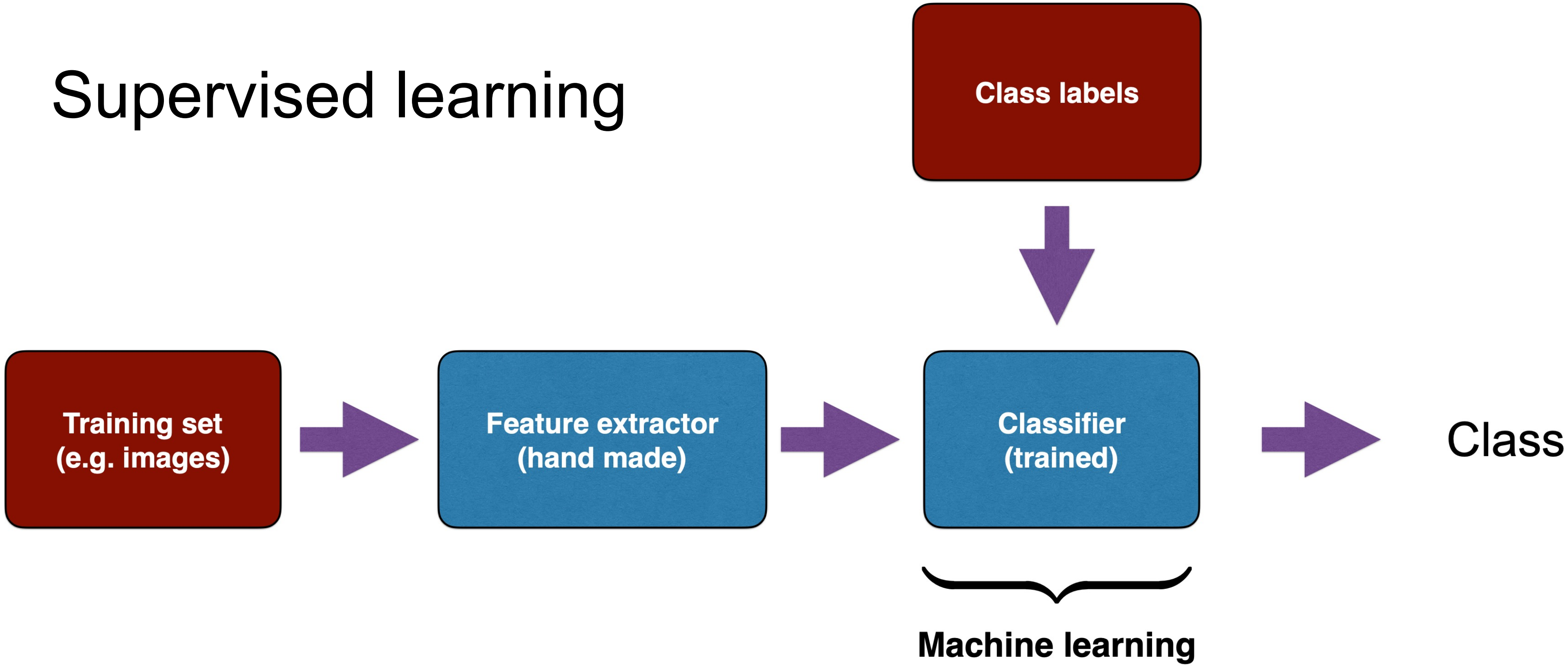
Introduction to Machine Learning

Idar Dyrdal

Machine learning (Pattern recognition)

- Recognition of individuals (instance recognition)
- Discrimination between classes (pattern recognition, classification)

Supervised learning



Pattern recognition in practice

Working applications of Image Pattern recognition:

- Reading license plates, postal codes, bar codes
- Character recognition
- Automatic diagnosis of medical samples
- Fingerprint recognition
- Face detection and recognition
- …

Credit: L. Lazebnik

Classification system

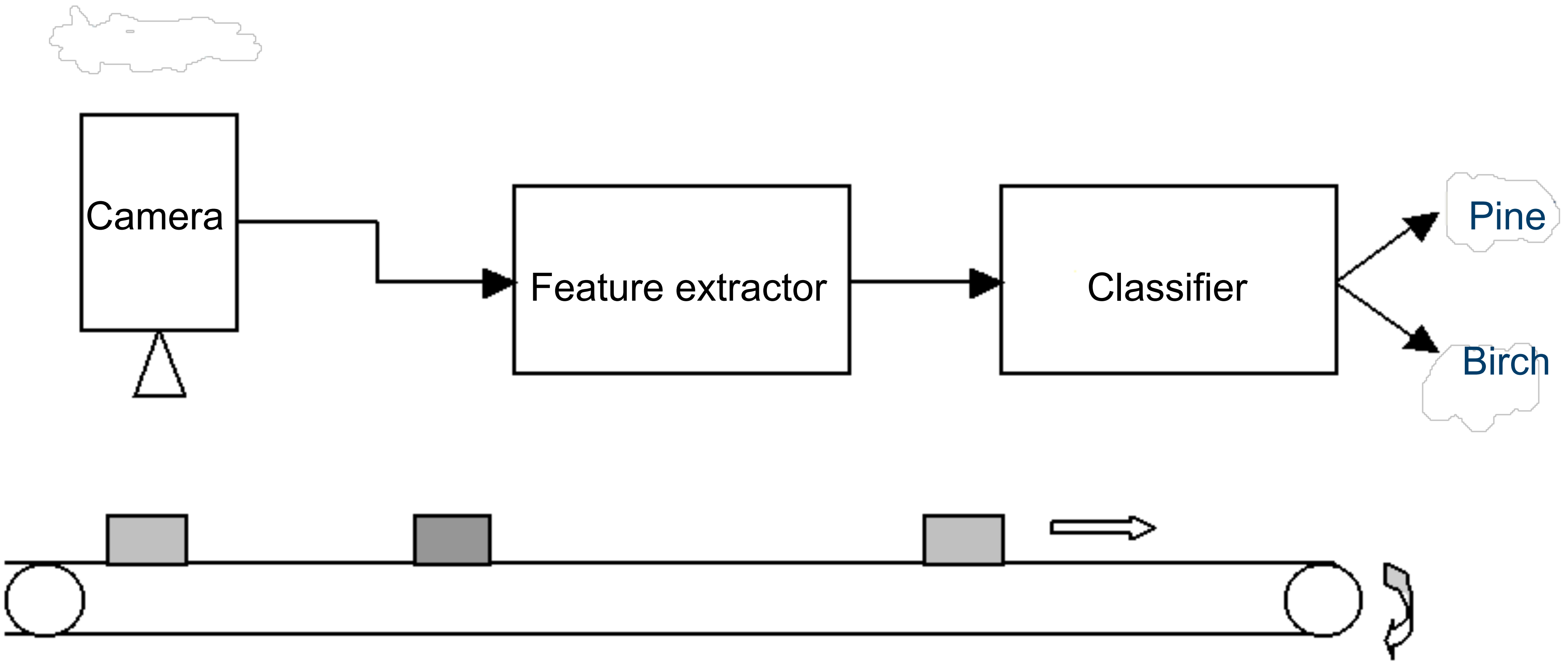
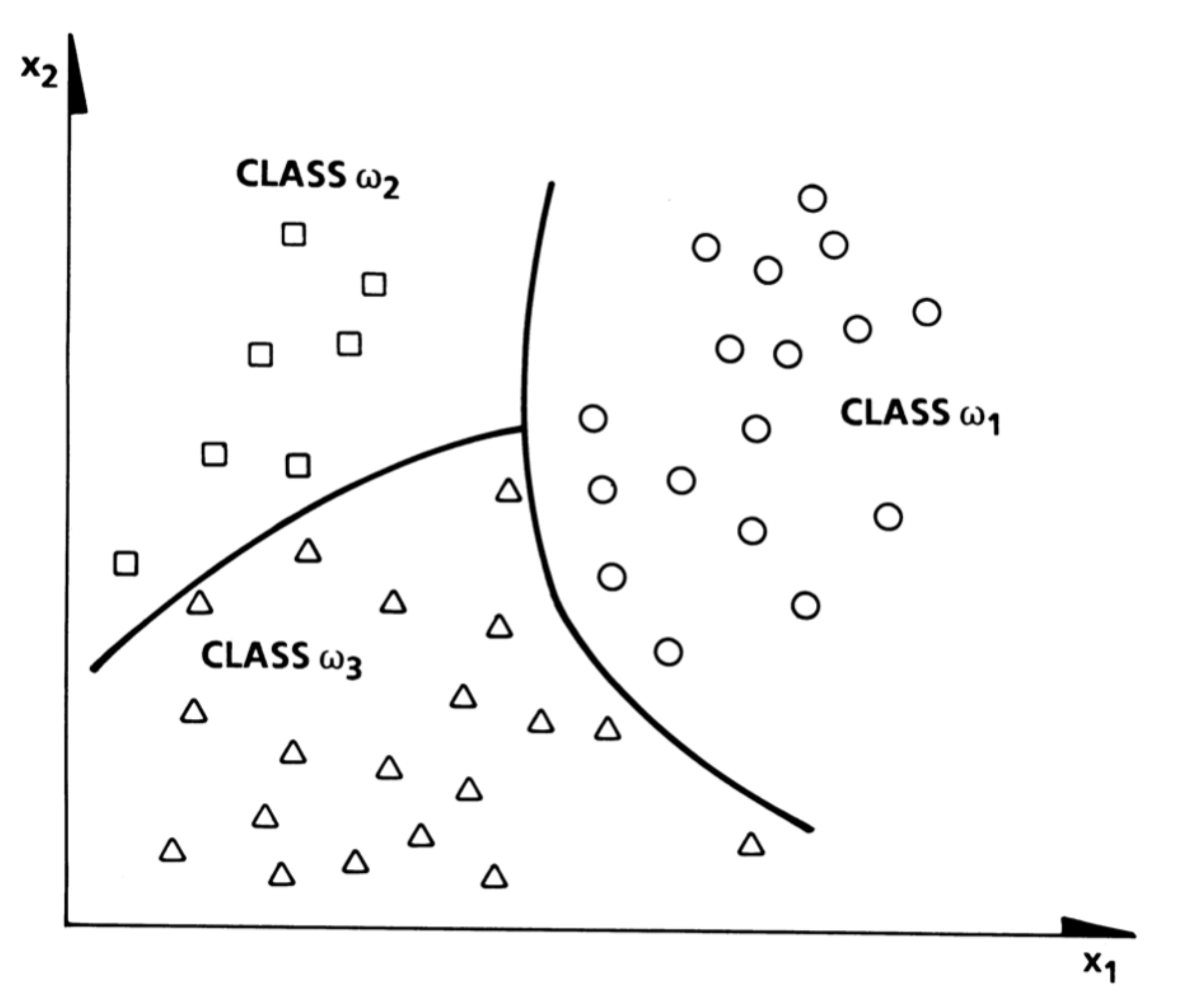


