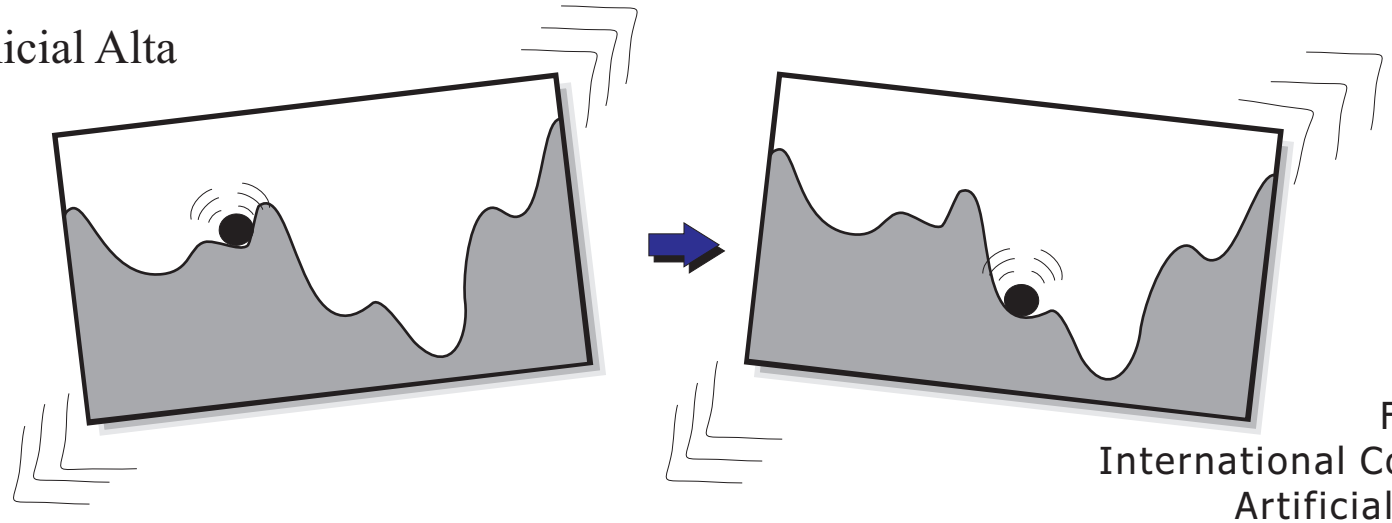
En el templado simulado la estructura representa una solución codificada del problema, y la temperatura es usada para determinar como y cuando nuevas soluciones deben ser aceptadas.

3. Templado Simulado para Inicializar una red (5).

Tip:

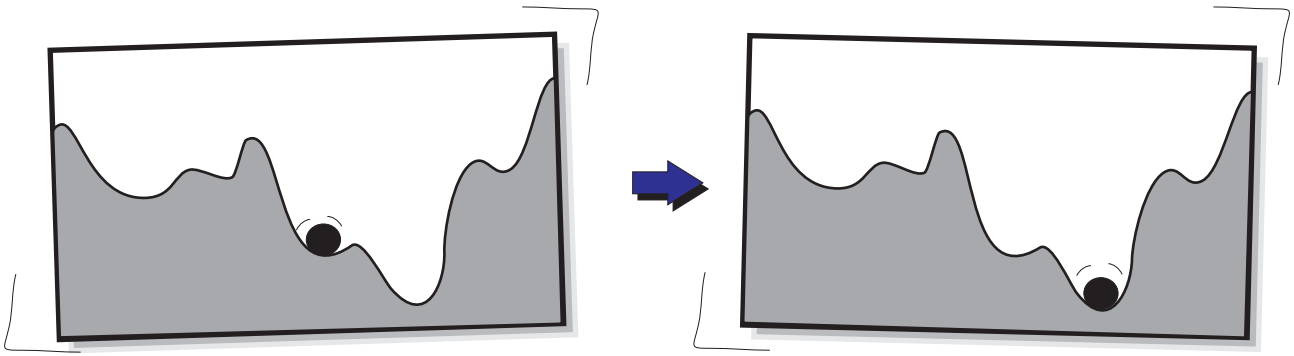
Cuando el sólido ha alcanzado su punto de fusión, grandes cantidades de energía están presentes en el material. A medida que la temperatura se reduce, la energía interna del material disminuye. Otra forma de ver al proceso de templado es como una sacudida o una perturbación. El objetivo de sacudir es encontrar un mínimo global como se muestra en las figuras.

Temperatura Inicial Alta



Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

Temperatura Baja



3. Posible implementación del Templado simulado para sacudir una solución (6).

```
void Templado::Sacudir(int numeroPesos, double *pesosEntrada, double *pesosSalida)
{
 for(int i = 0; i<numeroPesos; i++)
 {
  pesosSalida[i] = pesosEntrada[i] + random.generarValorEntre(-20.0, 20.0);

  //Asegurar que los pesos de salida tomen valores apropiados
  if (pesosSalida[i]<-20)
  {
   pesosSalida[i] = -20.0;
  }
  else if (pesosSalida[i]>20)
  {
   pesosSalida[i] = 20.0;
  }
 }
}
```

Templado.cpp

3. Posible implementación del Templado simulado para saber cuando aceptar una solución (7).

```
bool Templado::SeAceptaLaSolucion(double error, double errorNuevo, double temperature)
{
 bool aceptar = false;
 double deltaError = errorNuevo-error;

 if (deltaError<0)
 {
  aceptar = true;  //Aceptar la solucion si el error es menor
 }
 else
 {
  //Aceptar la solucion con cierta probabilidad
  if (Random.generarValorEntre(0, 1) < exp(-deltaError/temperature))
    aceptar = true;
  else
    aceptar = false;
 }
 return aceptar;
}
```

Templado.cpp

Tips:

Inicialización con Templado Simulado (Simulated Annealing Init)

Start Temperature. La temperatura de inicio. En el caso del algoritmo Metropolis se debe iniciar con una temperatura ligeramente mayor al valor del error inicial. En el caso del algoritmo Normal, valores de cuatro y para arriba indican una búsqueda en una región más amplia.

Stop Temperature. La temperatura final. En el caso del algoritmo Metropolis la temperatura final afecta el error final obtenido por el algoritmo; un valor pequeño indica un error más pequeño. En el caso del algoritmo Normal, valores más pequeños indican que se realiza una búsqueda del mínimo en una región más pequeña.

No Temperatures. El número de temperaturas. Entre mayor sea el número de temperaturas el algoritmo puede ofrecer mejores resultados, sin embargo, el tiempo de ejecución aumenta.

No Iterations. El número de iteraciones en cada temperatura. Entre mayor sea el número de iteraciones el algoritmo puede ofrecer mejores resultados, sin embargo, el tiempo de ejecución aumenta.

Setback. Si se encuentra una solución mejor, el número de iteraciones se incrementa en esa temperatura. Este valor no debe ser mayor al número de iteraciones. Entre mayor sea el setback la probabilidad de encontrar un mejor solución aumenta.

Metropolis o Normal. El algoritmo de Metropolis utiliza probabilidades para seleccionar una solución. El algoritmo normal escoge solamente soluciones mejores a la actual.

