UNIVERSITÀ DEGLI STUDI DI MILANO

Neural Networks for Classification

Enrique Muñoz Ballester

Dipartimento di Tecnologie dell'Informazione
via Bramante 65, 26013 Crema (CR), Italy
enrique.munoz@unimi.it

Material

- Download slides data and scripts:

https://homes.di.unimi.it/munoz/teaching.html

Classification

- Classification is one of the most frequently encountered decision making tasks of human activity.
- A classification problem occurs when an object needs to be assigned into a predefined group or class based on a number of observed attributes related to that object.

G.P. Zhang, "Neural networks for classification: a survey," in *IEEE Transactions on Systems, Man, and Cybernetics*, Part C: Applications and Reviews, vol.30, no.4, pp.451- 462, November 2000.

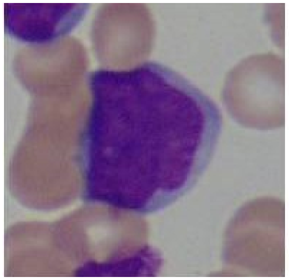
Classification with NN

- Neural networks have emerged as an important tool for classification.
- Advantages:
  - NN are data driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model
  - NN are universal functional approximators in that neural networks can approximate any function with arbitrary accuracy
  - NN are non-linear models, which makes them flexible in modeling real world complex relationships

G.P. Zhang, "Neural networks for classification: a survey," in *IEEE Transactions on Systems, Man, and Cybernetics*, Part C: Applications and Reviews, vol.30, no.4, pp.451- 462, November 2000.

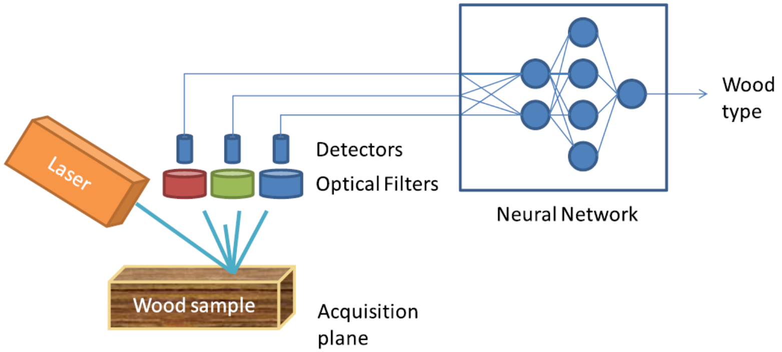
Examples of classification with NN

- Some our works
  - Acute Limphoblastic Leucemia
    - "healty cell" & "lymphoblast“



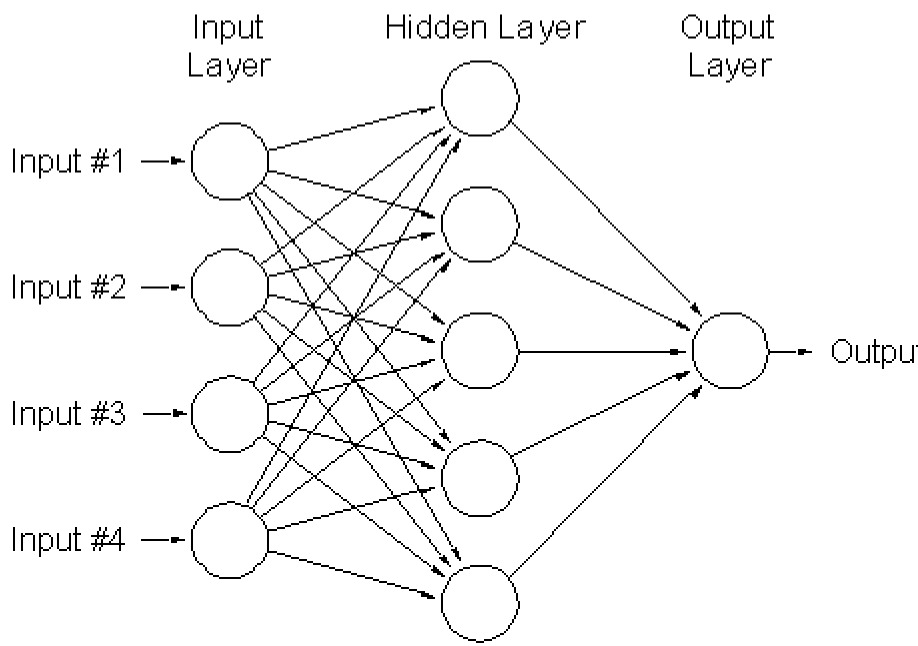
  - Wildfires
    - "Smoke frame" & "not smoke frame"

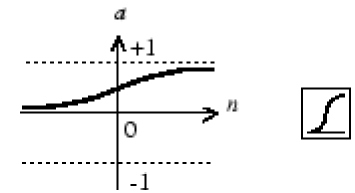
  - Wood
    - 21 classes



Classification with NN in Matlab

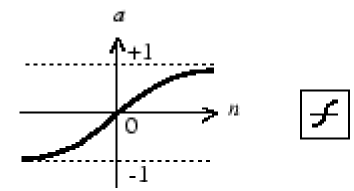
- We will use:
  - Neural Network Toolbox
  - Feedforward Neural Networks