Image features for object recognition

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$

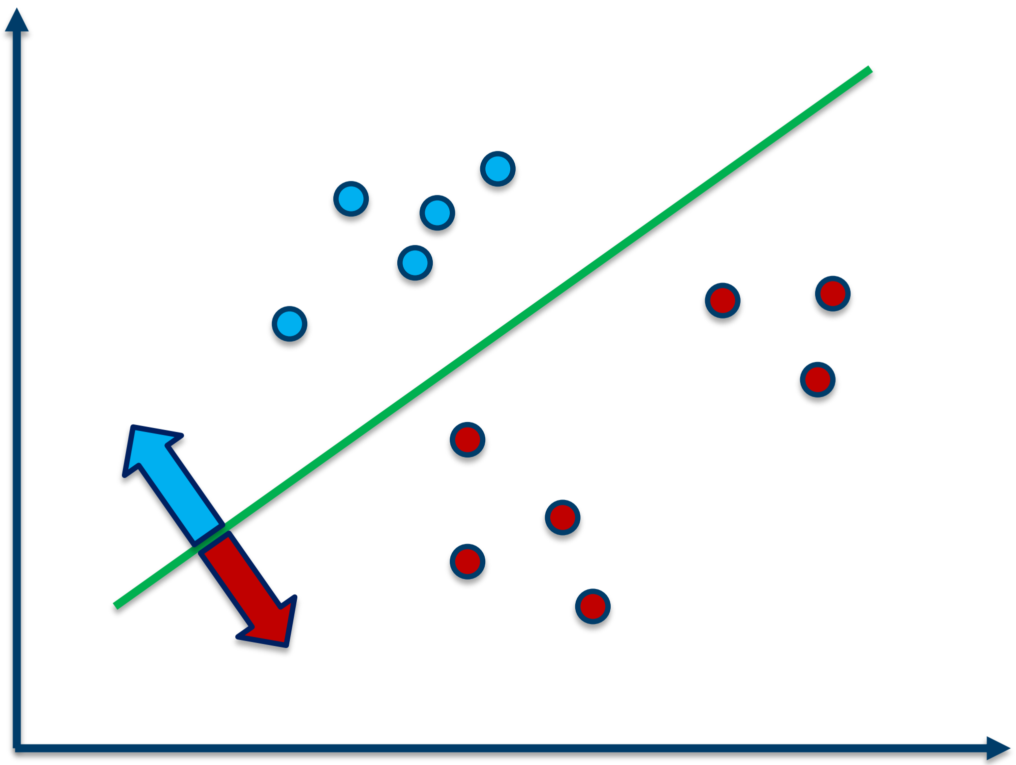
Feature vector and feature space

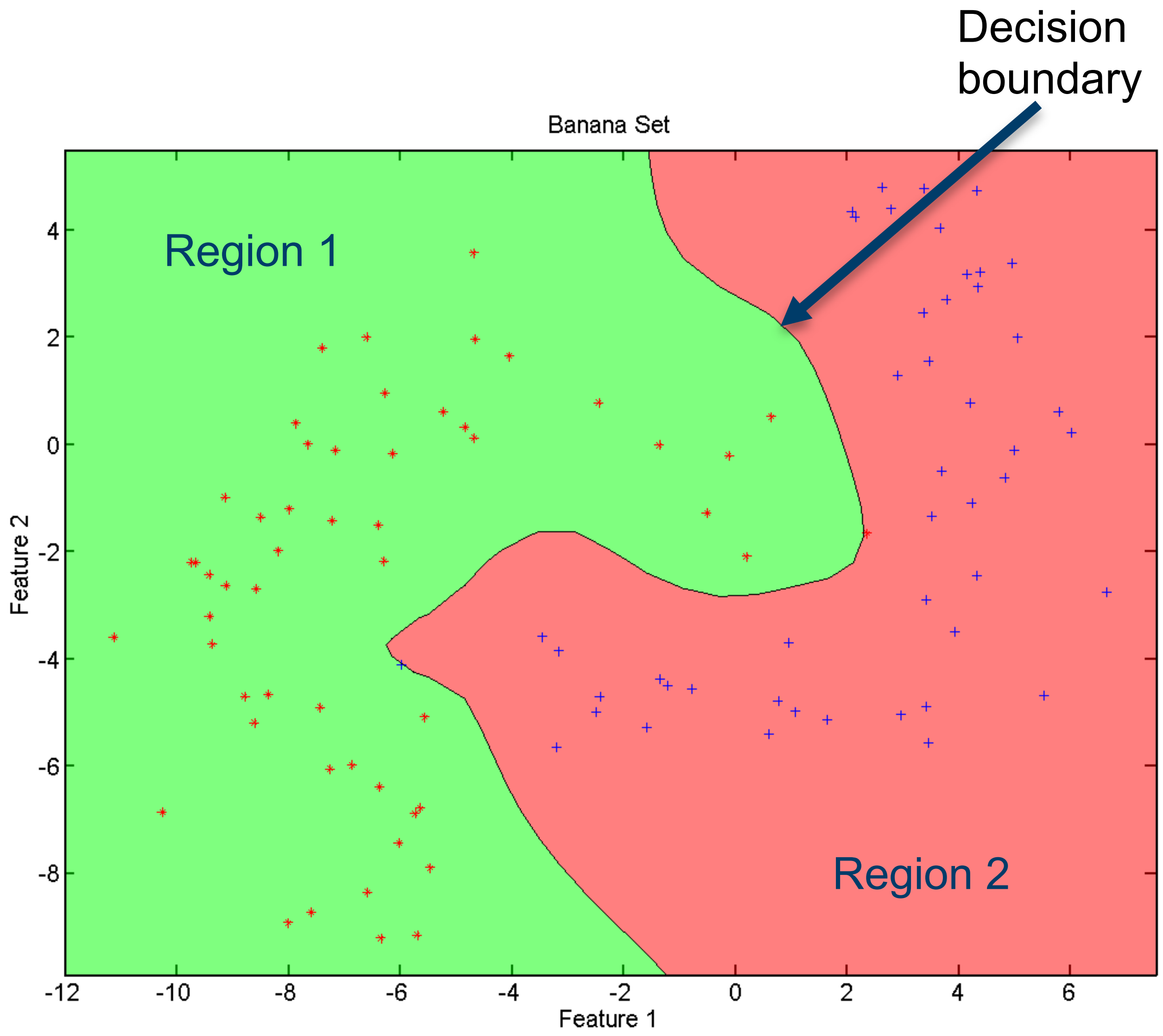
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$



Training of classifiers

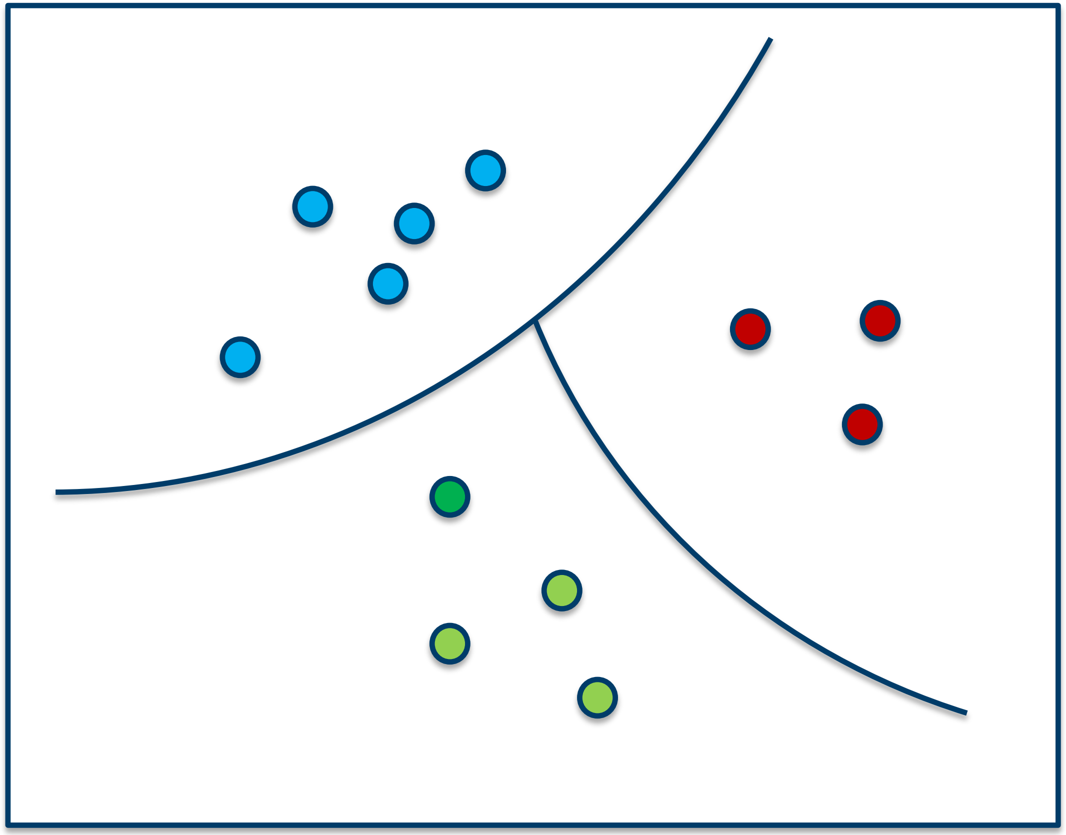
Learn a function to predict the class from the given features



