COMPUTER VISION

Machine Learning & Neural Networks

6.- Supervised Neural Networks: Multilayer Perceptron

by
Pascual Campoy
Grupo de Visión por Computador
U.P.M. - DISAM

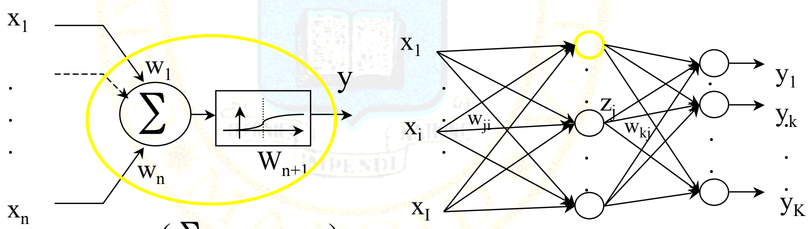
COMPUTER VISION

topics

- Artificial Neural Networks
- Perceptron and the MLP structure
- The back-propagation learning algorithm
- MLP features and drawbacks
- The auto-encoder

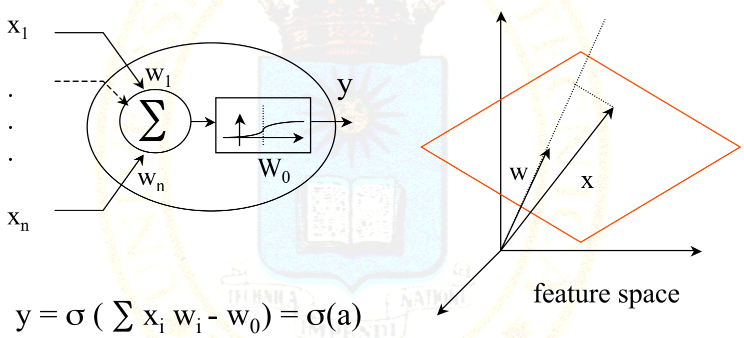
Artificial Neural Networks

- *"A net of simple, adaptable & interconnected units, having parallel processing capability, whose objective is to interact with the environment in a similar way as the natura neural network do"*