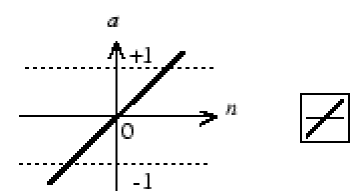
$a = logsig(n)$

Log-Sigmoid Transfer Function



$a = tansig(n)$

Tan-Sigmoid Transfer Function



$a = purelin(n)$

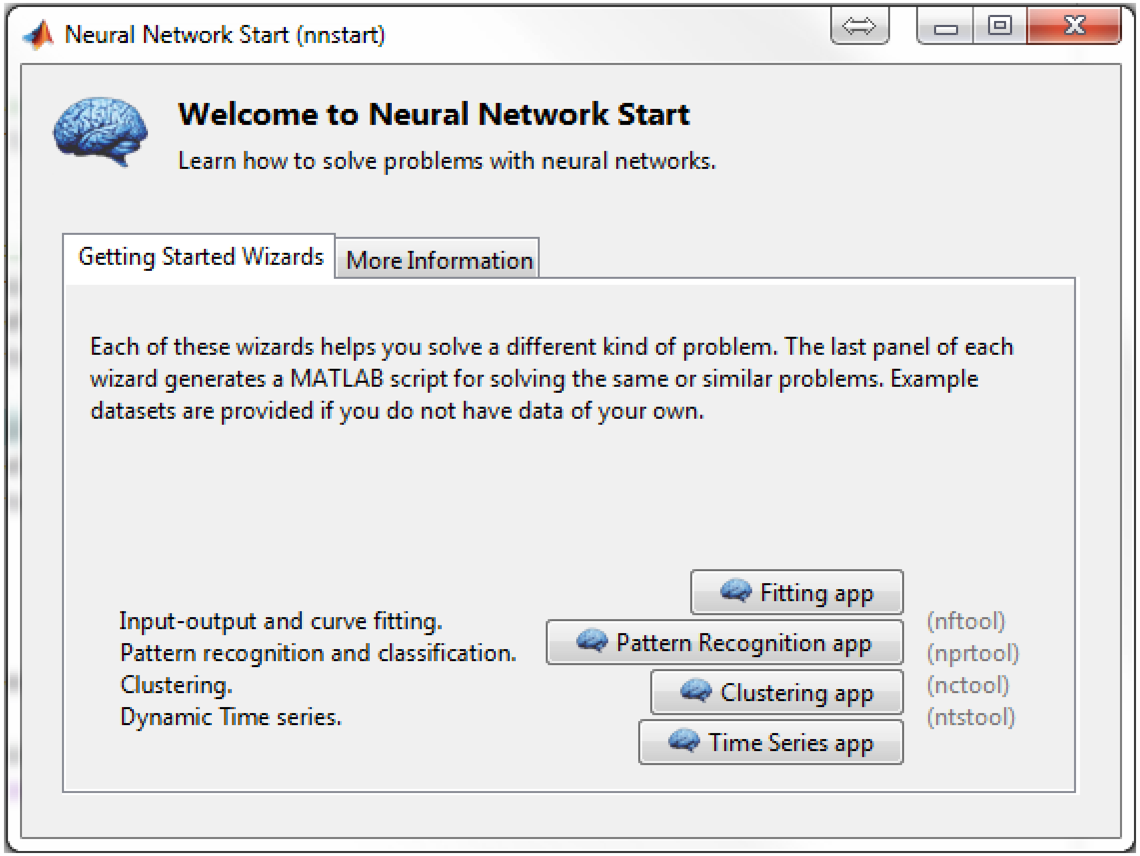
Linear Transfer Function

Neural networks in Matlab

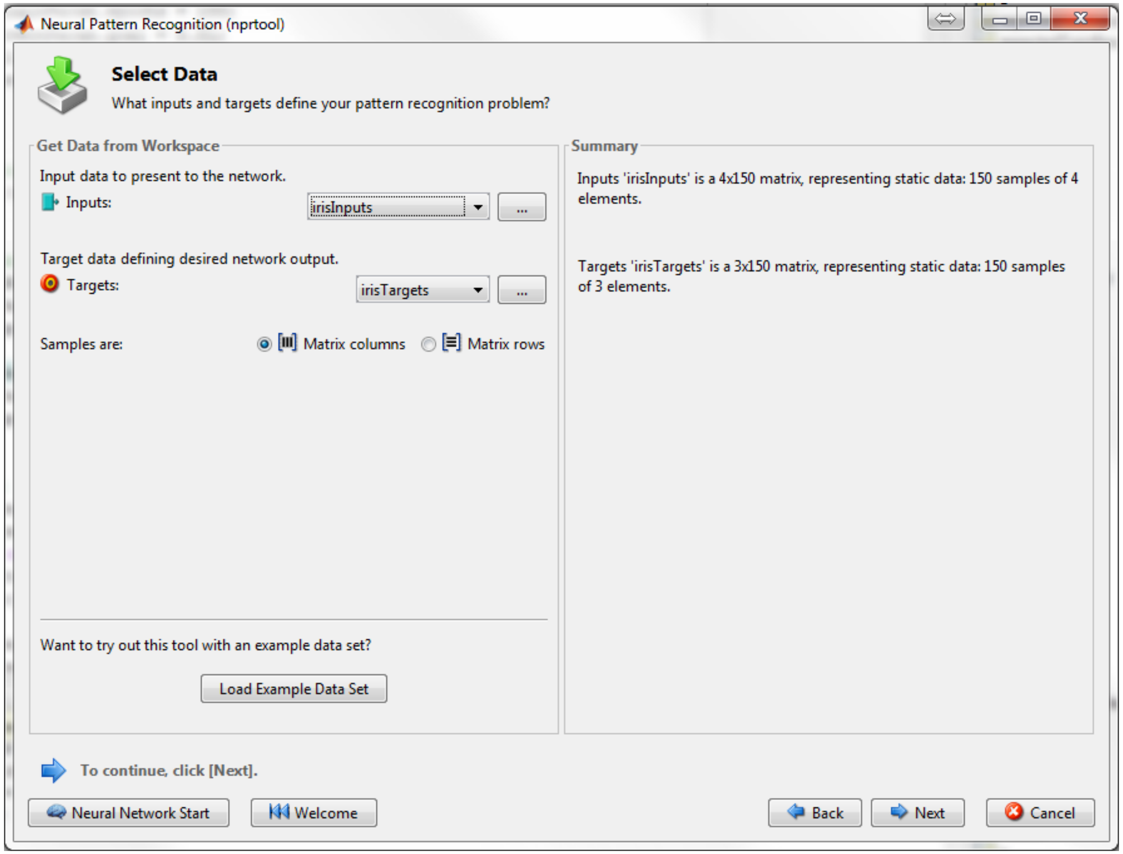
1. Loading data source
2. Selecting attributes required
3. Decide training, validation, and testing data
4. Data manipulations and Target generation
5. Neural Network creation (selection of network architecture) and initialisation
6. Network Training and Testing
7. Performance evaluation