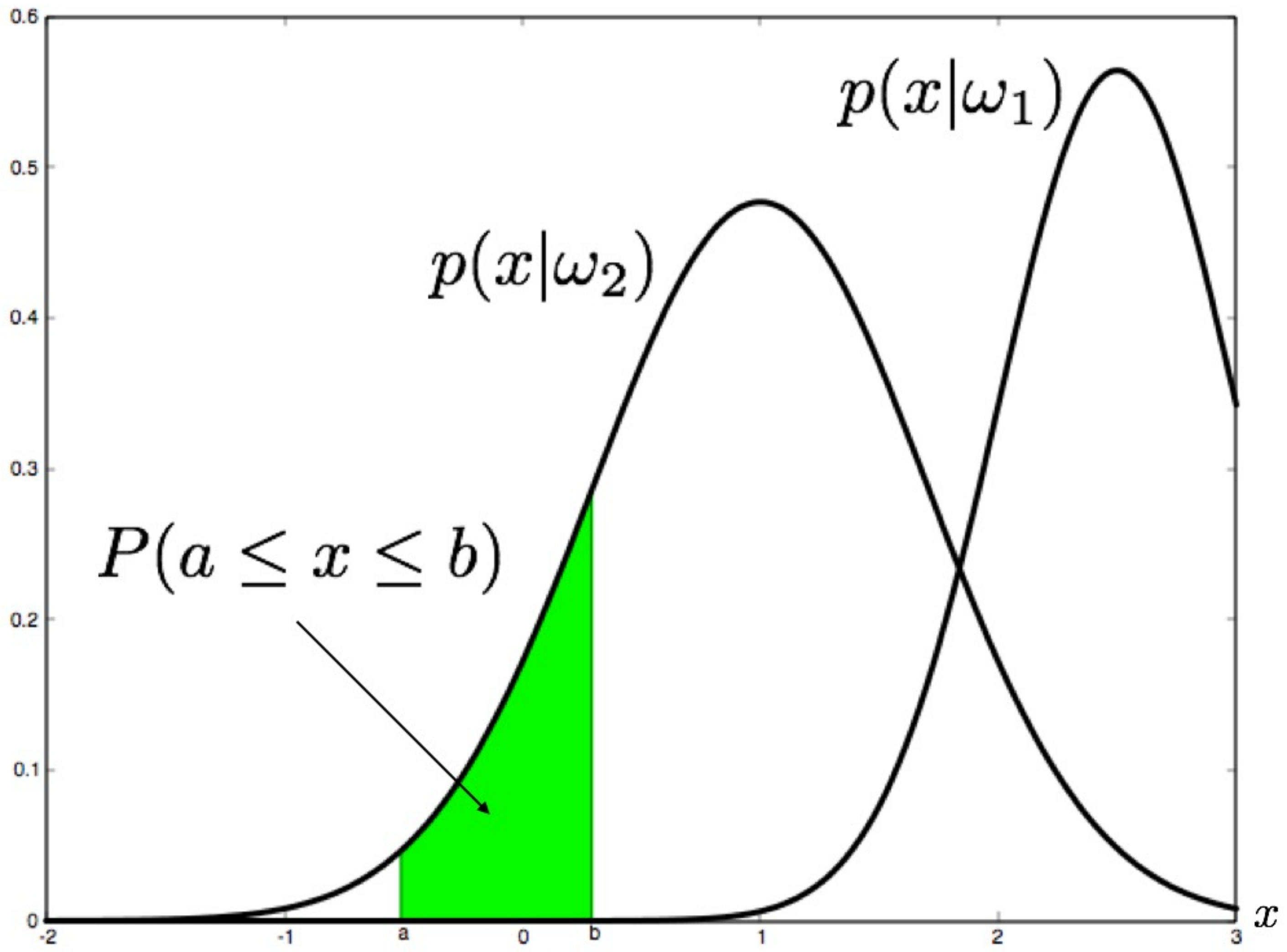


Classifiers and training methods

- Bayes (parametric) classifier
- Nearest-neighbors and K-nearest-neighbors
- Parzen windows
- Linear and higher order discriminant functions
- Neural nets
- Support Vector Machines (SVM)
- Decision trees
- Random forest
- …



Class conditional probability density functions



Bayesian decision theory

Overview

Class conditional densities:

$$p(\boldsymbol{x}|\omega_i), \text{ for each class } \omega_1, \omega_2, \ldots, \omega_c$$

Prior probabilities:

$$P(\omega_1), P(\omega_2), \ldots, P(\omega_c)$$

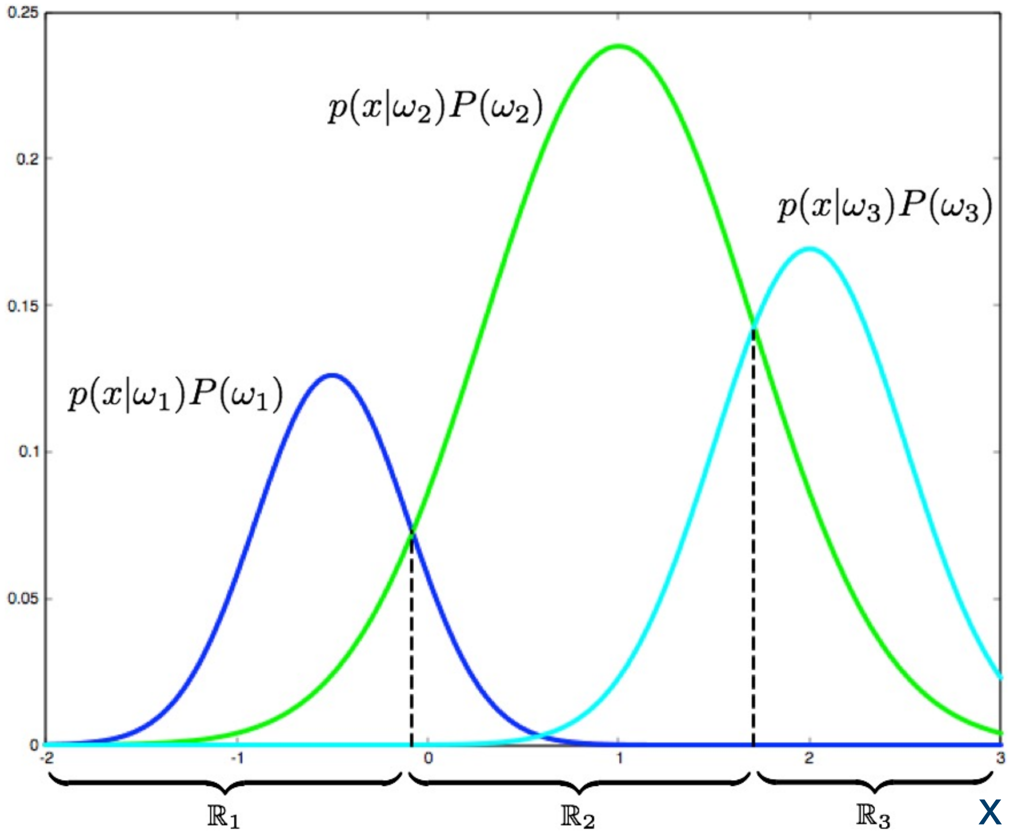
Posterior probabilities given by Bayes rule:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(\boldsymbol{x}|\omega_j)P(\omega_j)}, \; i = 1, \ldots, c$$

(a function of the measured feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^t$).

Minimum error rate classification:

Assign the unknown object to the class with maximum posterior probability!