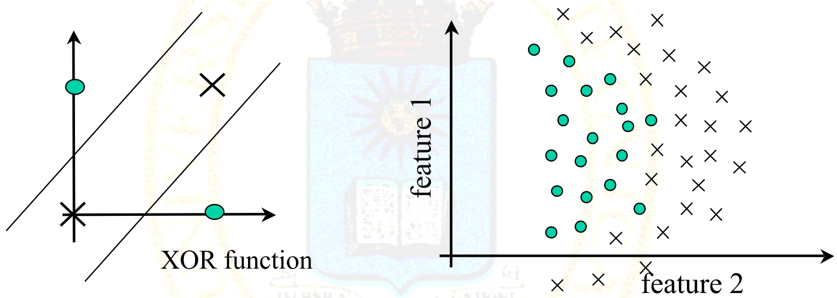
$$y = \sigma (\sum x_i w_i - w_0)$$

The perceptron: working principle

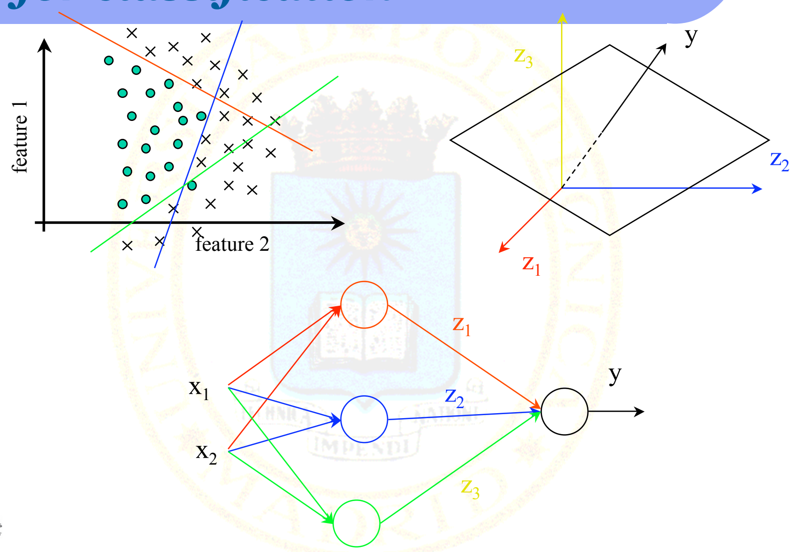


$$y = \sigma (\sum x_i w_i - w_0) = \sigma(a)$$

The perceptron for classification



(MLP): for classification



The multilayer Perceptron: Mathematical issues

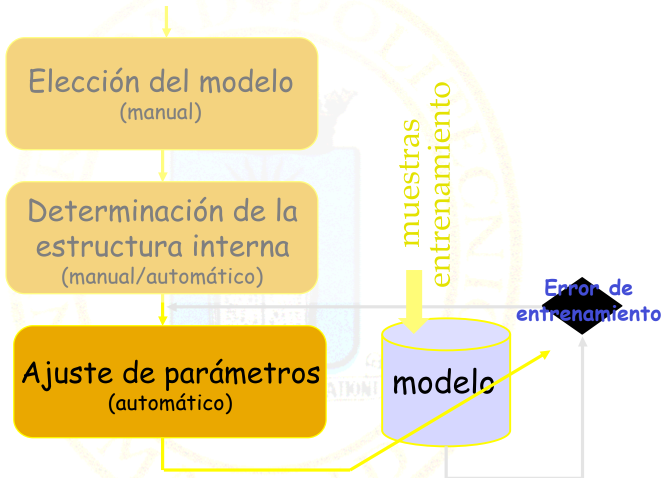
- Un MLP de dos capas puede representar cualquier función lógica con frontera convexa.
- Un MLP de tres capas puede representar cualquier función lógica con frontera arbitraria.

Un MLP de dos capas puede aproximar cualquier función continua con una precisión arbitraria.

topics

- *Artificial Neural Networks*
- *Perceptron and the MLP structure*
- *The back-propagation learning algorithm*
- *MLP features and drawbacks*
- *The auto-encoder*

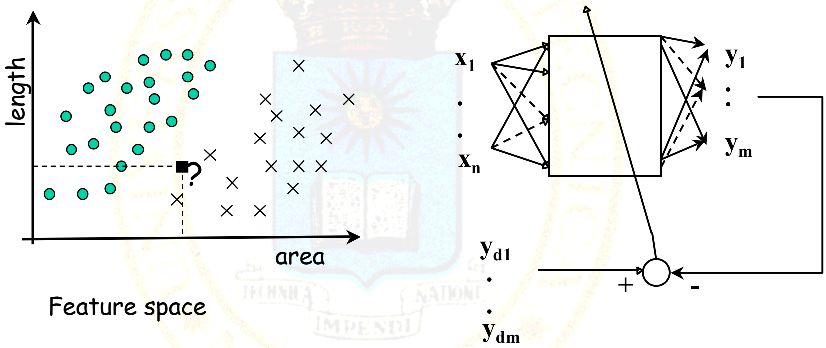
Building machine learning models: levels



Supervised learning

Supervised learining concept

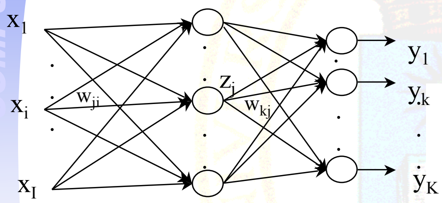
Working structure



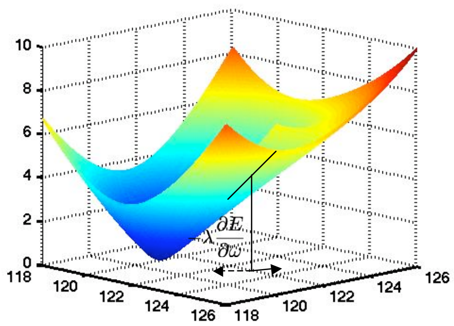
$R^n \Rightarrow R^m$ function generalitation