Neural Network Pattern Recognition Tool: GUI

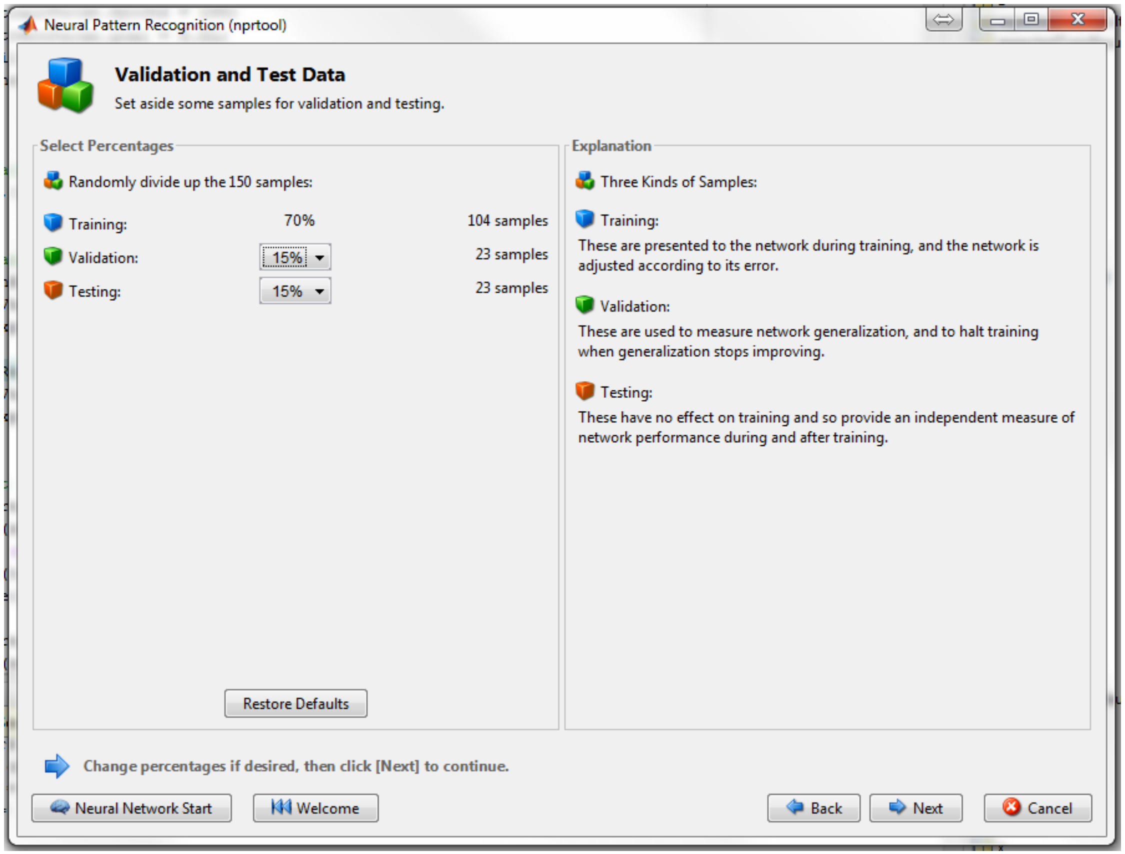
- nnstart



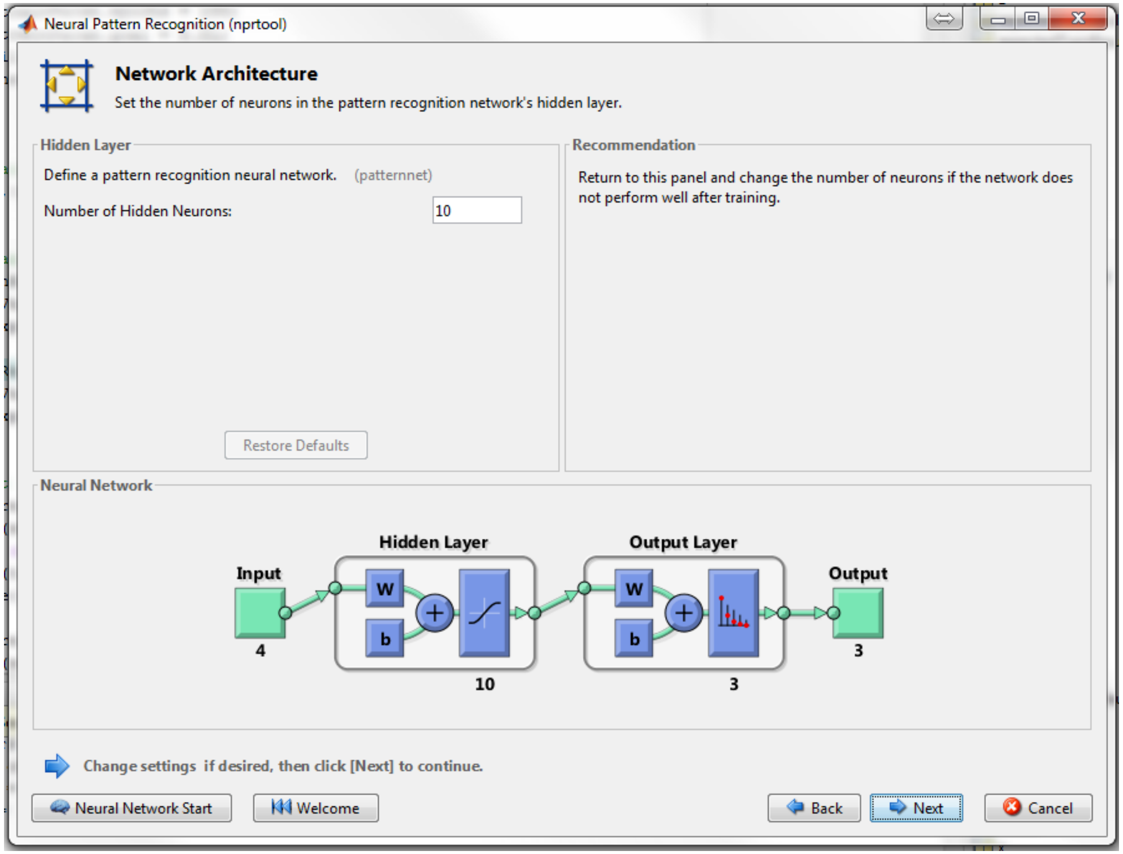
Neural Network Pattern Recognition Tool: GUI