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

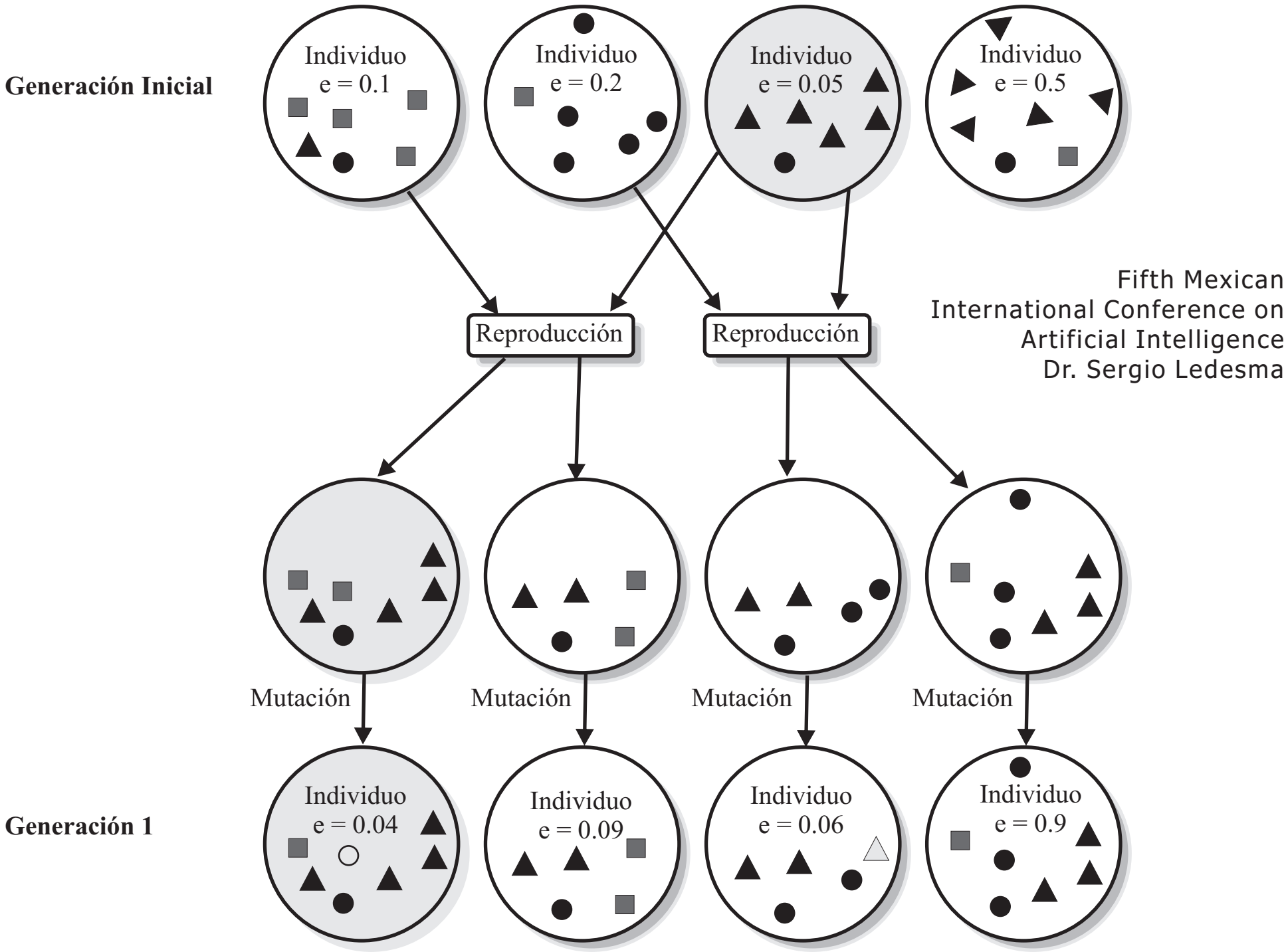
Definición:

Un algoritmo genético es un técnica de optimización basaba en el fenómeno de la evolución, en el cual las especies se adaptan para sobrevivir en ambientes complejos. El proceso de optimización ocurre en la estructura genética de los individuos, la cual afecta la supervivencia y reproducción de las especies.

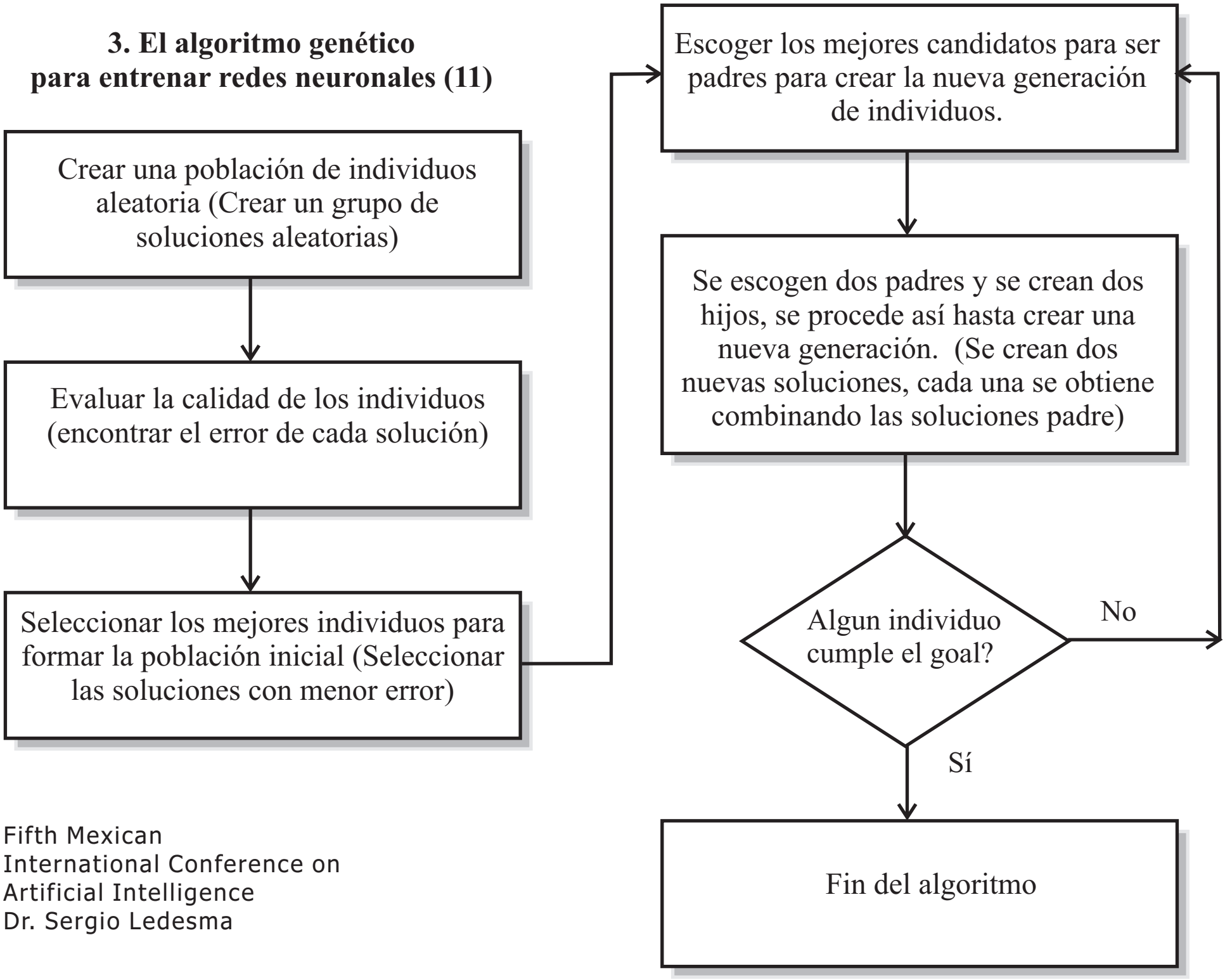
Tip:

En el caso de las redes neuronales cada solución es un individuo, el cual puede heredar sus mejores características a las nuevas generaciones. En cada generación se escogen sólo los mejores individuos (las soluciones con el menor error).

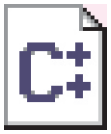
3. Optimización usando algoritmos genéticos (10).



3. El algoritmo genético para entrenar redes neuronales (11)



3. Posible implementación del Algoritmo Genético de la función para mutar un individuo (12).



Genetic.cpp

```
void Genetic::mutar(Individuo individuo)
{
 int indice = 0;
 int numeroGenes = individuo.numeroGenes;
 for(int i = 0; i<numeroGenes; i++)
 {
  //Se debe mutar el gene?
  if (random.generarNumeroEntre(0, 1) < probabilidadMutar)
  {
   indice = random.generarNumeroEntre(0, numeroGenes);
   if (individuo.gene[indice]==0)
   {
    individuo.gene[indice] = 1;
   }
   else
   {
    individuo.gene[indice] = 0;
   }
  }
 }
}
```

3. Posible implementación del Algoritmo Genético de la función para inicializar un individuo (13).

```
void Genetic::inicializar(Individuo individuo)
{
 for(int i = 0; i<individuo.numeroGenes; i++)
 {
  if (random.generarNumeroEntre(0, 1)<0.5)
  {
   individuo.gene[i] = 0;
  }
  else
  {
   individuo.gene[i] = 1;
  }
 }
}
```

Genetic.cpp

3. Como seleccionar los parámetros de la inicialización genética (14).

Tips:

Inicialización Genética (Genetic Init)

Cuando se habla de un individuo en algorithmos genéticos usualmente se habla de una solución.

Hill Climbing. Preservar el mejor individuo en todas las generaciones. Seleccione esta opción solamente cuando usted crea que el mínimo está muy bien definido. En general, esta opción no debe usarse.

Probability of Crossover. Probabilidad de recombinar genes. Cuando es muy baja los mejores individuos son preservados entre generaciones. Lo cual disminuye la creación de nuevos y posibles mejores individuos.

Probability of Mutation. La probabilidad de mutación debe usarse con moderación. En general, la mutación introduce nuevo material genético.

Initial Population. Tamaño de la población inicial. Entre más grande sea la población, el algoritmo puede ofrecer mejores resultados, sin embargo, el tiempo de ejecución aumenta.