Density estimation

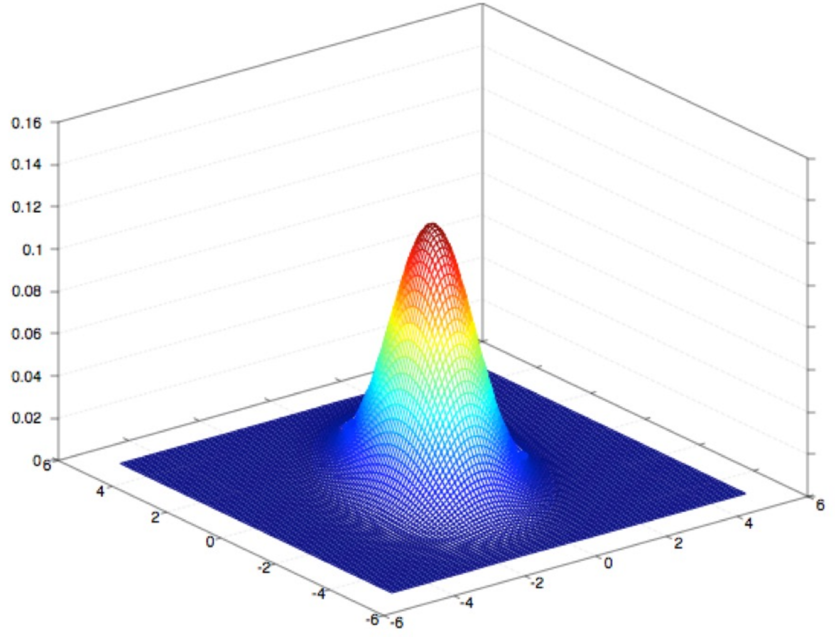
Parametric methods:

- Assume a given shape of the density function
- Use the training set to estimate the unknown parameters.

Non-parametric (distribution free) methods:

- Point estimation of the density using the training set directly
- Parzen windows
- Nearest neighbor estimation (leads directly to the nearest-neighbor and k-nearest-neighbor classifiers).

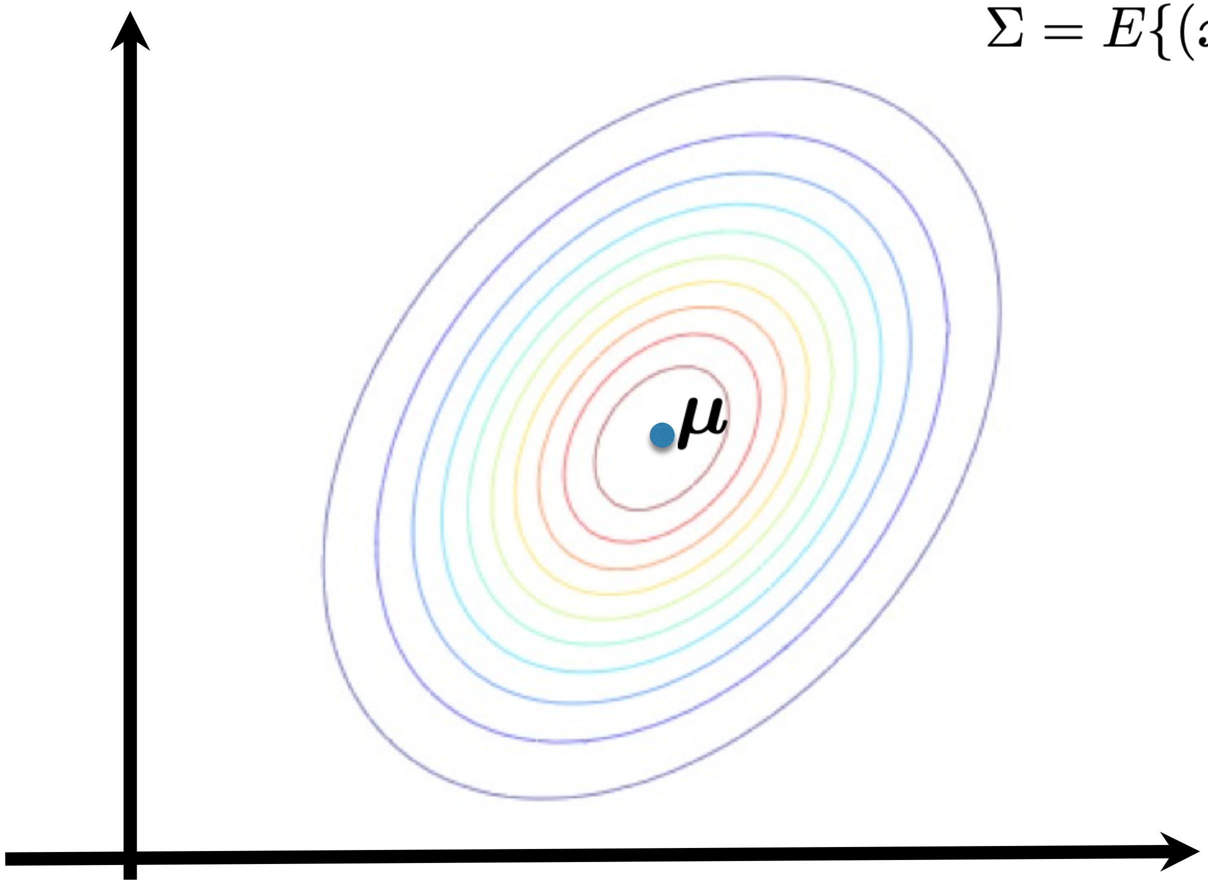
Example – Gaussian distribution:



$$p(\boldsymbol{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right]$$

Parameters: $\boldsymbol{\mu}_i$ and Σ_i

Parameter estimation



$$\Sigma = E\{(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^t\} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1d} \\ \vdots & \vdots & & \vdots \\ \sigma_{d1} & \sigma_{d2} & \dots & \sigma_{dd} \end{bmatrix}$$

Parameter estimates:

$$\hat{\boldsymbol{\mu}} = \boldsymbol{m} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{x}_k$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n} (\boldsymbol{x}_k - \boldsymbol{m})(\boldsymbol{x}_k - \boldsymbol{m})^t$$

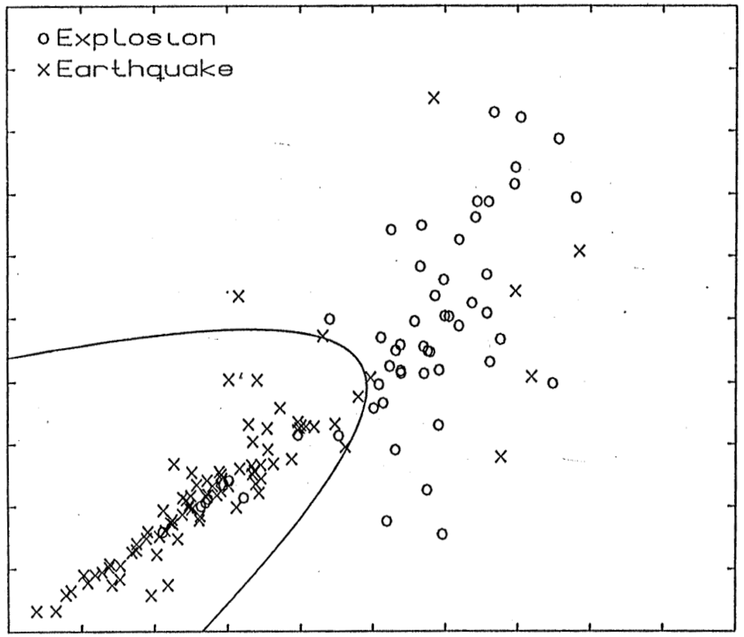
Discriminant functions

Estimate of the density in a given point:

$$\hat{p}(\boldsymbol{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\hat{\Sigma}_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\hat{\boldsymbol{\mu}}_i)^t\hat{\Sigma}_i^{-1}(\boldsymbol{x}-\hat{\boldsymbol{\mu}}_i)\right]$$

From Bayes rule:

$$\hat{P}(\omega_i|\boldsymbol{x}) = \frac{\hat{p}(\boldsymbol{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c}\hat{p}(\boldsymbol{x}|\omega_j)P(\omega_j)}$$



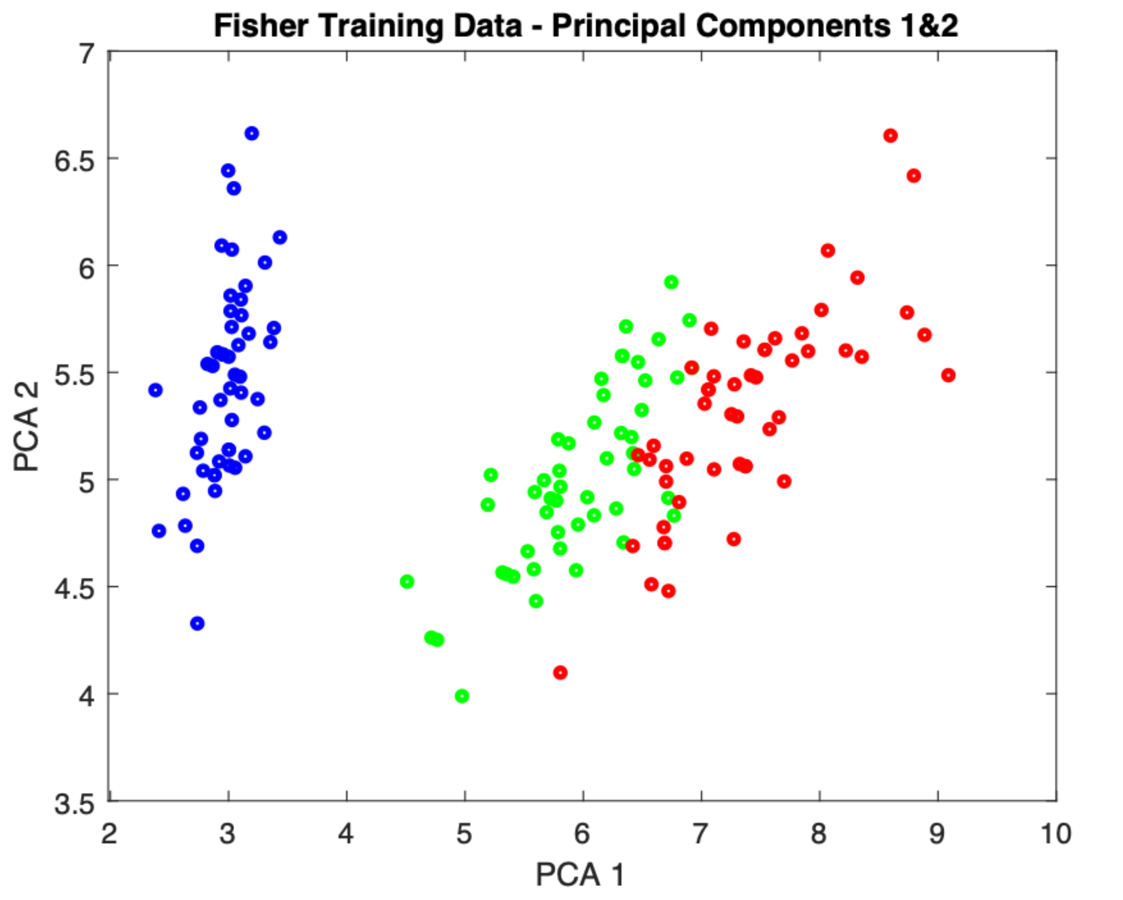
Examples of discriminant functions:

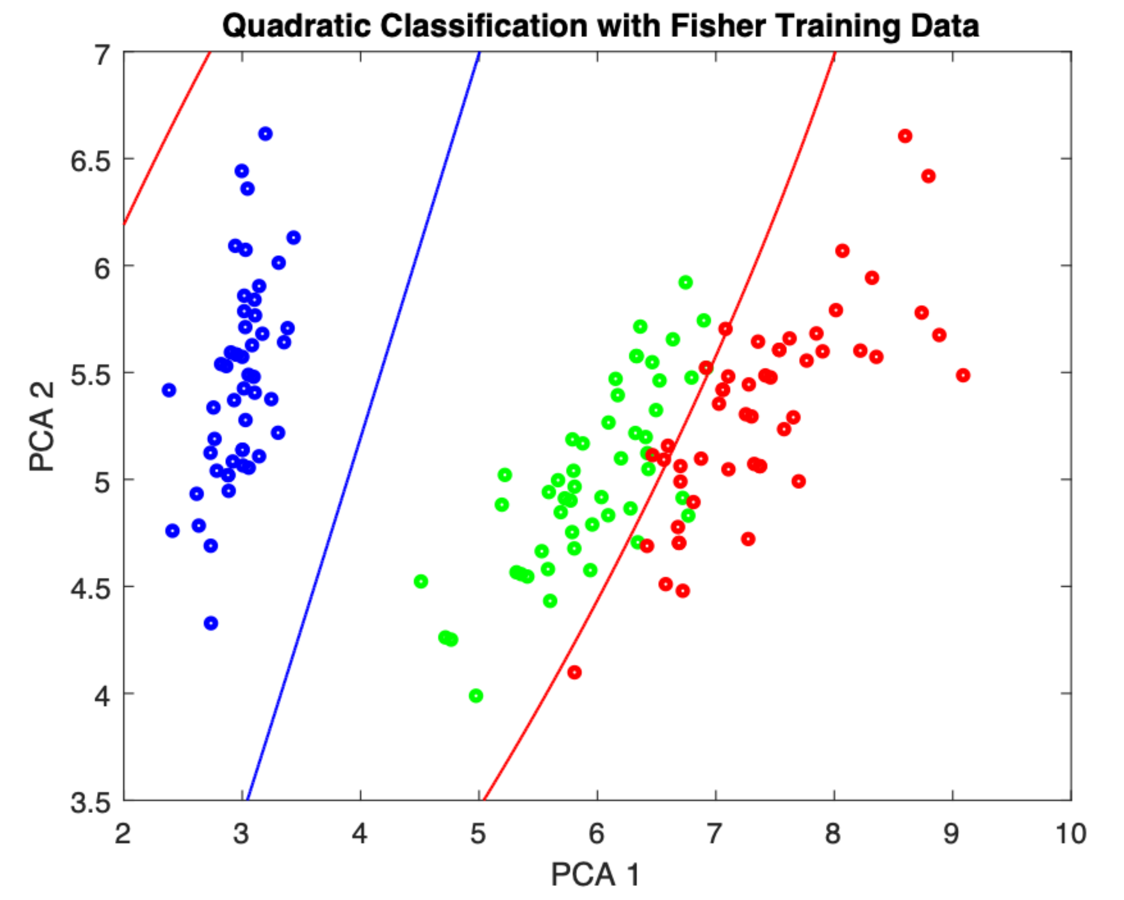
$$g_i(\boldsymbol{x}) = \ln \hat{P}(\omega_i|\boldsymbol{x}) \quad \text{or} \quad g_i(\boldsymbol{x}) = \ln \hat{p}(\boldsymbol{x}|\omega_i) + \ln P(\omega_i)$$

Decision rule:

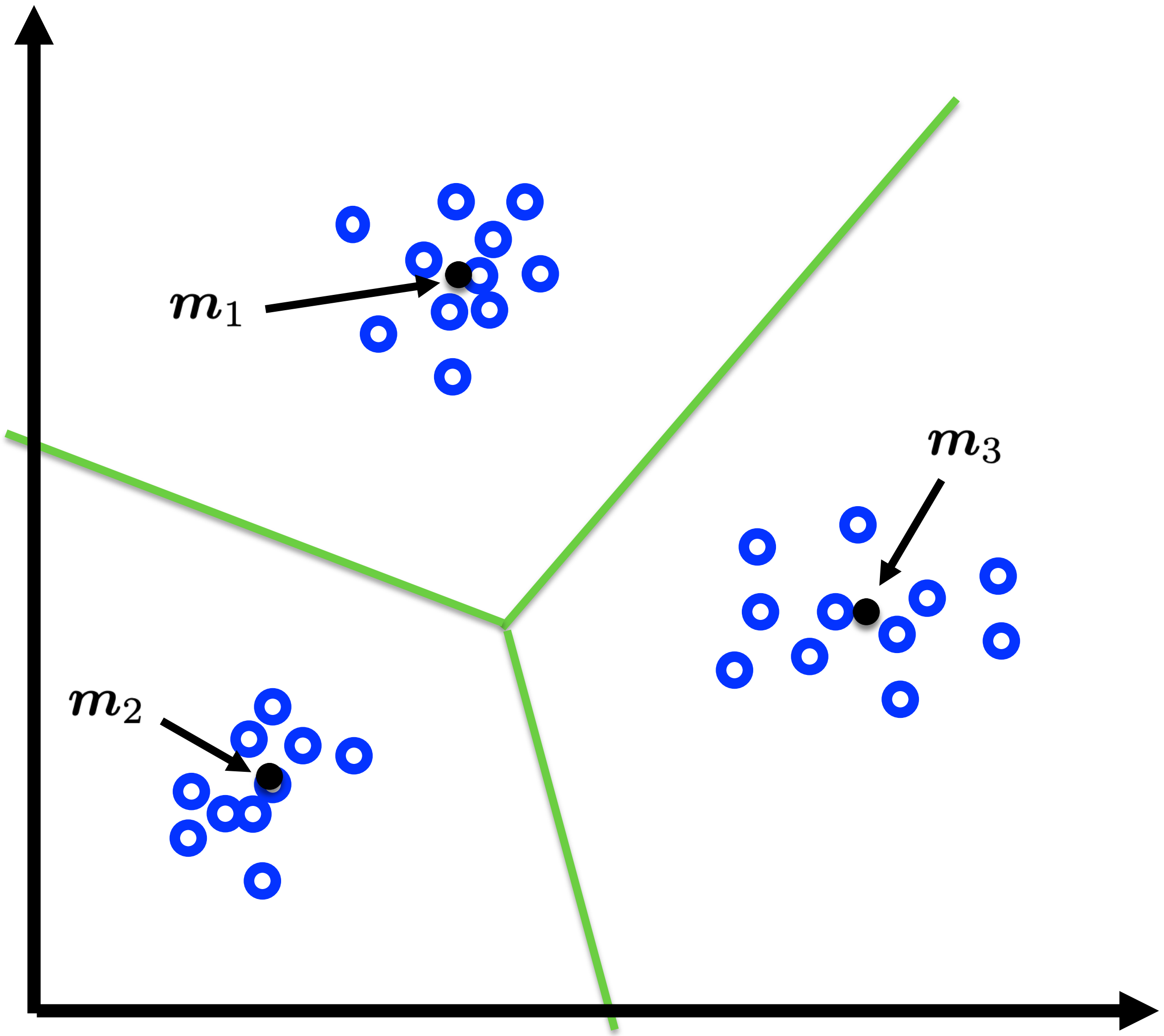
Choose the class with maximum discriminant function value.