The back-propagation learning algorithm: working principle

$$E = \frac{1}{2}\sum_k (y_k^n - y_{dk}^n)^2 = \frac{1}{2}\sum_k (y_k^n(\omega_{kj}, \omega_{ji}, x_i) - y_{dk}^n)^2$$



$$\Delta\omega = -\lambda \frac{\partial E}{\partial \omega}$$



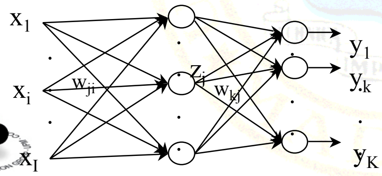
The back-propagation learning algorithm: equations

$$E = \frac{1}{2}\sum_k (y_k^n - y_{dk}^n)^2 = \frac{1}{2}\sum_k (y_k^n(\omega_{kj}, \omega_{ji}, x_i) - y_{dk}^n)^2$$

$$\boxed{\frac{\partial E}{\partial w_{kj}} = (y_k - y_{dk}) z_j}$$

$$\frac{\partial E}{\partial w_{ji}} = \sum_k (y_k - y_{dk}) \frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial w_{ji}} = \sum_k (y_k - y_{dk}) w_{kj} \frac{\partial z_j}{\partial a_j} x_i$$

$$\boxed{\frac{\partial E}{\partial w_{ji}} = \sum_k (y_k - y_{dk}) w_{kj} z_j (1 - z_j) x_i}$$



$$y_j = \frac{1}{1+e^{-a_j}} \Rightarrow \frac{\partial y_j}{\partial a_j} = \frac{e^{-a_j}}{(1+e^{-a_j})^2} = (1-y_j)y_j$$

$$y_j = tanh(a_j) \Rightarrow \frac{\partial y_j}{\partial a_j} = 1 - y_j^2$$

Matlab commands:

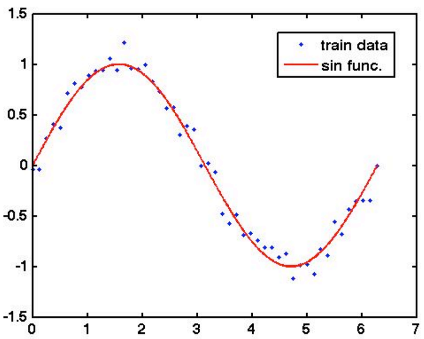
```
% MLP building
>> net = newff(minmax(p.valor),[nL1 noL],{'tansig' 'purelin'},'trainlm');

% MLP training
>> [net,tr]=train(net,p.valor,p.salida);

% answer
>> anst=sim(net,t.valor);
>> errortest=mse(t.salida-anst);
```


MLP for function generalization

```
% training data
Ntra=50; xe=linspace(0,2*pi,Ntra); %xe= 2*pi*rand(1,Ntra);
for i=1:Ntra
yd(i)=sin(xe(i))+normrnd(0,0.1);
end
% test data
Ntest=500;
xt=linspace(0,2*pi,numtest);
yt_gt=sin(xt);
for i=1:Ntest
yt(i)=yt_gt(i)+normrnd(0,0.1);
end
plot(xe,yd,'b.'); hold on;
plot(xt,yt,'r-');
```



MLP for function generalization

Using above mentioned data generation procedure:

Plot in the same figure the training set, the output of the MLP for the test set, and the underlying sin function. Evaluate the train error, the test error and the ground truth error. In the following cases:

a) Choosing an **adequate MLP** structure and training set

Compare and analyze the results:

b) Changing the **training parameters**:
   initial values, (# of epochs, optimization algorithm)