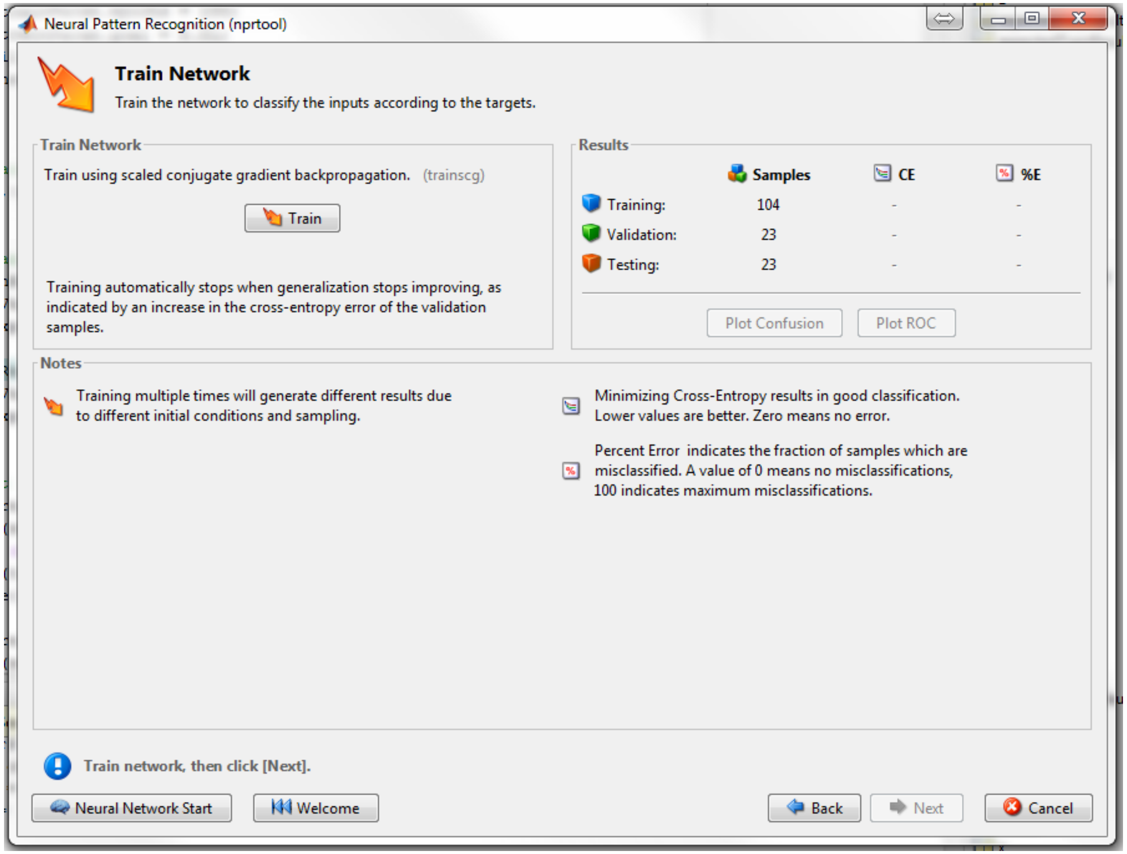
Neural Network Pattern Recognition Tool: GUI



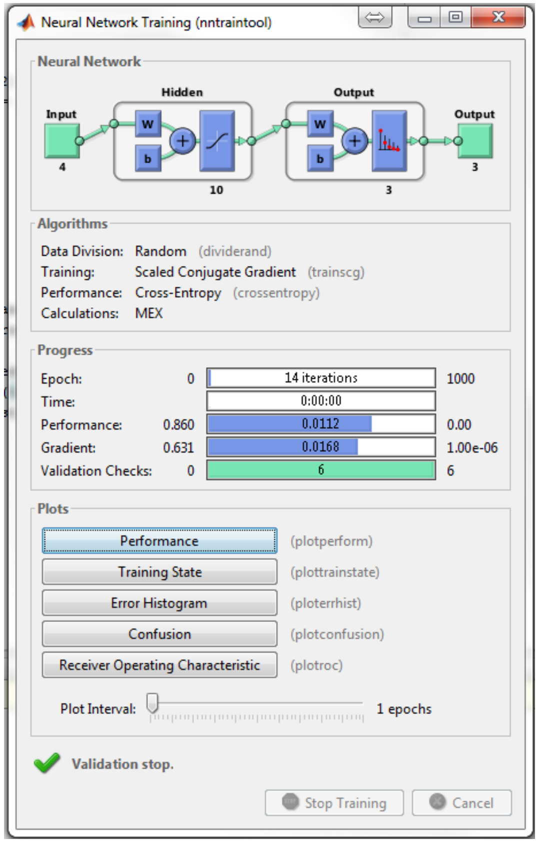
Neural Network Pattern Recognition Tool: GUI