Quadratic classifier - example





Linear classifier



Example:

Uncorrelated features and common covariance matrices

$$\Downarrow$$

Linear decision boundaries

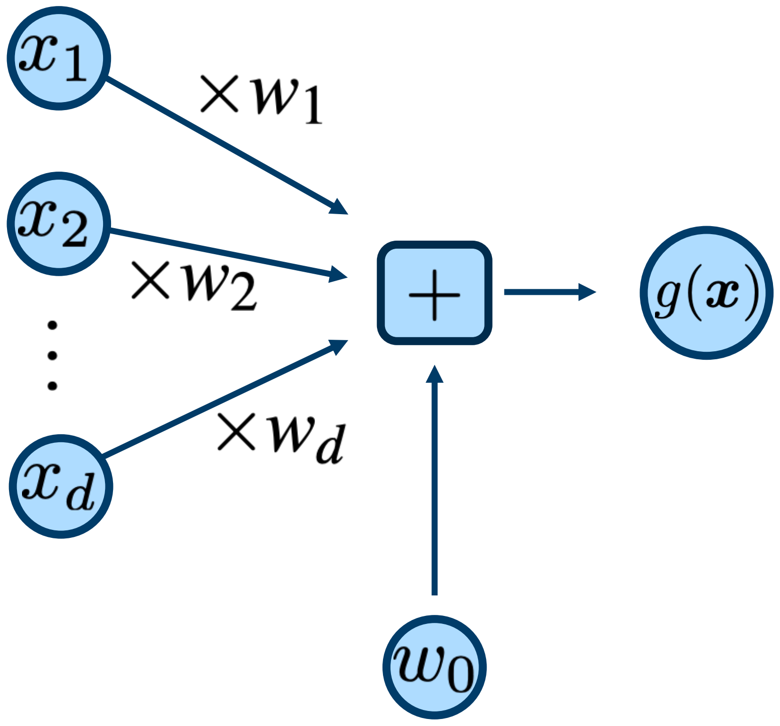
Linear classifier (contd.)

Discriminant function:

$$g(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + w_0 = w_0 + \sum_{i=1}^{d} w_i x_i$$

Can be rewritten as:

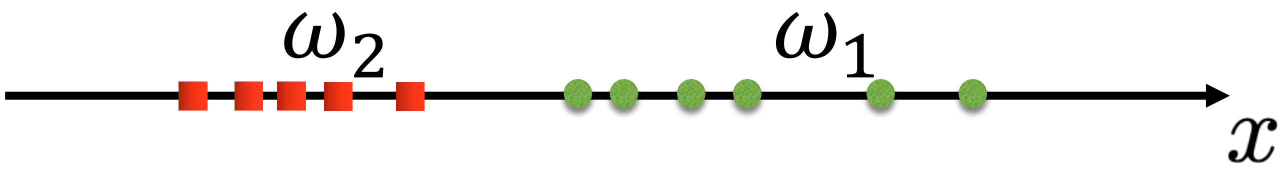
$$g(\boldsymbol{x}) = [w_0, w_1, \ldots, w_d] \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \boldsymbol{a}^t\boldsymbol{y}$$

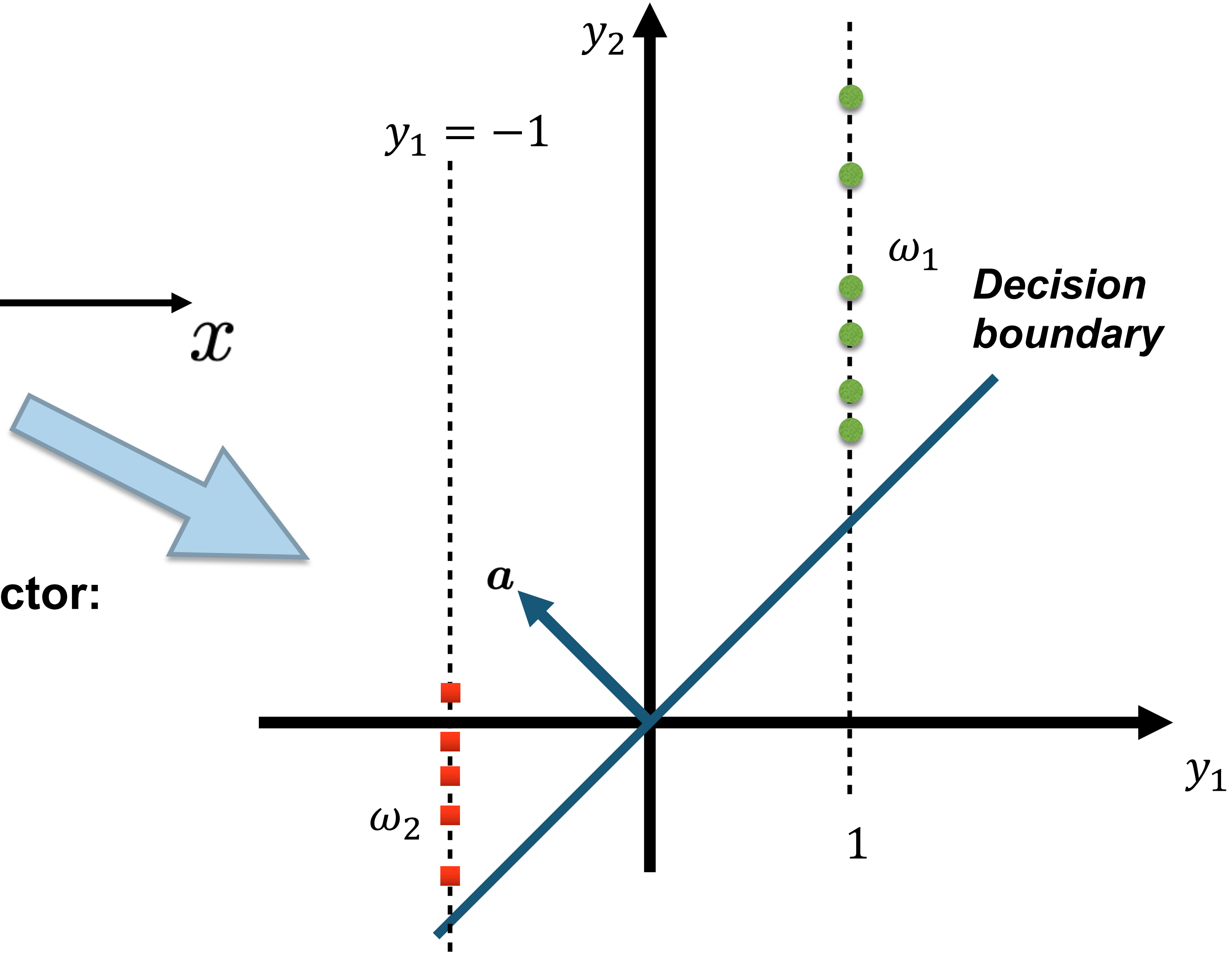


Decision rule (two-class case):

Decide ω_1 if $\boldsymbol{a}^t\boldsymbol{y} > 0$ and ω_2 if $\boldsymbol{a}^t\boldsymbol{y} \leq 0$

Augmented feature space





Normalized samples in augmented feature space.

Augmented weight vector and feature vector:

$$\boldsymbol{a} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Gradient descent

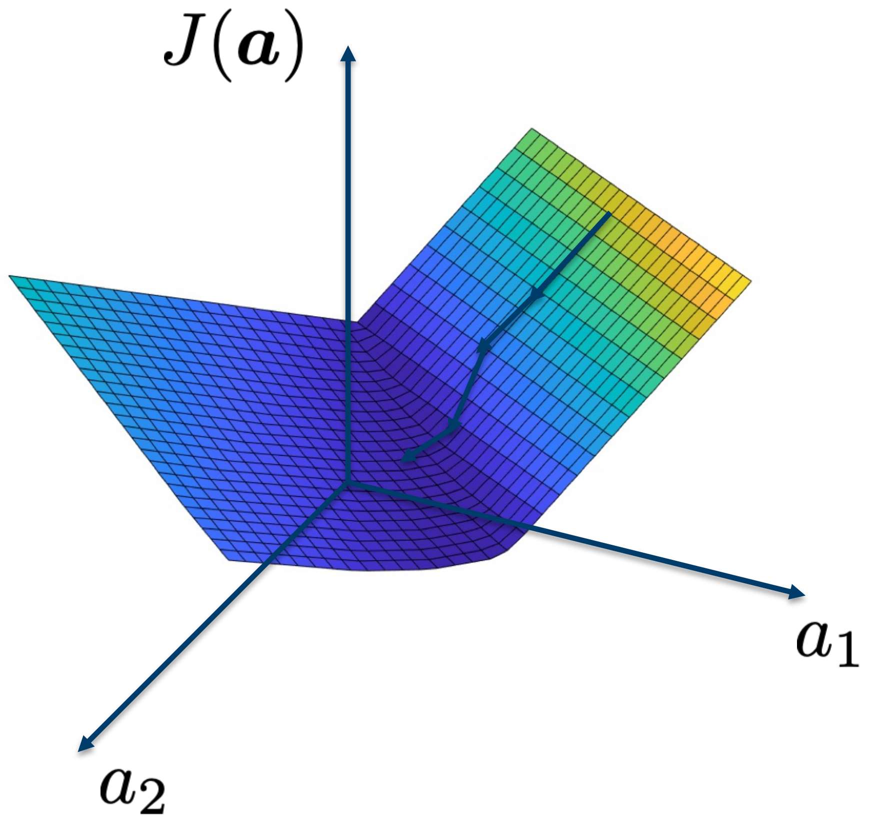
Find the minimum of a criterion function:

$$J(\boldsymbol{a}) \geq 0$$

Basic algorithm:

$$\boldsymbol{a}_1 = \text{arbitrary}$$

$$\boldsymbol{a}_{k+1} = \boldsymbol{a}_k - \rho_k \nabla J(\boldsymbol{a}_k),\ k = 1, 2, \ldots\ldots$$