Over Init. Se generan más individuos de los que se necesitan. En la primer generación, sólo se escogen los mejores individuos. Entre más grande sea el Over Init, el algoritmo puede ofrecer mejores resultados, sin embargo, el tiempo de ejecución aumenta.

Generations. El número de generaciones. Entre más generaciones se usen la probabilidad de obtener mejores individuos (soluciones) es más alta.

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

3. Tipos de Entrenamiento (15).

Definición:

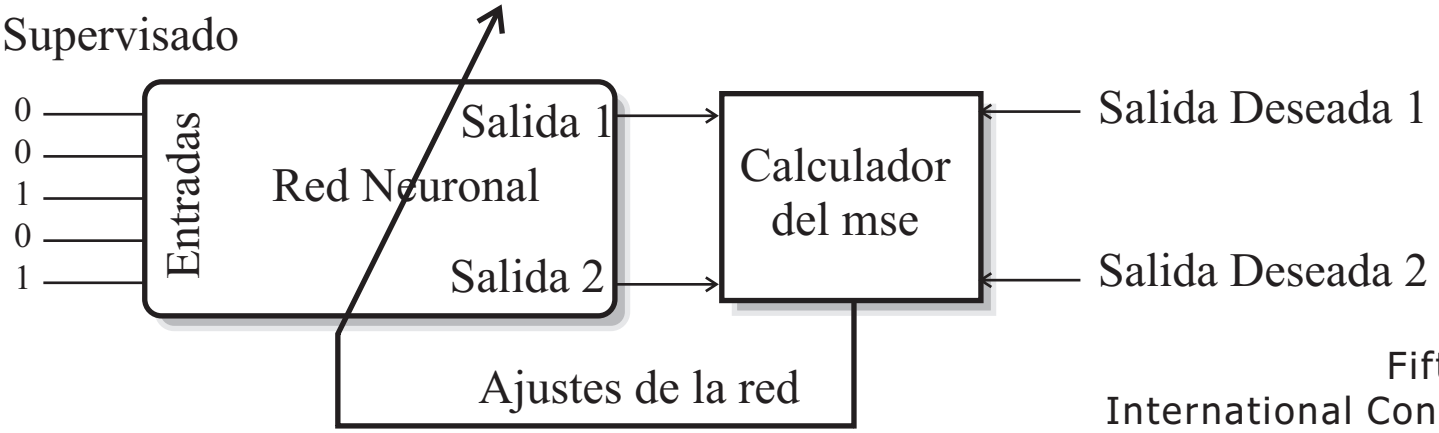
Existen dos tipos de entrenamiento: supervisado y sin supervisión. En el entrenamiento supervisado, a la red se le proporciona la salida deseada y la red se ajusta para producir esta salida. En el entrenamiento sin supervisión, la red descubre patrones por sí sola para clasificar objetos.

Tip:

Para poder ajustar una red que usa entrenamiento supervisado, usualmente se usa el error medio cuadrático entre la salida desea y la salida producida por la red.

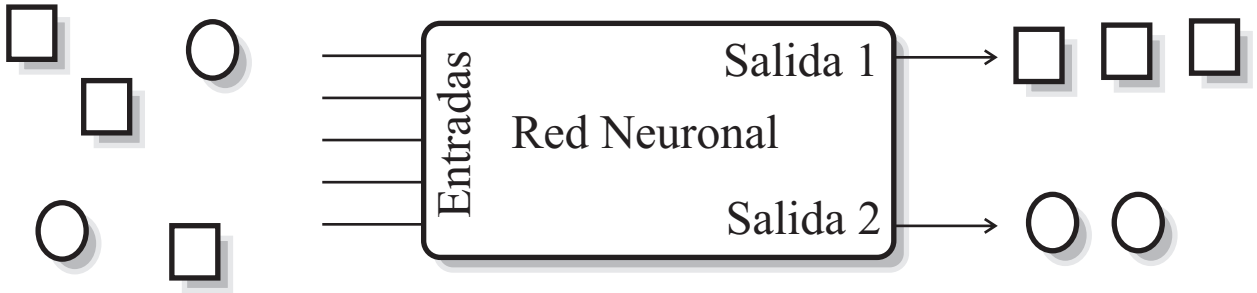
$$mse = (Salida_{deseada} - Salida_{red})^2$$

Entrenamiento Supervisado



Fifth Mexican International Conference on Artificial Intelligence
Dr. Sergio Ledesma

Entrenamiento sin Supervisión

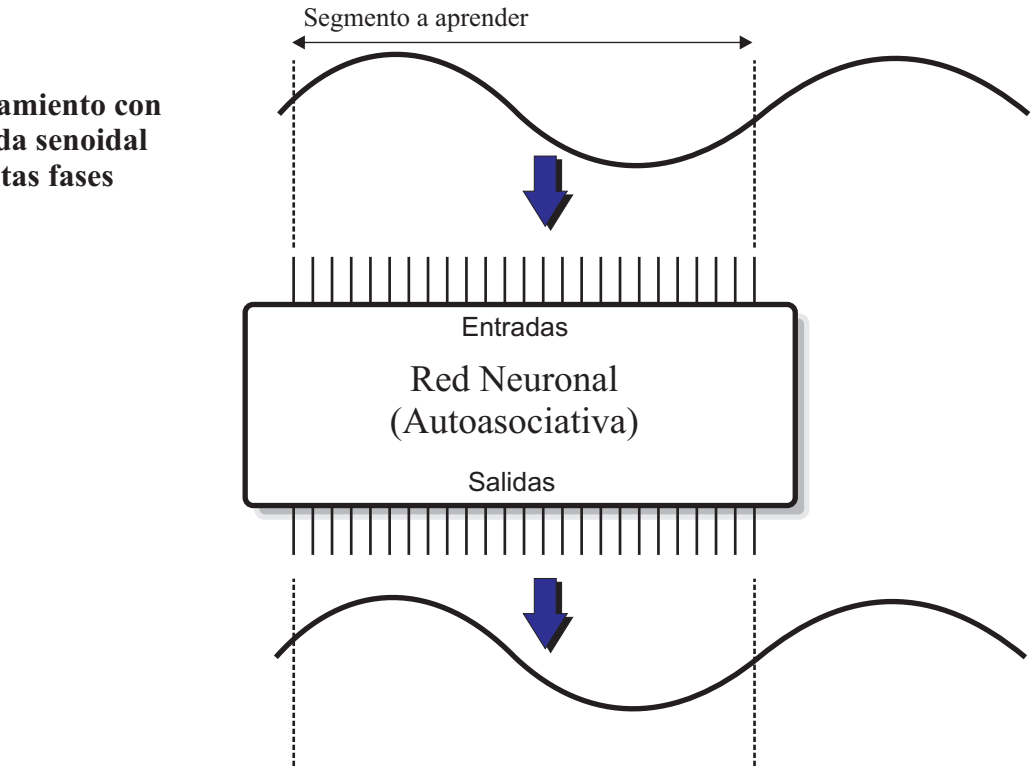


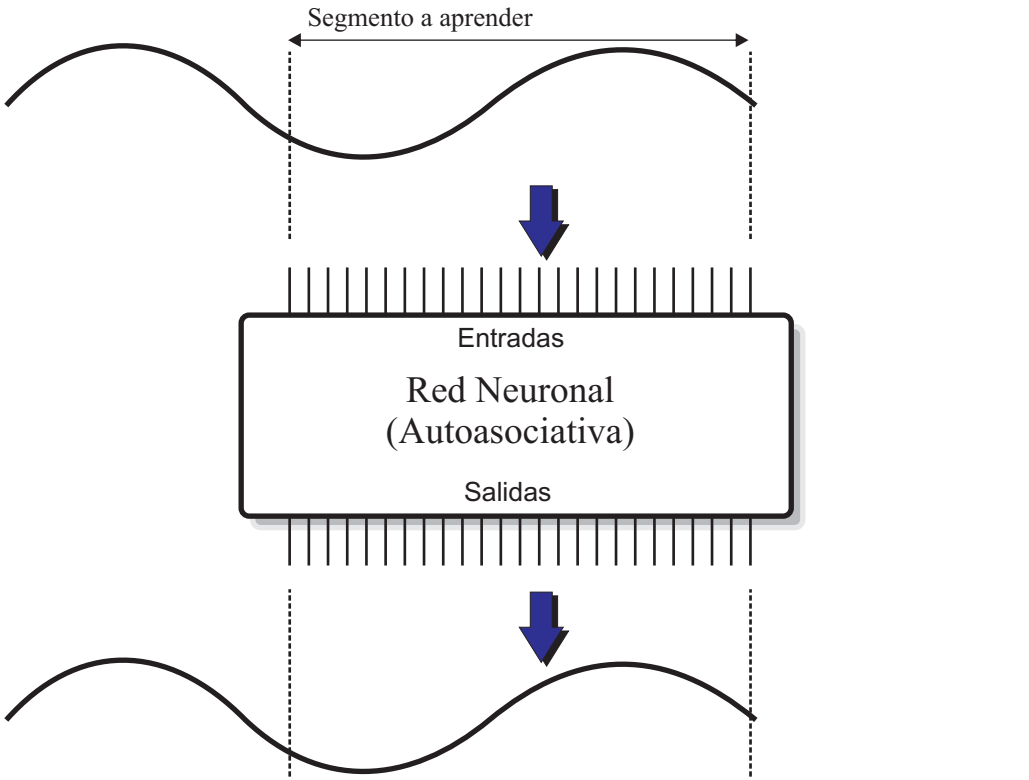
4. Reducción de Ruido(1).