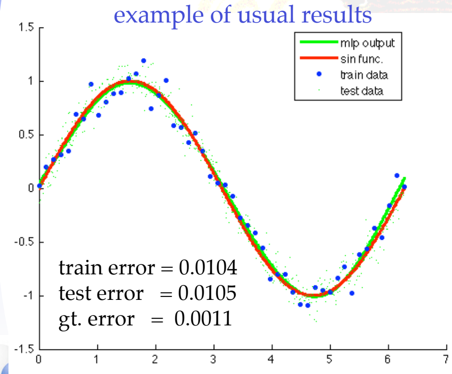
c) Changing the **training data**:
   # of samples
   order of samples, their representativiness

d) Changing the **net structure**:
   # of neurons

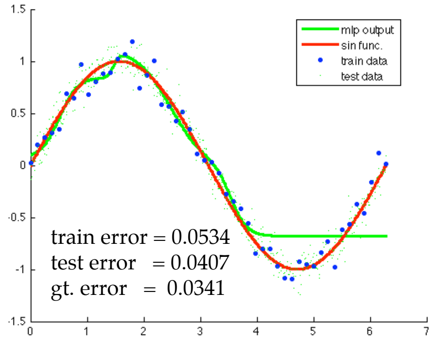
Results for exercise 6.1: a) b) changes of training parameters ...

- 50 training samples
- 4 neurons in hidden layer

example of usual results



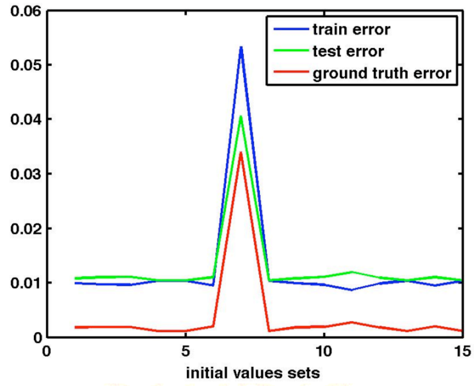
results for a bad initial values



COMPUTER VISION

Results for exercise 6.1: b) … changes of training parameters

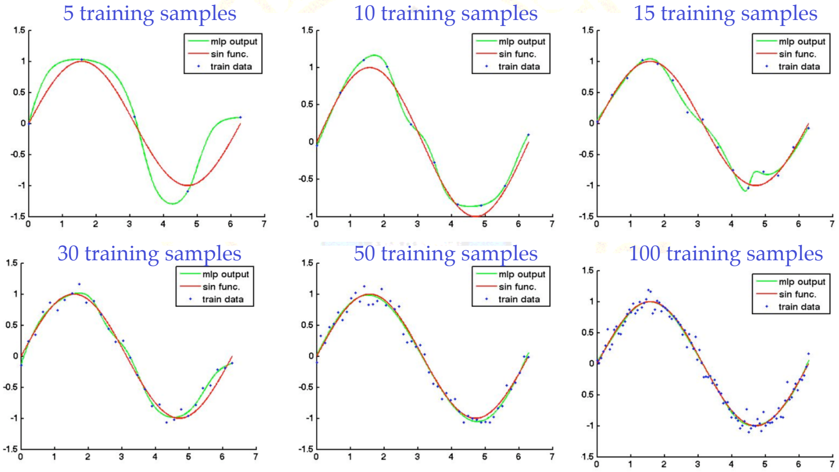
- 50 training samples
- 4 neurons in hidden layer



COMPUTER VISION

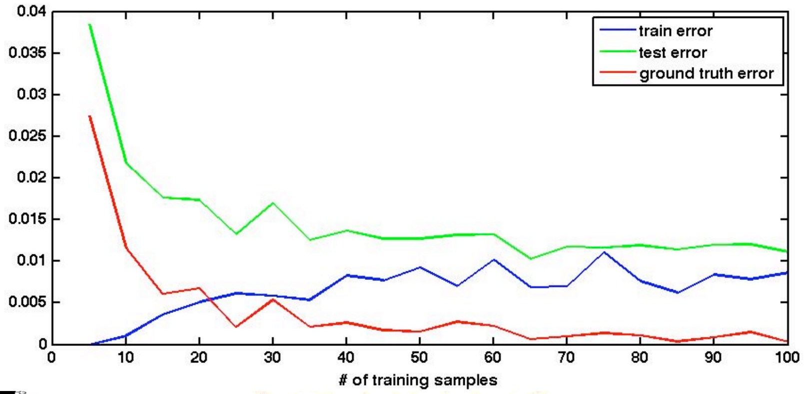
Results for exercise 6.1: c) changes in # of training samples …