Optimal step length (increment):

$$\rho_k = \frac{\|\nabla J\|^2}{\nabla J^t D \nabla J} \quad \text{where} \quad D_{ij} = \frac{\partial^2 J(\boldsymbol{a}_k)}{\partial a_j \partial a_k} \quad \text{are the components of matrix } D$$

Example

Perceptron criterion function:

$$J_p(\boldsymbol{a}) = -\sum_{\boldsymbol{y}\in\mathcal{Y}} \boldsymbol{a}^t\boldsymbol{y} \quad \text{where } \mathcal{Y} = \{\boldsymbol{y} : \boldsymbol{a}^t\boldsymbol{y} \leq 0\}$$

The goal is to satisfy a set of inequalities:

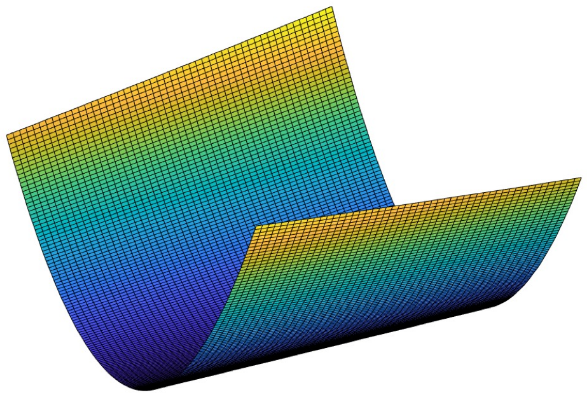
$$\boldsymbol{a}^t\boldsymbol{y}_i > 0 \;\forall\, \boldsymbol{y}_i \in \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\}$$

Linear least-squares optimization

Solve a set of linear equations

$$\boldsymbol{a}^t \boldsymbol{y}_i = b_i \quad \text{where } b_i > 0, \quad i =, 1, \ldots, n \quad \text{(positive margins)}$$

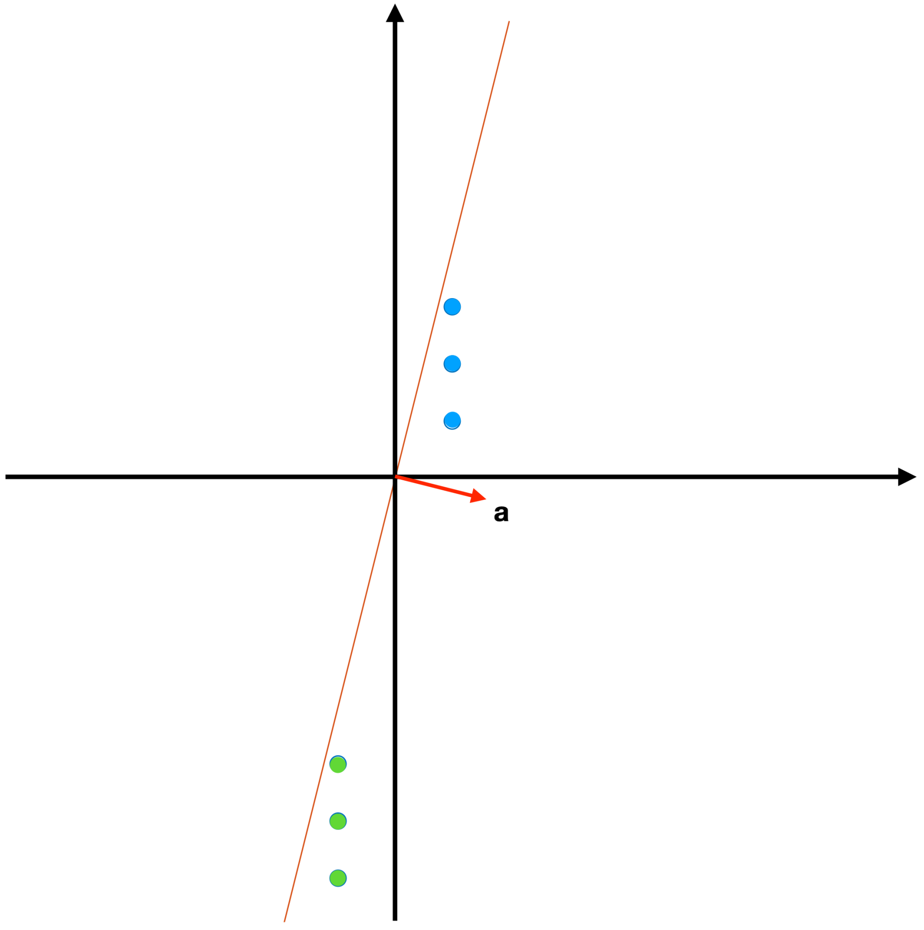
Find the minimum of a least-squares criterion function



$$J_s(\boldsymbol{a}) = \sum_{i=1}^{n} (\boldsymbol{a}^t \boldsymbol{y}_i - b_i)^2$$

Can be solved by gradient descent or by solving the *normal equations*.

Linear least-squares optimization (contd.)



The criterion function may be written as

$$J_s(\boldsymbol{a}) = \|\boldsymbol{e}\|^2 = \|Y\boldsymbol{a} - \boldsymbol{b}\|^2$$

where

$$Y = \begin{bmatrix} \boldsymbol{y}_1^t \\ \vdots \\ \boldsymbol{y}_n^t \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Zero gradient of the criterion function leads to

$$\boldsymbol{a} = (Y^tY)^{-1}Y^t\boldsymbol{b}$$

Artificial Neural Network (ANN)

Used in Machine Learning and Pattern Recognition:

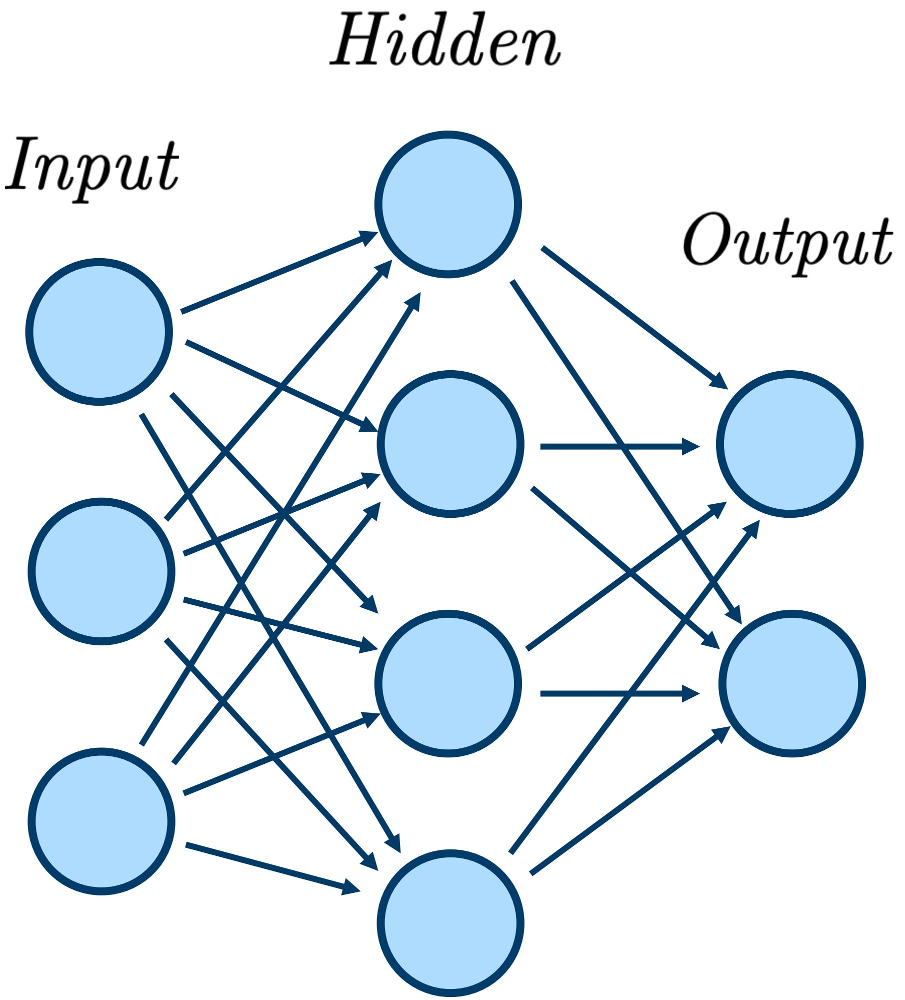
- Regression
- Classification
- Clustering
- …

Applications:

- Speech recognition
- Recognition of handwritten text
- Image classification
- …