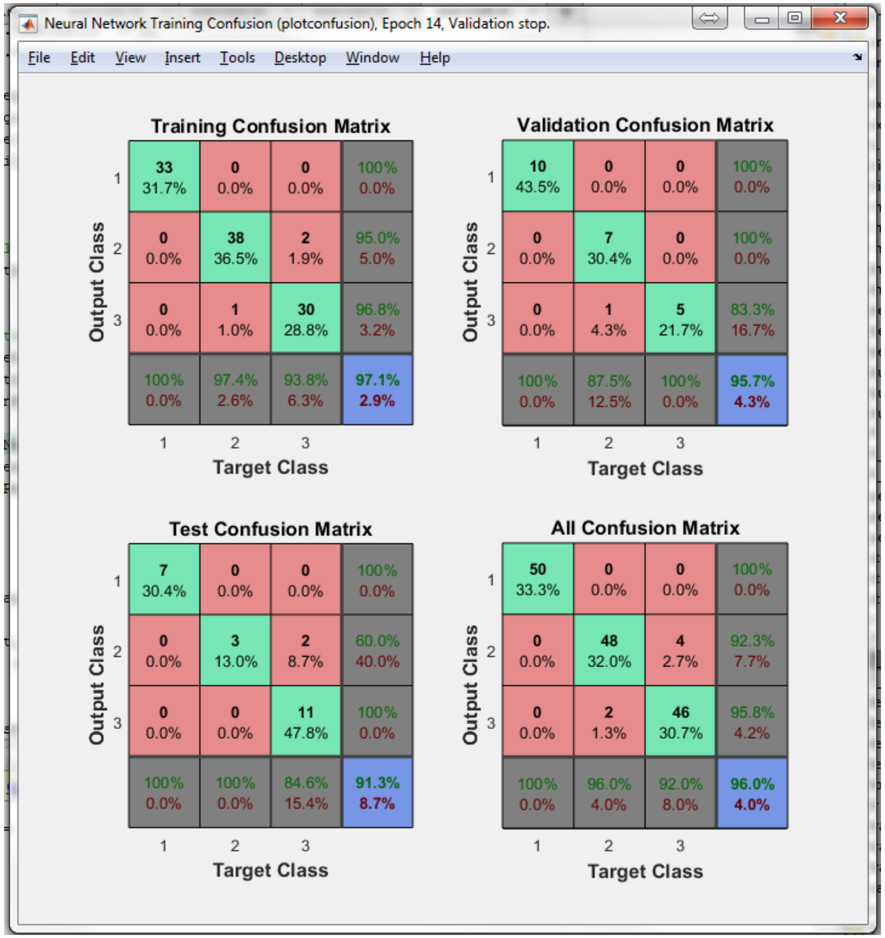
Neural Network Pattern Recognition Tool: GUI



Neural Network Pattern Recognition Tool: GUI



Neural Network Pattern Recognition Tool: GUI



Example 1

- Train a neural classifier to identify if glass is a window or not from glass chemistry using the GUI:
  - 9 features: Refractive index, Sodium (unit measurement: weight percent in corresponding oxide) , Magnesium, Aluminum, Silicon, Potassium, Calcium, Barium, Iron
  - 2 classes

Exercises

1. Train two neural classifiers using the GUI:
   - Identify if breast tumor is malignant or not
     - Nine features: Clump thickness, Uniformity of cell size, Uniformity of cell shape, Marginal Adhesion, Single epithelial cell size, Bare nuclei, Bland chomatin, Normal nucleoli
     - Two classes: non-malignant, malignant
   - Identify the species of iris flowers
     - Four physical characteristics of flowers are considered: sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)
     - Three classes (setosa, virginia, versicolor)
   - Experiment with different numbers of neurons
   - Load the data from the example datasets in Matlab

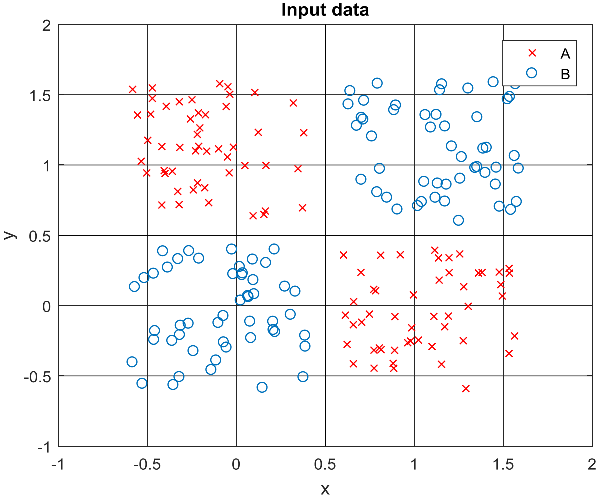
Neural Networks in real applications

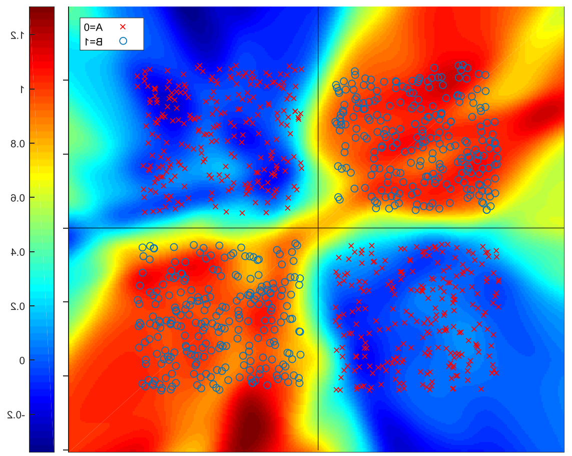
- The GUI has been used only for discussing basic concepts

- In real applications, it is better to use command-line functions

Example 2

- Two classes classification
  - train a neural classifier
  - evaluate the obtained results in a graphical mode
  - evaluate the obtained error