- 4 neurons in hidden layer



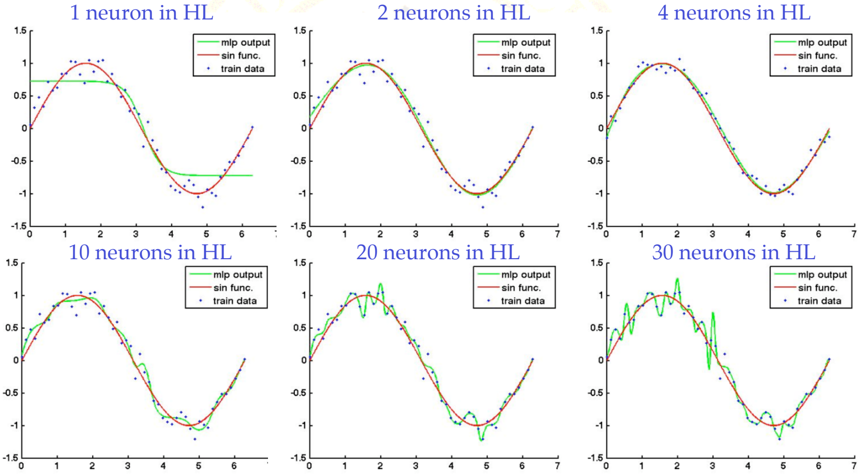
Results for exercise 6.1: c) … changes in # of training samples

- 4 neurons in hidden layer (mean error over 4 tries)



Results for exercise 6.1: d) changes in # neurons …

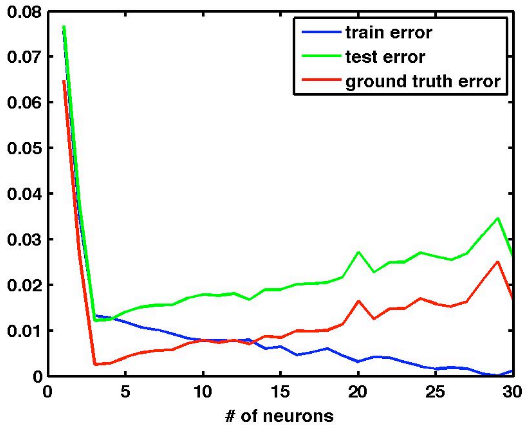
- 50 training samples



CVG-UPM

Results for exercise 6.1: d) … changes in # neurons

- 50 training samples (mean error over 4 tries)

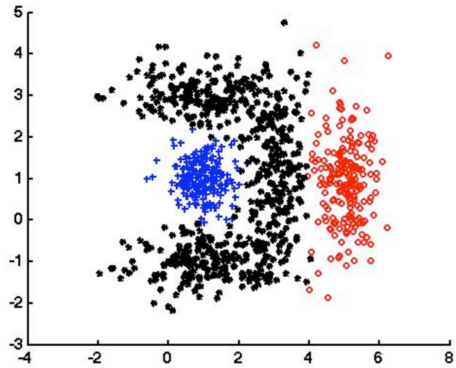


CVG-UPM

Exercise 6.2: MLP as a classifier

-The output is a discriminant function

>> load datos_2D_C3_S1.mat



P. Campoy Machine Learning and Neural Networks

CVG-UPM

COMPUTER VISION

Exercise 6.2: MLP as a classifier

Using the classified data: >> load datos_2D_C3_S1.mat
Evaluate the train error and the test error in following cases:

a) Choosing an **adequate** MLP structure, training set and test set. Plot the linear classification limits defined by each perceptron of the intermediate layer.