Tip:

Es posible entrenar una red neural usando una onda senoidal pura con distintas fases. Si después se le muestra la red una onda senoidal contaminada con ruido, la red tratará de eliminar el ruido. Los resultados de esta sección son para ilustrar la operación de una red neuronal, sin embargo, otras técnicas de procesamiento digital de señales pueden ser más apropiadas para reducción de ruido.

Entrenamiento con una onda senoidal a distintas fases

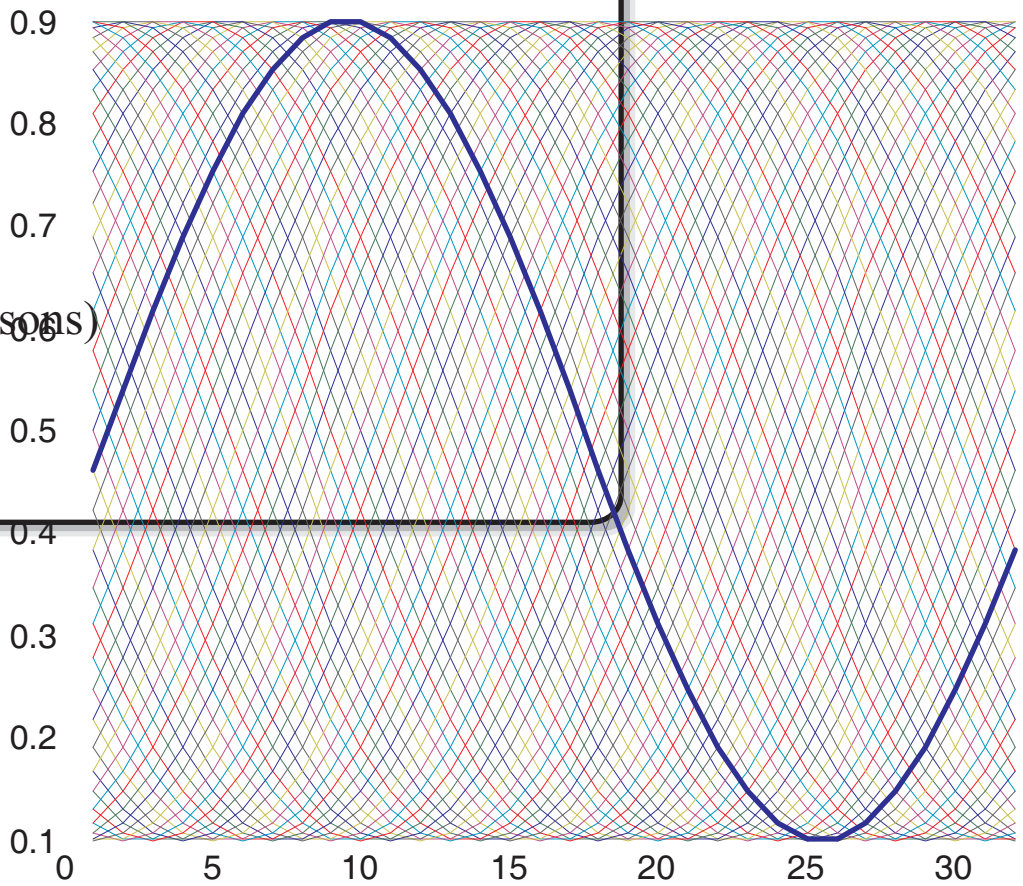




4. Creación de un Training Set usando Matlab (2).

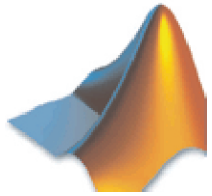
```
% Crear el Training Set para aprender el sin(x)
clear;
LESSON_COUNT = 64;
IN_COUNT = 32;
RANGE = 2.0*pi;
IN_DELTA = RANGE/IN_COUNT;
SET_DELTA = RANGE/LESSON_COUNT;
P = zeros(IN_COUNT, LESSON_COUNT);
PN = zeros(IN_COUNT, LESSON_COUNT);
for i=1 : LESSON_COUNT
    fase = (i-1)*SET_DELTA;
    for j=1 : IN_COUNT;
        P(j, i) = sin(fase+(j-1)*IN_DELTA);
    end
end
size(PN) % 32 rows(inputs)and 64 cols (lessons)
PN = (P + 1) .* (0.8/2)+0.1;
plot(PN);
```

Matlab

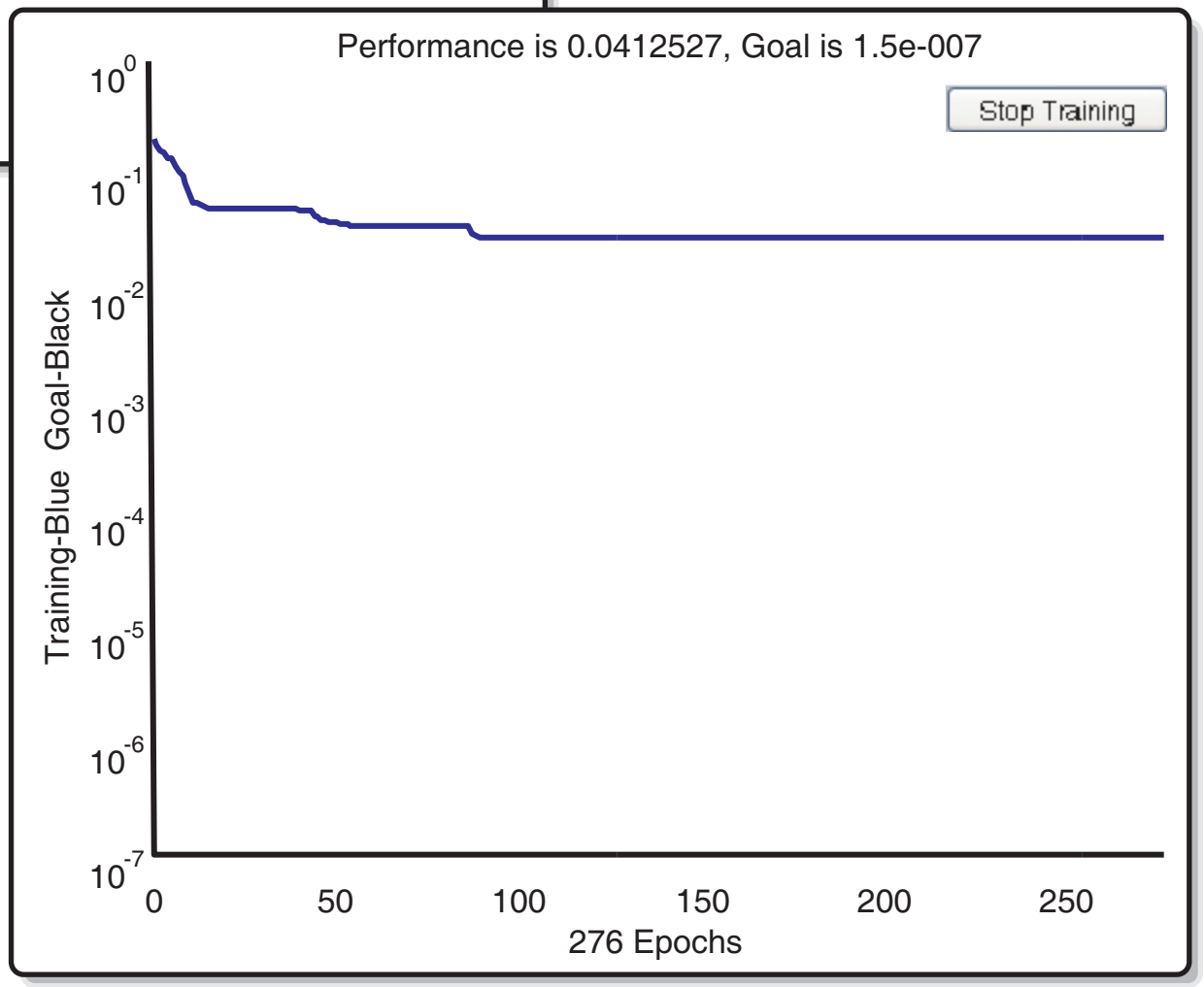


```
%Crear la matriz de rangos de entrada
RI = zeros(IN_COUNT, 2);
for i=1: IN_COUNT
    RI(i, 1) = 0.1;
    RI(i, 2) = 0.9;
end
%  Crear y entrenar la red
net = newff(RI, [5, IN_COUNT], {'logsig', 'logsig'}, 'traingdx');
net.trainParam.goal = 0.00000015;
net.trainParam.epochs = 700;
net = train(net, PN, PN);
```