Note: download code from https://homes.di.unimi.it/munoz/teaching.html

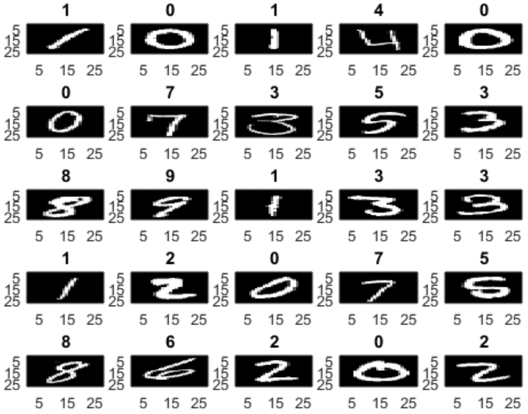
Exercises

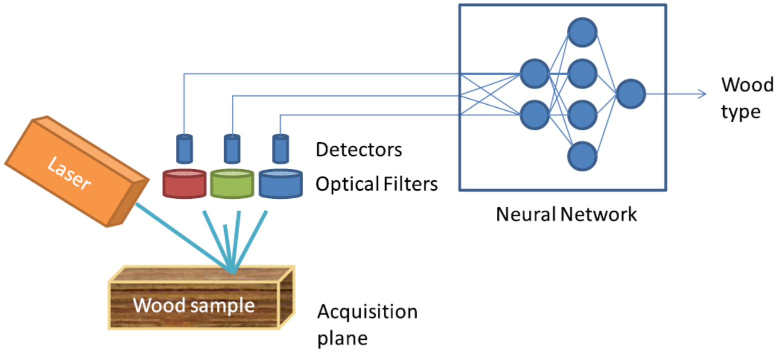
2. Evaluate the impact of changes in the dataset and neural network from example 2
   – change the number of the input points
   – reduce the separation between classes (parameter q)
   – change the parameters of the neural network

Note: download code from https://homes.di.unimi.it/munoz/teaching.html

Classification with more than two classes

- In many cases the number of classes is greater than two
  - Digits
  - Flowers
  - Wood types…





Classification with more than two classes: method I

- One output neuron
- Assign to each class a different integer identifier
- Training:

```
tA = zeros(1, size(A,2));
tB = ones(1, size(A,2));
tC = ones(1, size(A,2))*2;
…
T=[tA, tB, tC, …];
```

- Classification:

```
testResult = net(P_test);
testResult = round(testResult);
testResult(testResult<0)=0;
testResult(testResult>numClasses-1)=numClasses-1;
```

Classification with more than two classes: method II

- N output neurons, one per each class
- Assign to each class a different target vector, with ones for samples belonging to the class and zeros for the rest
- Training:

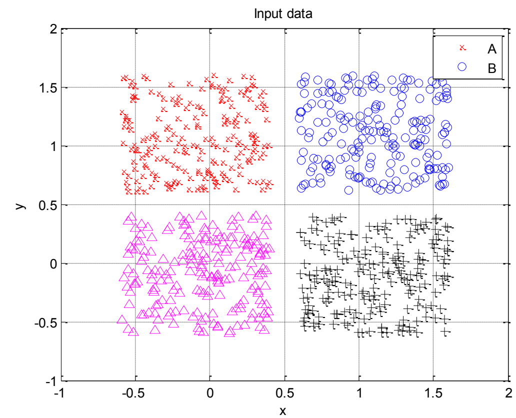
```
tA = zeros(1, N*4);
tA(1:N)=1;
tB = zeros(1, N*4);
tB(N+1:2*N) = 1;
tC = zeros(1, N*4);
tC(2*N+1:3*N)=1;
…
```

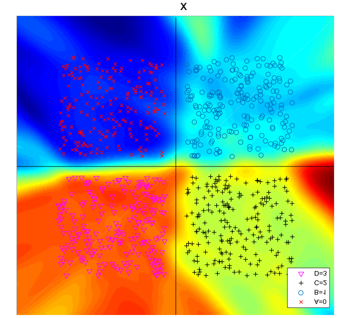
- Classification:

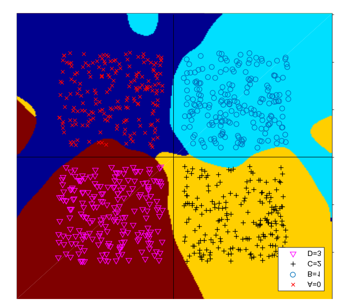
```
trainResult = net(P_test);
[~,testResult] = max(testResult);
```

Exercises

3. Four classes classification with one output neuron
   - Generate four classes as those depicted in the figure
   - Train a neural classifier using method I (one output neuron)
   - Optimize the parameters (in terms of test classification error)
   - Plot and analyze results graphically
4. Four classes classification with four output neurons
   - Train a neural classifier using method II (four output neurons)
   - Optimize the parameters (in terms of test classification error)
   - Plot and analyze results graphically







Note: download code of example2 from https://homes.di.unimi.it/munoz/teaching.html

Accuracy evaluation

- Error rate:

  Error rate % = incorrect predictions / total predictions * 100
- Classification accuracy:

  classification accuracy % = 100 – error rate %
- Good performance measure, but may have problems for particular applications
- It hides the detail needed to better understand the performance of a classification model:
  - When data is not balanced. Example: achieving 90% of accuracy, for a dataset where 90 samples out of 100 belong to one class. It could be that we are predicting that all samples belong to the dominant class.
  - When data has more than 2 classes. We don’t know if all classes are being predicted equally well or whether one or two classes are being neglected by the model.