Compare and analyze the results:

b) Changing the **data set and the test set**:

c) Changing the **net structure** (i.e. # of neurons)

CVG-UPM

COMPUTER VISION

topics

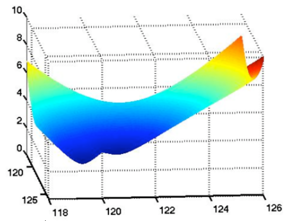
- ***Artificial Neural Networks***
- ***Perceptron and the MLP structure***
- ***The back-propagation learning algorithm***
- ~~***MLP features and drawbacks***~~
- ~~***The auto-encoder***~~

P. Campoy Machine Learning and Neural Networks

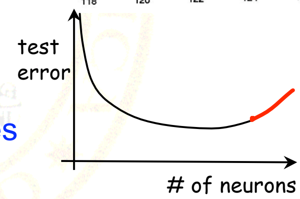
MLP features and drawbacks

- Learning by minimizing non-linear functions:
  - local minima
  - slow convergence

  (depending on initial values & minimization algorithm)



- Over-learning
- Extrapolation in non learned zones



topics

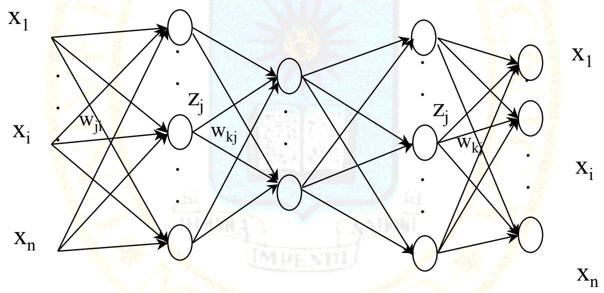
- Artificial Neural Networks
- Perceptron and the MLP structure
- The back-propagation learning algorithm
- MLP features and drawbacks
- The auto-encoder

CVG-UPM

Auto-encoder: MLP for dimensionality reduction

COMPUTER VISION

- *The desired output is the same as the input and there is a hiden layer having less neurons than dim(x)*



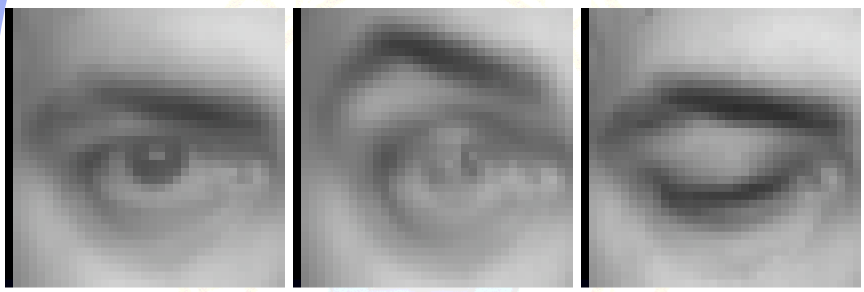
CVG-UPM

Example: auto-encoder for compression

COMPUTER VISION

original PCA 5 PCA 25 - MLP 5

Example: auto-encoder for synthesis



1 D (test 1) | 1 D (test 2) | 1 D (test 3) escaled

Auto-encoder: Matlab code

```
% Procesamiento con una MLP para compresioón (salida=entrada)
net=newff(minmax(p_entr),[floor((Dim+ndimred)/2),ndimred,floor((Di
   m+ndimred)/2),Dim],{'tansig' 'purelin' 'tansig' 'purelin'},
   'trainlm');
[net,tr]=train(net,p_entr,p_entr);

% Creación de una red mitad de la anterior que comprime los datos
netcompr=newff(minmax(p_entr),[floor((Dim+ndimred)/2),
   ndimred],{'tansig' 'purelin'},'trainlm');
netcompr.IW{1}=net.IW{1}; netcompr.LW{2,1}=net.LW{2,1};
netcompr.b{1}=net.b{1}; netcompr.b{2}=net.b{2};

%creación de una red que descomprime los datos
netdescompr=newff(minmax(p_compr),[floor((Dim+ndimred)/2),Dim],{'t
   ansig' 'purelin'}, 'trainlm');
netdescompr.IW{1}=net.LW{3,2}; netdescompr.LW{2,1}=net.LW{4,3};
netdescompr.b{1}=net.b{3};      netdescompr.b{2}=net.b{4};
```