Matlab



4. Entrenamiento auto asociativo para la señal seno en Matlab (3).

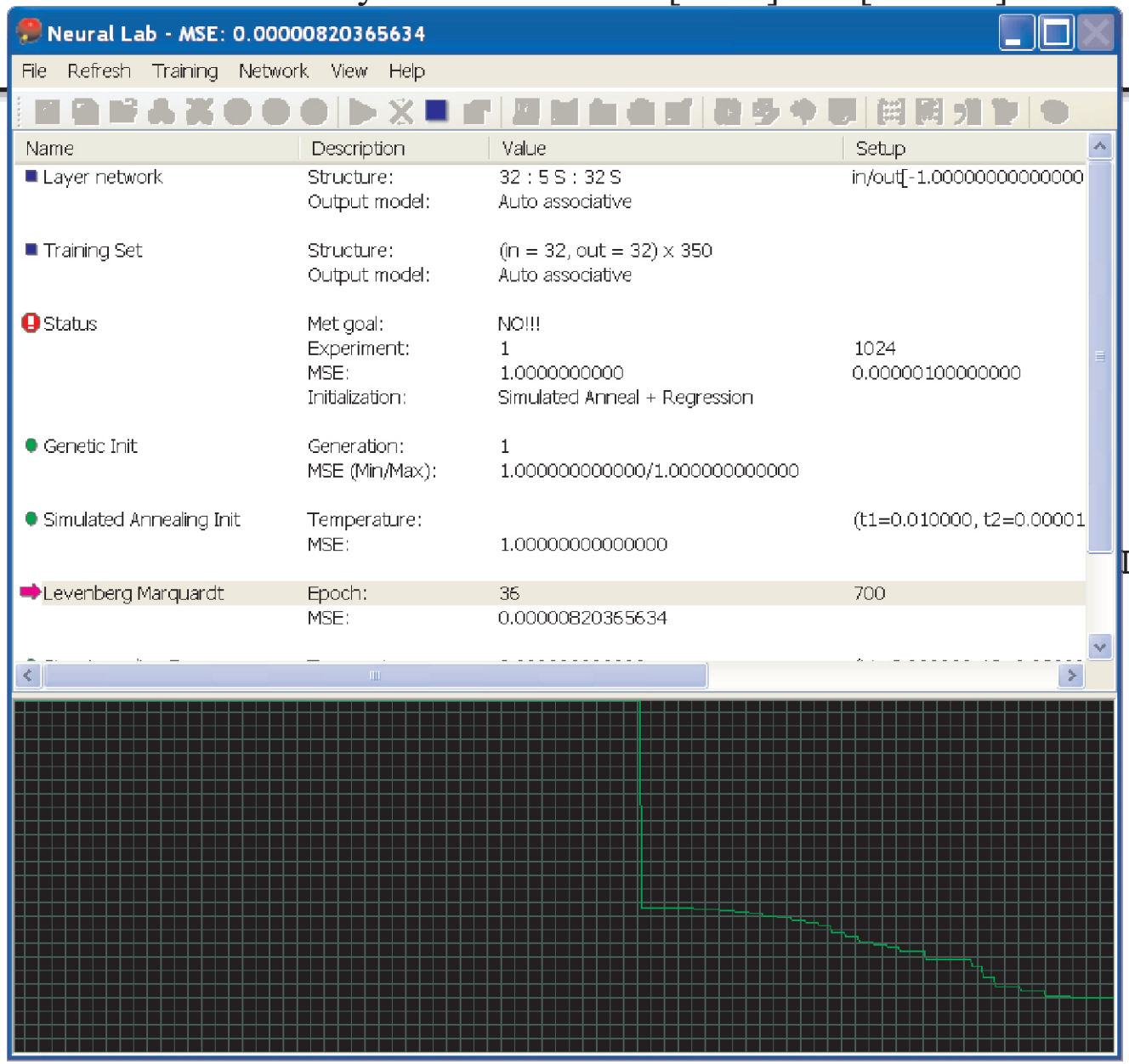


4. Una red auto asociativa en Neural Lab (4).

Actividad:

Usando Neural Lab cree una red auto asociativa para la señal senoidal, usa una red 64 : 20S : 64S con un training set de 64 entradas y 350 training cases. No se olvide es escalar las entradas y las salidas desde [-1 1] a [0.1 0.9].

Neural Lab

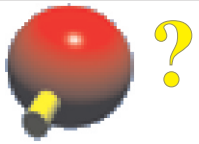


Fifth Mexican International Conference on Artificial Intelligence

Dr. Sergio Ledesma

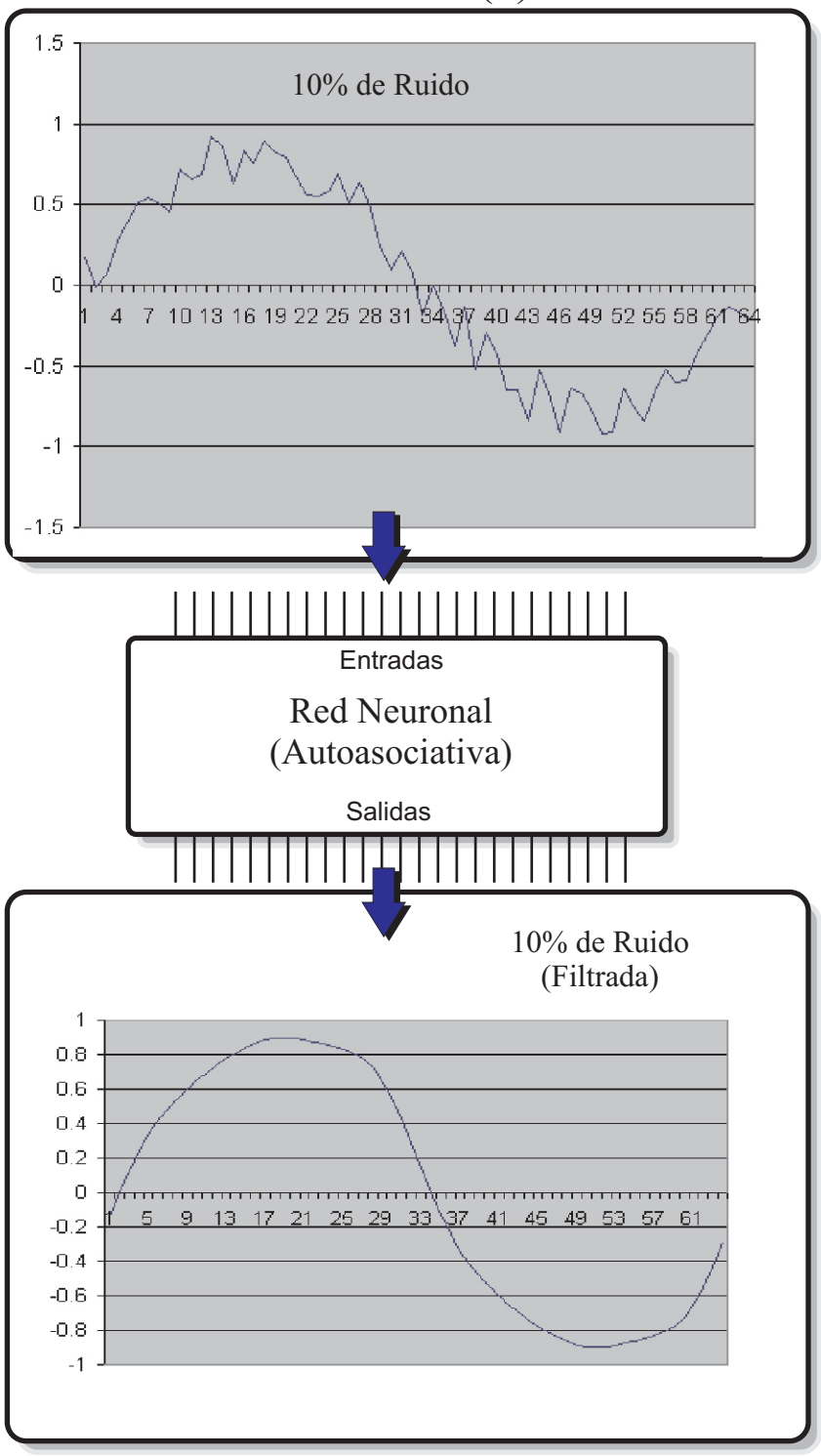
4. Una red auto asociativa en Neural Lab eliminando ruido (5).