Confusion matrix

- A summary of prediction results on classification problems
- Correct and incorrect predictions are counted and displayed for each class
- It shows the way in which a classification model (neural network) makes mistakes with its predictions
- It overcomes the limitation of using classification accuracy alone

Two-class confusion matrix

- Generally, in a two-class problem, we tray to discriminate between special samples and normal observations
  - E.g. disease or no disease
- Two classes
  - *True positives (TP)* - the number of elements correctly classified as positive by the test;
  - *True negatives (TN)* - the number of elements correctly classified as negative by the test;
  - *False positive (FP)* - also known as type I error, is the number of elements classified as positive by the test, but they are not;
  - *False positive (FN)* - also known as type II error, is the number of elements classified as negative by the test, but they are not.
- Matlab code

```
cm = confusionmat(actualT, predictedT); % matrix
plotconfusion(actualT, predictedT); % visual plot
```

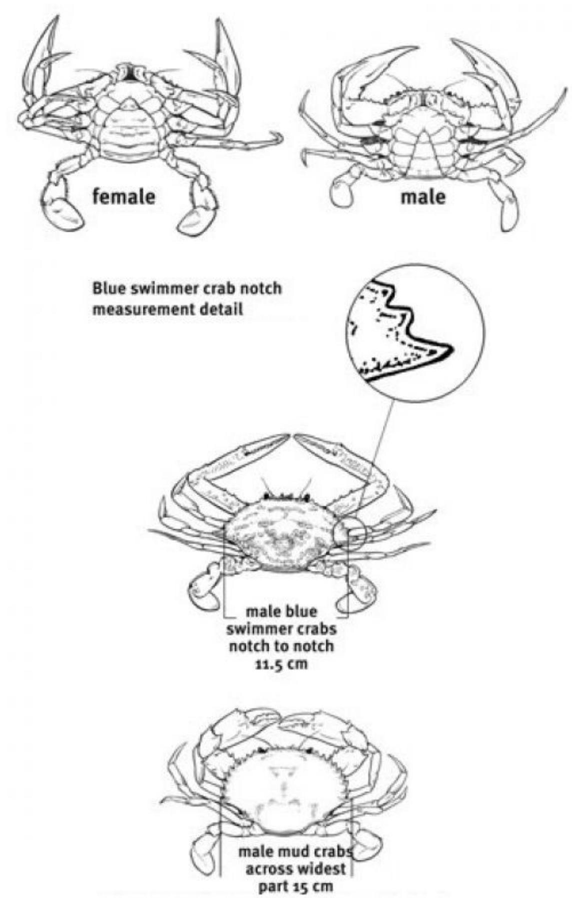
		prediction outcome		
		p	***n***	**total**
actual value	***p'***	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
total		P	N	

Exercises

5. Two-class classification with confusion matrix
   - Train a neural classifier to identify the gender of crabs from physical dimensions of the crab:
     - Six physical characterstics of a crab are considered: species, frontallip, rearwidth, length, width and depth.
     - 2 classes (male, female)
   - Load the dataset using the commands:

```
[x,t] = crab_dataset;
P = x;
T = t(1,:);
```

   - Divide in training and test sets
   - Try to optimize the parameters to minimize test error rate
   - Analyze the results using a confusion matrix



N-class confusion matrix

- Similar to two-class confusion matrix
- Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class
- Matlab code

  cm = confusionmat(actualT,predictedT); % matrix

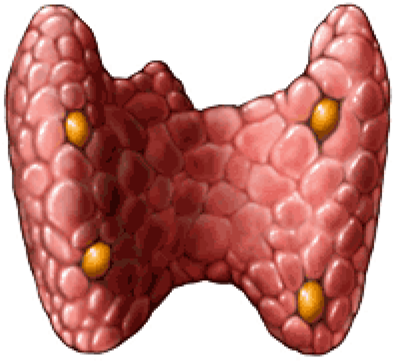
  Plotconfusion(actualT,predictedT); % visual plot

	box	clap	wave	jog	Run	Walk
Box	100	0	0	0	0	0
Clap	0	94	6	0	0	0
Wave	0	1	99	0	0	0
Jog	0	0	0	91	7	2
Run	0	0	0	10	89	1
Walk	0	0	0	0	6	94

Exercises

6. N-class classification with confusion matrix
   – Train a neural classifier to detect thyroid malfunctioning
     • 21 features describing patient attributes
     • Three classes corresponding to: normal, hyperthyroidism, hypothyroidism
   – Load the dataset with the command

     [x,t] = thyroid_dataset;
   – Divide in training and test sets
   – Try to optimize the parameters to minimize test error rate
   – Analyze the results using a confusion matrix

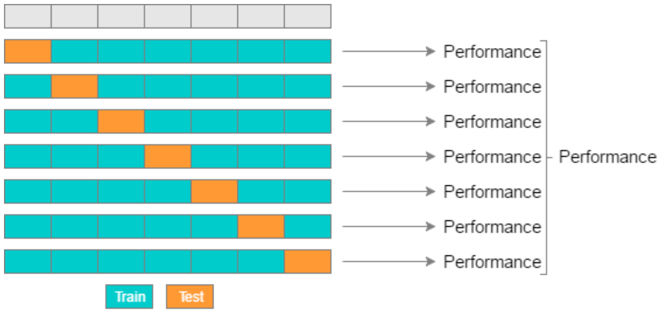


Cross Validation

- Cross-validation is a model evaluation method for assessing how the results of a statistical analysis will generalize:
  - **Holdout method**: simplest method. Data separated into training and test. Disadvantage: evaluation dependent on the partition.
  - **K-fold cross-validation method**: divide data into k folds and repeat holdout k times. Each time a fold is used for training and the rest for test. Results variance is reduced with a larger k. Disadvantage: computational time.
  - **Leave one out method**: extreme k-fold. Each fold contains just one sample. Disadvantage: computational time.

k-Fold cross validation

- Algorithm
  - In *k*-fold cross-validation, the original sample is randomly partitioned into *k* equal size subsamples.
  - Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining *k* − 1 subsamples are used as training data.
  - The cross-validation process is then repeated *k* times (the *folds*), with each of the *k* subsamples used exactly once as the validation data.
  - The *k* results from the folds then can be averaged (or otherwise combined) to produce a single estimation.
- All observations are used for both training and validation, and each observation is used for validation exactly once.
- 10-fold cross-validation is the most commonly used.