Actividad:



Neural Lab

Una vez completado el entrenamiento aplique una señal senoidal contaminada con ruido blanco y observe la salida de la red auto asociativa.

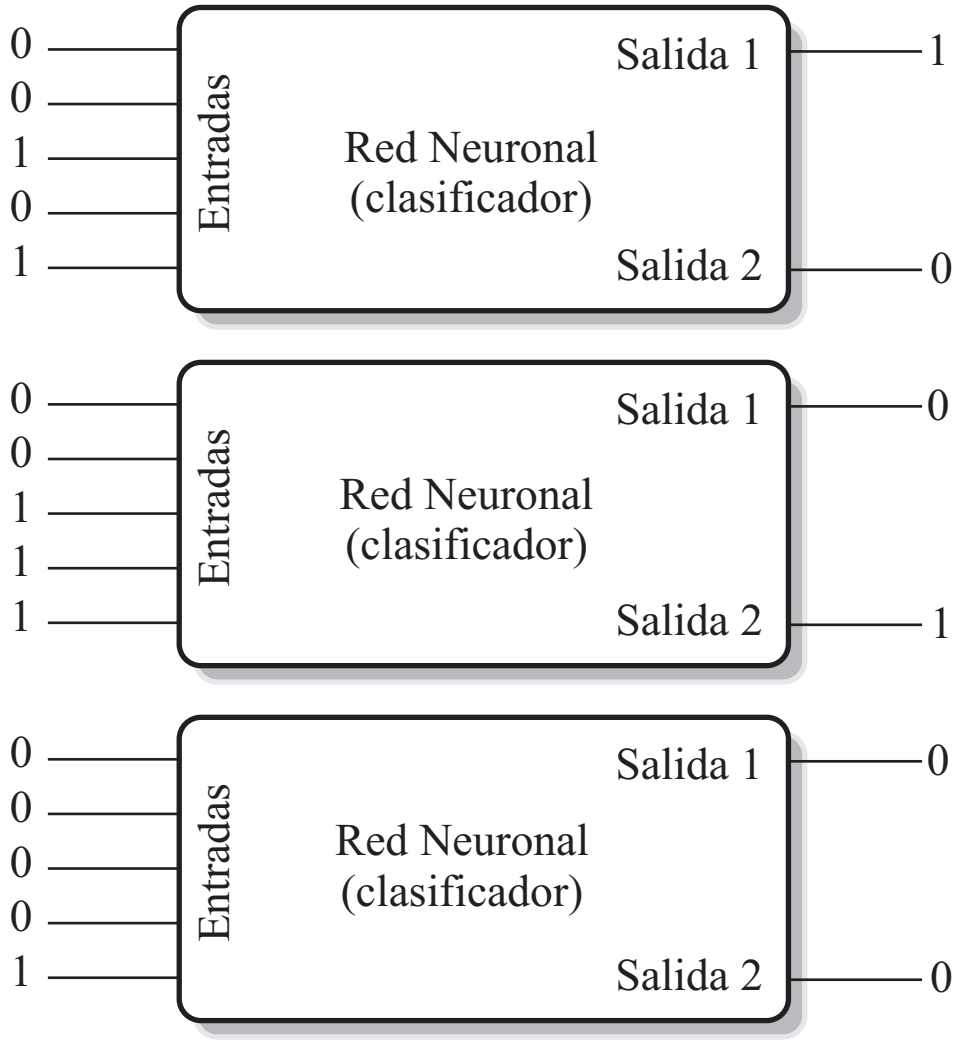


5. Clasificación (1).

Definición:

Clasificación es el proceso de identificación de un objeto dentro de un conjunto posible de resultados. Una red puede ser entrenada para identificar y separar distintos tipos de objetos. Estos objetos pueden ser: números, imágenes, sonidos, señales, etc.

Ejemplo de un clasificador: la red neuronal identifica los números del 1 al 31 que son divisibles entre 5 y entre 7.



5. Evaluando la calidad de un clasificador (2).

Definición:

Una matriz de confusión se utiliza en clasificación para indicar la calidad de la solución. Si la red tiene un número m de salidas, esta matriz tiene m renglones y m+1 columnas. Esta matriz indica cuantos elementos fueron clasificados correctamente. Cuando la red no comete errores la matriz de confusión es diagonal.

Tip:

La matriz de confusión debe tener idealmente un sólo elemento distinto de cero en cada renglón. En caso de que la matriz de confusión contenga elementos distintos de cero fuera de la diagonal principal, esto indica que la red confundió un objeto de un tipo con otro tipo de objeto. Por otro lado, la última columna de esta matriz, (Reject) indica los elementos que no pudieron clasificarse, por lo que debe contener sólo ceros.

Matriz de Confusión sin Errores

	out 1	out 2	Reject
out 1	6	0	0
out 2	0	11	0

Matriz de Confusión con Errores

	out 1	out 2	Reject
out 1	4	1	1
out 2	0	8	3

5. Usando una red neuronal como clasificador (3).

Actividad:

El archivo ClaPri1220.csv contiene una lista de los números del 1 al 511 en formato binario. La última columna es el número de la neurona activa cuando esa entrada es aplicada a una red neuronal clasificadora. Usando Neural Lab entrene la red hasta que no obtenga ningún error. Use una red neuronal con una estructura 9 : 25S : 3S

Neural Lab

	A	B	C	D	E	F	G	H	I	J
1	0	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	0	1	0	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	0	1	1
6	0	0	0	0	0	0	1	1	0	0
7	0	0	0	0	0	0	1	1	1	1
8	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	1	0	0	1	0
10	0	0	0	0	0	1	0	1	0	0
11	0	0	0	0	0	1	0	1	1	1
12	0	0	0	0	0	1	1	0	0	2
13	0	0	0	0	0	1	1	0	1	1
14	0	0	0	0	0	1	1	1	0	0
15	0	0	0	0	0	1	1	1	1	0
16	0	0	0	0	1	0	0	0	0	0
17	0	0	0	0	1	0	0	0	1	1
18	0	0	0	0	1	0	0	1	0	0
19	0	0	0	0	1	0	0	1	1	1
20	0	0	0	0	1	0	1	0	0	3
21	0	0	0	0	1	0	1	0	1	0
22	0	0	0	0	1	0	1	1	0	0
23	0	0	0	0	1	0	1	1	1	1
24	0	0	0	0	1	1	0	0	0	2
25	0	0	0	0	1	1	0	0	1	0
26	0	0	0	0	1	1	0	1	0	0
27	0	0	0	0	1	1	0	1	1	0
28	0	0	0	0	1	1	1	0	0	0
29	0	0	0	0	1	1	1	0	1	1
30	0	0	0	0	1	1	1	1	0	0
31	0	0	0	0	1	1	1	1	1	1

	A	B	C	D	E	F	G	H	I	J
32	0	0	0	1	0	0	0	0	0	0
33	0	0	0	1	0	0	0	0	1	0
34	0	0	0	1	0	0	0	1	0	0
35	0	0	0	1	0	0	0	1	1	0
36	0	0	0	1	0	0	1	0	0	2
37	0	0	0	1	0	0	1	0	1	1
38	0	0	0	1	0	0	1	1	0	0
39	0	0	0	1	0	0	1	1	1	0
40	0	0	0	1	0	1	0	0	0	3
41	0	0	0	1	0	1	0	0	1	1
42	0	0	0	1	0	1	0	1	0	0
43	0	0	0	1	0	1	0	1	1	1
44	0	0	0	1	0	1	1	0	0	0
45	0	0	0	1	0	1	1	0	1	0
46	0	0	0	1	0	1	1	1	0	0
47	0	0	0	1	0	1	1	1	1	1
48	0	0	0	1	1	0	0	0	0	2
49	0	0	0	1	1	0	0	0	1	0
50	0	0	0	1	1	0	0	1	0	0
51	0	0	0	1	1	0	0	1	1	0
52	0	0	0	1	1	0	1	0	0	0
53	0	0	0	1	1	0	1	0	1	1
54	0	0	0	1	1	0	1	1	0	0
55	0	0	0	1	1	0	1	1	1	0
56	0	0	0	1	1	1	0	0	0	0
57	0	0	0	1	1	1	0	0	1	0
58	0	0	0	1	1	1	0	1	0	0
59	0	0	0	1	1	1	0	1	1	1
60	0	0	0	1	1	1	1	0	0	3
61	0	0	0	1	1	1	1	0	1	1
62	0	0	0	1	1	1	1	1	0	0

	A	B	C	D	E	F	G	H	I	J
332	1	0	1	0	0	1	1	0	0	0
333	1	0	1	0	0	1	1	0	1	0
334	1	0	1	0	0	1	1	1	0	0
335	1	0	1	0	0	1	1	1	1	0
336	1	0	1	0	1	0	0	0	0	2
337	1	0	1	0	1	0	0	0	1	1
338	1	0	1	0	1	0	0	1	0	0
339	1	0	1	0	1	0	0	1	1	0
340	1	0	1	0	1	0	1	0	0	3
341	1	0	1	0	1	0	1	0	1	0
342	1	0	1	0	1	0	1	1	0	0
343	1	0	1	0	1	0	1	1	1	0
344	1	0	1	0	1	1	0	0	0	0
345	1	0	1	0	1	1	0	0	1	0
346	1	0	1	0	1	1	0	1	0	0
347	1	0	1	0	1	1	0	1	1	1
348	1	0	1	0	1	1	1	0	0	2
349	1	0	1	0	1	1	1	0	1	1
350	1	0	1	0	1	1	1	1	0	0
351	1	0	1	0	1	1	1	1	1	0
352	1	0	1	1	0	0	0	0	0	0
353	1	0	1	1	0	0	0	0	1	1
354	1	0	1	1	0	0	0	1	0	0
355	1	0	1	1	0	0	0	1	1	0
356	1	0	1	1	0	0	1	0	0	0
357	1	0	1	1	0	0	1	0	1	0
358	1	0	1	1	0	0	1	1	0	0
359	1	0	1	1	0	0	1	1	1	1
360	1	0	1	1	0	1	0	0	0	3
361	1	0	1	1	0	1	0	0	1	0
362	1	0	1	1	0	1	0	1	0	0

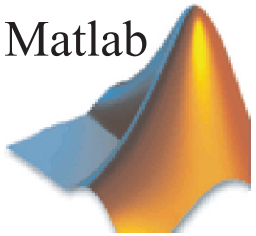
	A	B	C	D	E	F	G	H	I	J
482	1	1	1	1	0	0	0	1	0	0
483	1	1	1	1	0	0	0	1	1	0
484	1	1	1	1	0	0	1	0	0	0
485	1	1	1	1	0	0	1	0	1	0
486	1	1	1	1	0	0	1	1	0	0
487	1	1	1	1	0	0	1	1	1	1
488	1	1	1	1	0	1	0	0	0	0
489	1	1	1	1	0	1	0	0	1	0
490	1	1	1	1	0	1	0	1	0	0
491	1	1	1	1	0	1	0	1	1	1
492	1	1	1	1	0	1	1	0	0	2
493	1	1	1	1	0	1	1	0	1	0
494	1	1	1	1	0	1	1	1	0	0
495	1	1	1	1	0	1	1	1	1	0
496	1	1	1	1	1	0	0	0	0	0
497	1	1	1	1	1	0	0	0	1	0
498	1	1	1	1	1	0	0	1	0	0
499	1	1	1	1	1	0	0	1	1	1
500	1	1	1	1	1	0	1	0	0	3
501	1	1	1	1	1	0	1	0	1	0
502	1	1	1	1	1	0	1	1	0	0
503	1	1	1	1	1	0	1	1	1	1
504	1	1	1	1	1	1	0	0	0	2
505	1	1	1	1	1	1	0	0	1	0
506	1	1	1	1	1	1	0	1	0	0
507	1	1	1	1	1	1	0	1	1	0
508	1	1	1	1	1	1	1	0	0	0
509	1	1	1	1	1	1	1	0	1	1
510	1	1	1	1	1	1	1	1	0	0
511	1	1	1	1	1	1	1	1	1	0
512										

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

5. Usando Matlab para crear el training set de la actividad anterior (4).

```
s = 1:511:
a = zeros(511, 1);
a(find(mod(s, 12) ==0)) = 2;
a(find(mod(s, 20)==0))=3;
a(primes(511))=1;
a(1) = 1;
m[(double(dec2bin(s))-'0'), a];
csvwrite('TestPri1220.csv', m);
```

Matlab



6. El Neuro controlador (1).

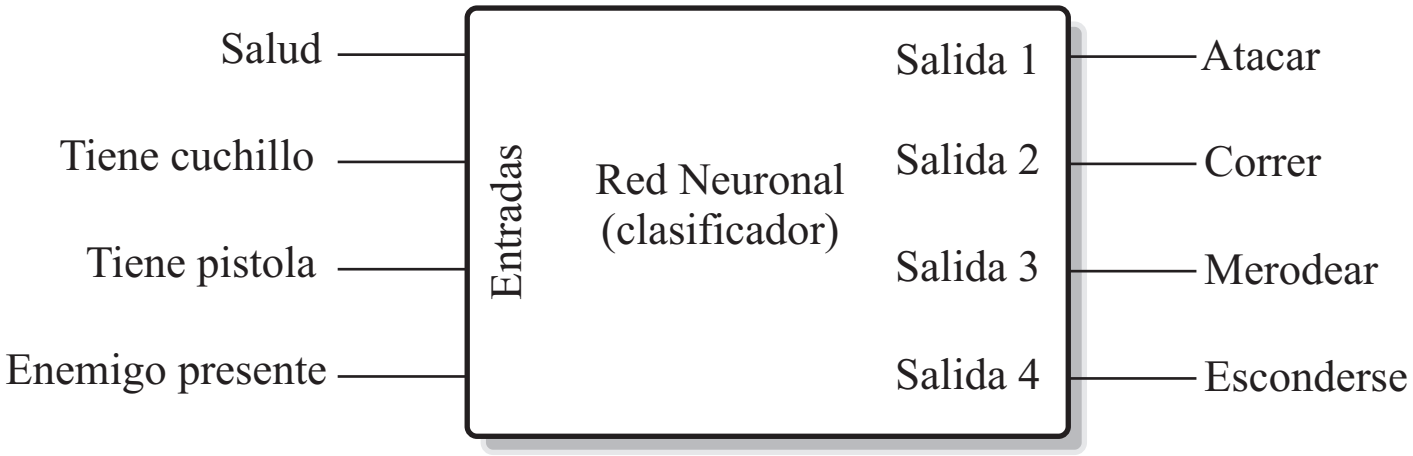
Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

Definición:

Es posible entrenar una red neuronal para que reaccione a situaciones conocidas. Sin embargo, es posible entrenar una red neuronal para que suguiera acciones en situaciones en las que no recibió entrenamiento.

Actividad

La red neuronal decide la acción a tomar de un guerrero virtual en un caso en el que no recibió entrenamiento. Complete la tabla mostrada en la siguiente hoja usando el sentido común.



6. Lista de acciones de un guerrero virtual operado por un neurocontrolador (2).

Salud	Cuchillo	Pistola	Enemigos	Acción
Saludable	No	No	0	
Saludable	No	No	1	
Saludable	No	Si	1	
Saludable	No	Si	2	
Saludable	Si	No	2	Esconderse
Saludable	Si	No	1	
Media	No	No	0	
Media	No	No	1	
Media	No	Si	1	
Media	No	Si	2	
Media	Si	No	2	
Media	Si	No	1	
Moribundo	No	No	0	
Moribundo	No	No	1	
Moribundo	No	Si	1	
Moribundo	No	Si	2	Correr
Moribundo	Si	No	2	Correr
Moribundo	Si	No	1	Esconderse

6. Resultados del neurocontrolador (3).

Actividad

Usando Microsoft Excel cree el Training Set como se muestra, llame al archivo. TraGuerrero.csv. Una vez creado el archivo utilize Neural Lab para crear una red que aprenda el comportamiento del guerrero con una estructura 4 : 4S : 4S. Complete la matriz de confusión.

	out 1	out 2	out 3	out 4	Reject
out 1					
out 2					
out 3					
out 4					

Actividad

Usando el sentido común complete la tabla. Luego cree el archivo TestGuerrero.csv con la información de la tabla mostrada debajo. Utilice, entonces, Neural Lab para clasificar el archivo.

Fifth Mexican
International Conference on
Artificial Intelligence
Dr. Sergio Ledesma

Salud	Cuchillo	Pistola	Enemigos	Acción (Sentido común)	Acción (Neuro Controlador)
Saludable	Si	Si	1		
Media	Si	Si	2		
Moribundo	No	No	0		
Moribundo	Si	Si	1		
Media	No	Si	3		
Media	Si	No	3		
Moribundo	Si	No	3		

Apéndice

Comandos Básicos para Redes Neuronales en Matlab.