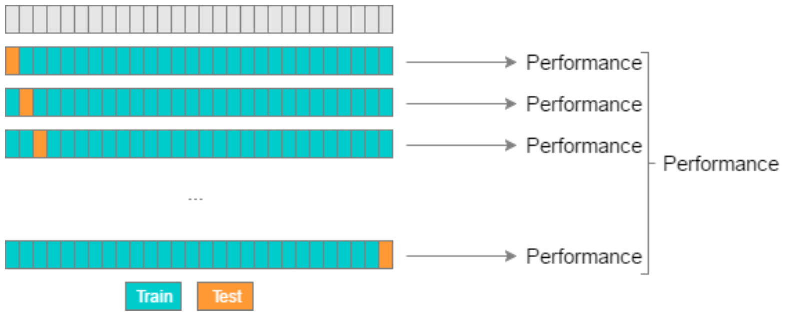


k-Fold cross validation: Matlab script

```
indices = crossvalind('Kfold', T ,k);

kFoldTrainResults = [];
kFoldTestResults = [];
kFoldTotalResults = [];

kFoldTrainTs = [];
kFoldTestTs = [];
kFoldTotalTs = [];
for i =1:k
   P_train=P(:,indices ~=i);
   T_train=T(:,indices ~=i);
   P_test=P(:,indices ==i);
   T_test=T(:,indices ==i);
   % train the network and evaluate performance
…
   % Aggregate results
   kFoldTrainResults=[kFoldTrainResults, trainResult];
   kFoldTestResults=[kFoldTestResults, testResult];
   kFoldTotalResults=[kFoldTotalResults, trainResult, testResult];
   kFoldTrainTs=[kFoldTrainTs, T_train];
   kFoldTestTs=[kFoldTestTs, T_test];
   kFoldTotalTs=[kFoldTotalTs, T_train, T_test];
end
% compute summary results
kFoldTrainErrors=kFoldTrainTs~=kFoldTrainResults;
kFoldTestErrors=kFoldTestTs~=kFoldTestResults;
kFoldTotalErrors=kFoldTotalTs~=kFoldTotalResults;


kFoldTrainErrorRate=sum(kFoldTrainErrors)/size(kFoldTrainErrors,2)*100;
kFoldTestErrorRate=sum(kFoldTestErrors)/size(kFoldTestErrors,2)*100;
kFoldTotalErrorRate=sum(kFoldTotalErrors)/size(kFoldTotalErrors,2)*100;

cm=confusionmat(kFoldTestTs,kFoldTestResults);
```

Leave-one-out cross validation

- Algorithm
  - In leave-one-out cross validation, the original sample is partitioned into as many subsets as samples contained in the dataset (n)
  - One sample is used for test and the remaining are used as training data
  - The cross-validation process is then repeated *n* times (the number of samples in the dataset)
  - The results from the process are aggregated to produce a single estimation
- All observations are used for both training and validation, and each observation is used for validation exactly once



Leave-one-out cross validation: Matlab script

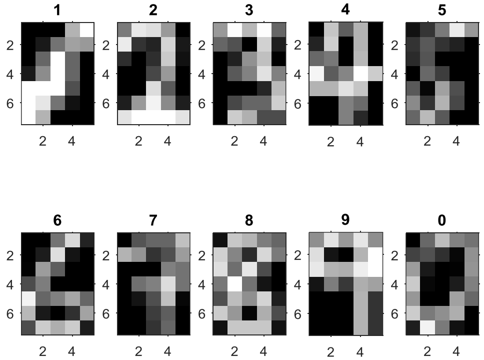
```
LOOTrainResults = [];
LOOTestResults = [];
LOOTotalResults = [];

LOOTrainTs = [];
LOOTestTs = [];
LOOTotalTs = [];
for i =1:size(T,2)
   P_train=P(:,[1:i-1 i+1:end]);
   T_train=T(:,[1:i-1 i+1:end]);
   P_test=P(:,i);
   T_test=T(:,i);
   % train the network and evaluate performance
...
   % Aggregate results
   LOOTrainResults=[LOOTrainResults, trainResult];
   LOOTestResults=[LOOTestResults, testResult];
   LOOTotalResults=[LOOTotalResults, trainResult,
testResult];
    LOOTrainTs=[LOOTrainTs, T_train];
   LOOTestTs=[LOOTestTs, T_test];
   LOOTotalTs=[LOOTotalTs, T_train, T_test];
end% compute summary results
LOOTrainErrors=LOOTrainTs~=LOOTrainResults;
LOOTestErrors=LOOTestTs~=LOOTestResults;
LOOTotalErrors=LOOTotalTs~=LOOTotalResults;


LOOTrainErrorRate=sum(LOOTrainErrors)/size(LOOTrainErrors
,2)*100;
LOOTestErrorRate=sum(LOOTestErrors)/size(LOOTestErrors,2)
*100;
LOOTotalErrorRate=sum(LOOTotalErrors)/size(LOOTotalErrors
,2)*100;


cm=confusionmat(LOOTestTs,LOOTestResults);
```

Exercises

7. Calculate the performance of the neural network developed for exercise 3 using:
   - Hold out
   - 5-fold cross validation
   - 10-fold cross validation
   - Leave-one-out cross validation

Exercises

8. Hand-written digit classification
   - Train a neural classifier to classify hand-written digits
     - 35 features (5x7 grayscale image)
     - 10 classes (0, …, 9)
   - Optimize parameters to minimize 10-fold cross validation error
   - Load images from digits directory and targets from digit_names.mat (download from https://homes.di.unimi.it/munoz/teaching.html)
   - Useful code to create P and T




```
load('digits_names.mat');
files=dir('digits/*.bmp');
for i=1:numel(files)
   im=imread(['digits/' files(i).name]);
   P(:,i)=double(reshape(im,[1,35]));
   for j=1:numel(names)
      if strcmp(files(i).name,names{j})
        T(:,i)=targetsByName(:,j);
        break;
      end
   end
end
```