Usted puede crear un archivo *.m para ejecutar una secuencia de comandos

Descripción	
Crear la red	net = newff([min1 max1; min2 max2; ...], [3, ..., 4], {'logsig',...}, 'traingdx');
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)

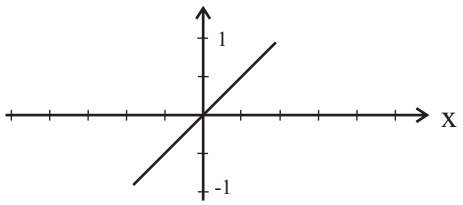
rango de entradas

Numero de Neuronas En Nivel Escóndido 1

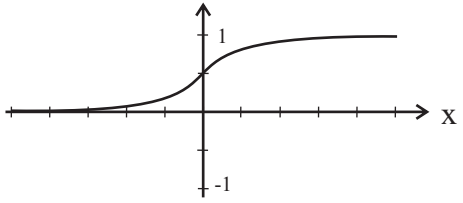
Numero de Neuronas de Salidas

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

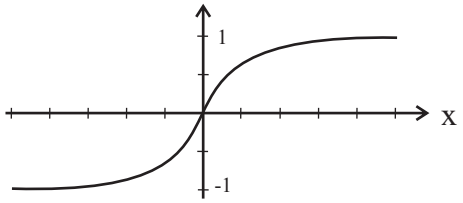
f(x) = purelin(x)



f(x) = logsig(x)



f(x) = tansig(x)



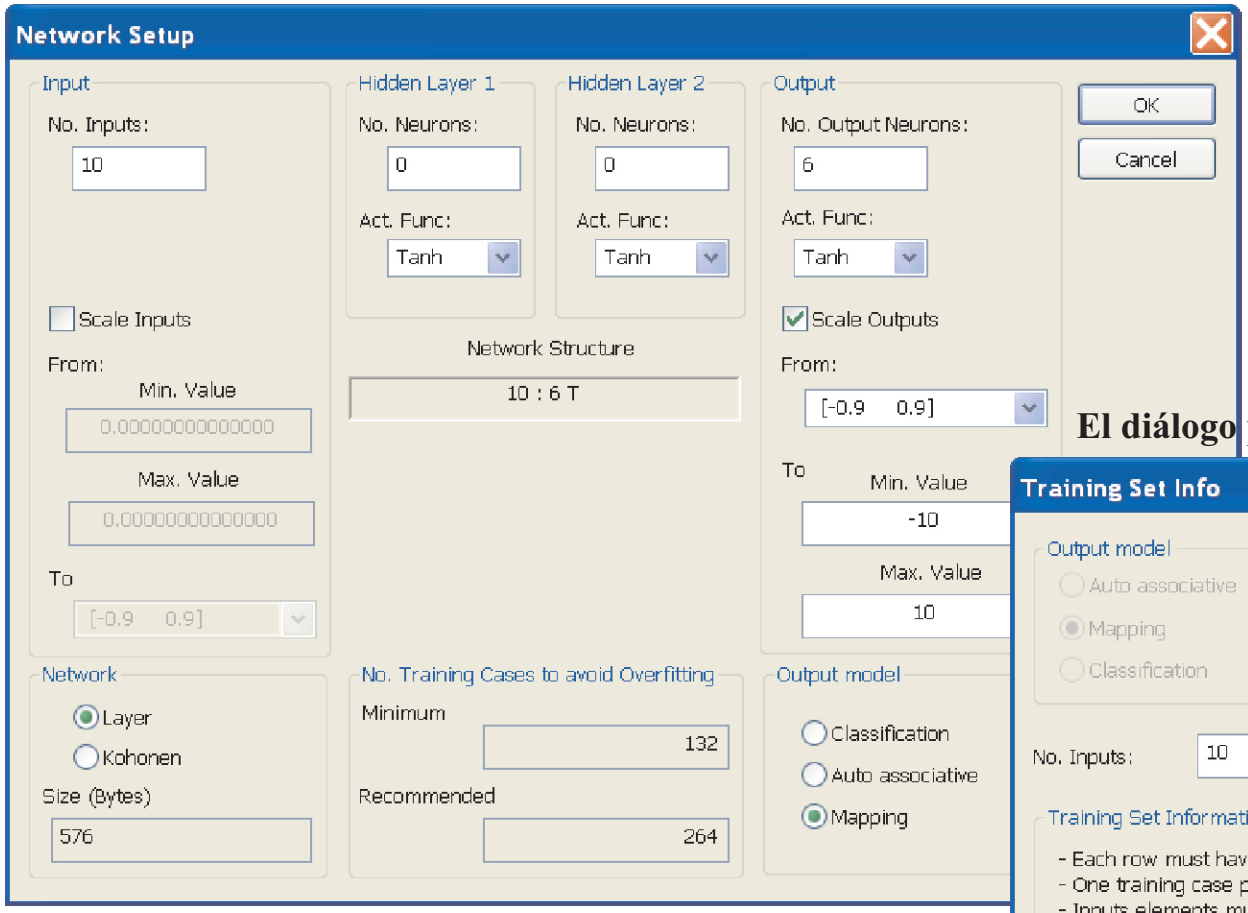
trainbfg: BFGS quasi-Newton backpr.
trainbr: Bayesian regularization
traincgb: Powell-Beale conj. grad. backpr.
traincgf: Fletcher-Powell conj. grad. backpr.
traincgp: Polak-Ribiere conj. grad. backpr.
traingd: Gradient descent backprop.
traingda: Gradient descent adaptive
traingdm: Gradient descent 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.

El Objeto Network

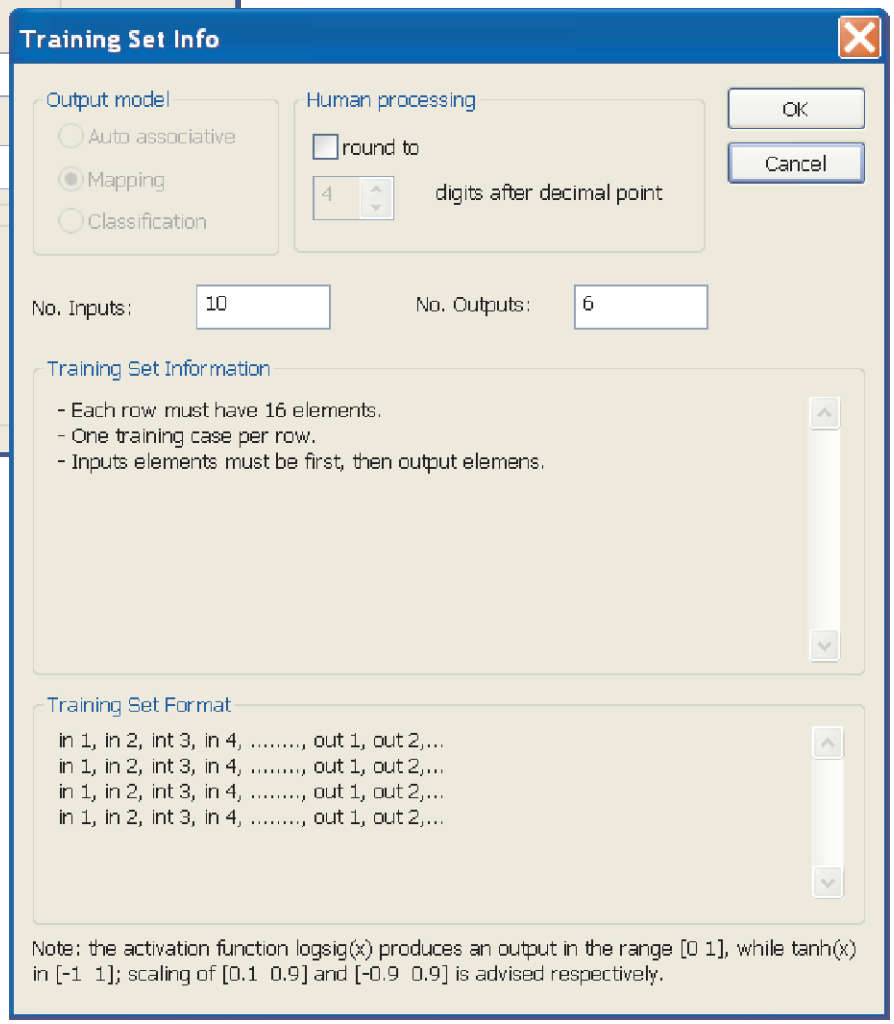
net.numInputs
net.numLayers
net.inputs
net.layers
net.outputs
net.targets
net.inputWeights
net.layerWeights
net.trainFcn
net.performFcn

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

El diálogo para crear una red en Neural Lab



El diálogo para crear un training set en Neural Lab



El diálogo para configurar el entrenamiento en Neural Lab