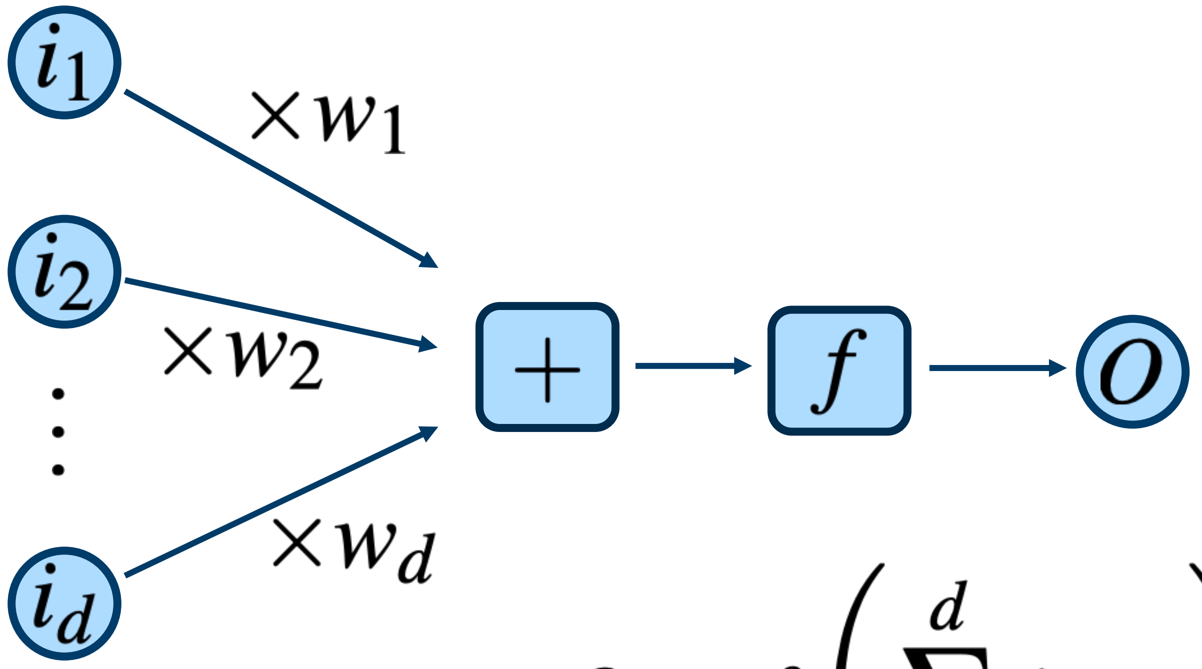
Network types:

- Feed-forward neural networks
- Recurrent neural networks (RNN)
- …



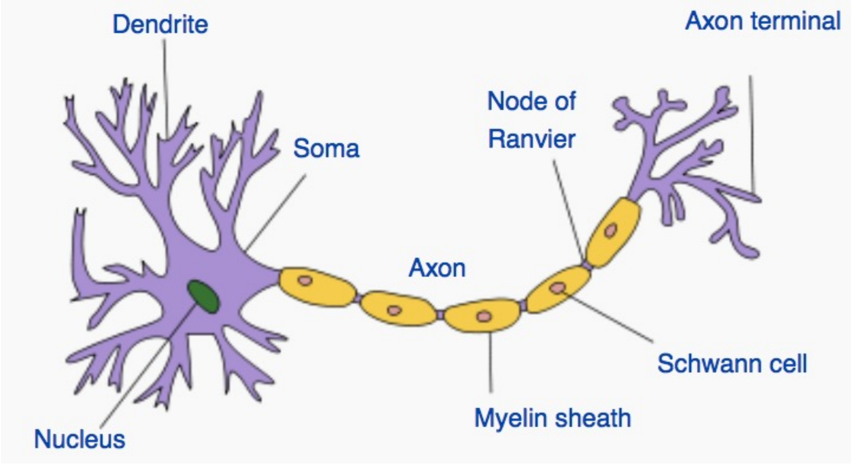
Feed-forward ANN (non-linear classifier)

Mark 1 Perceptron (Rosenblatt, 1957-59)



$$O = f\left(\sum_{k=1}^{d} i_k w_k\right)$$

Biological neuron

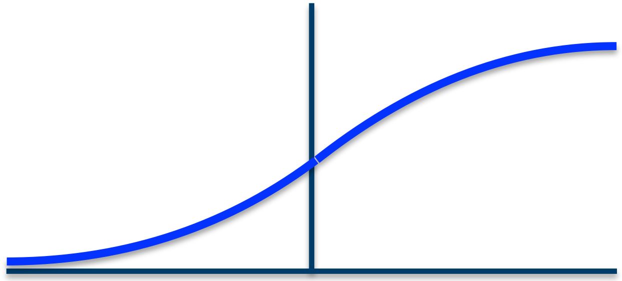


(Credit: Quasar Jarosz, English Wikipedia)

Activation functions

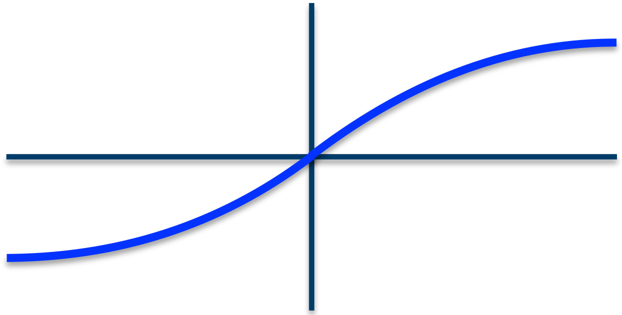
- Sigmoid (logistic function):

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



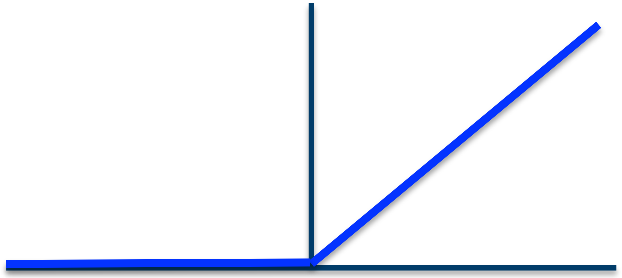
- Hyperbolic tangent:

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

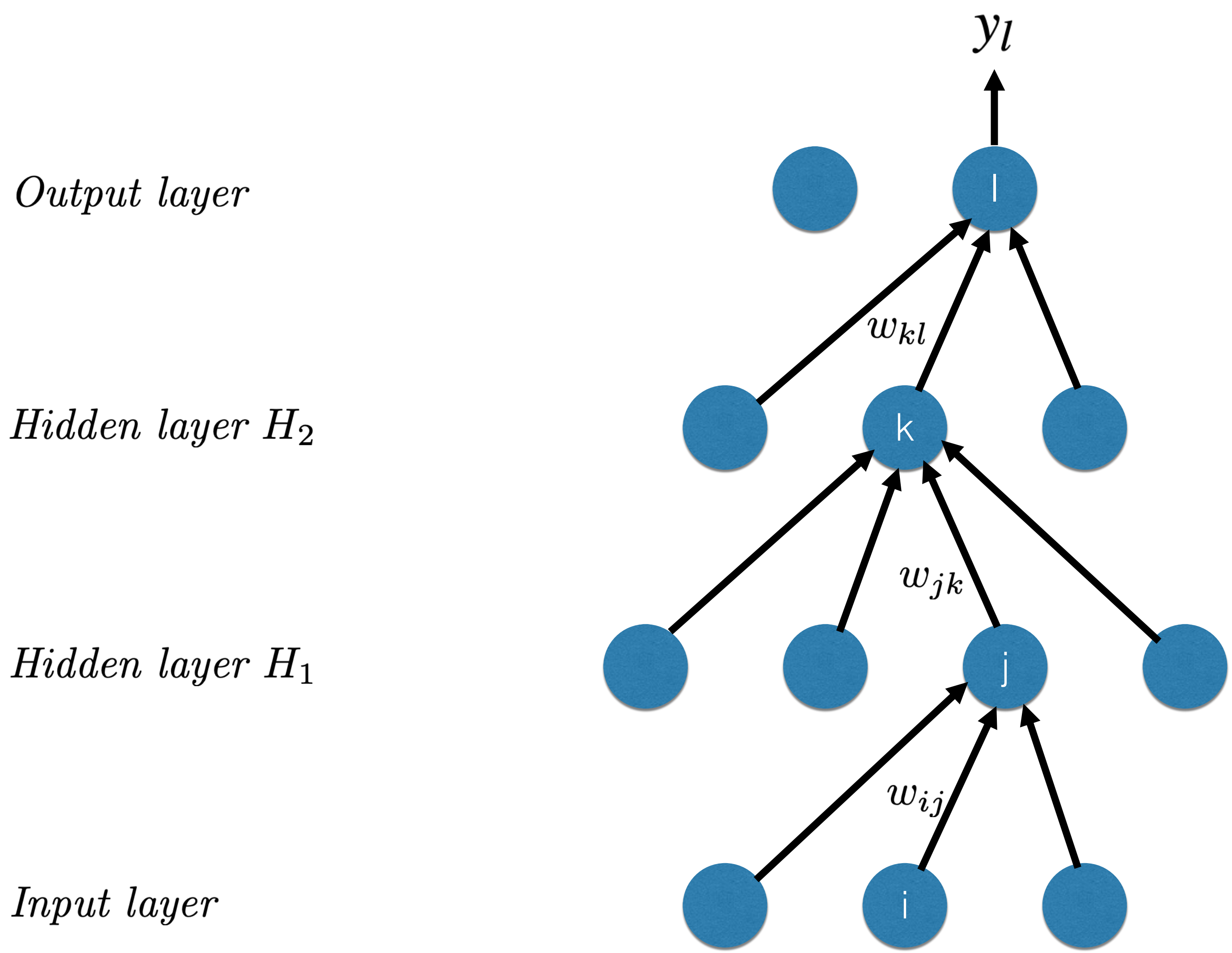


- Rectified linear unit (ReLU):

$$f(x) = \max(x, 0)$$



Feed-forward neural network



$$y_l = f(z_l)$$

$$z_l = \sum_{k \in H_2} w_{kl} x_k$$

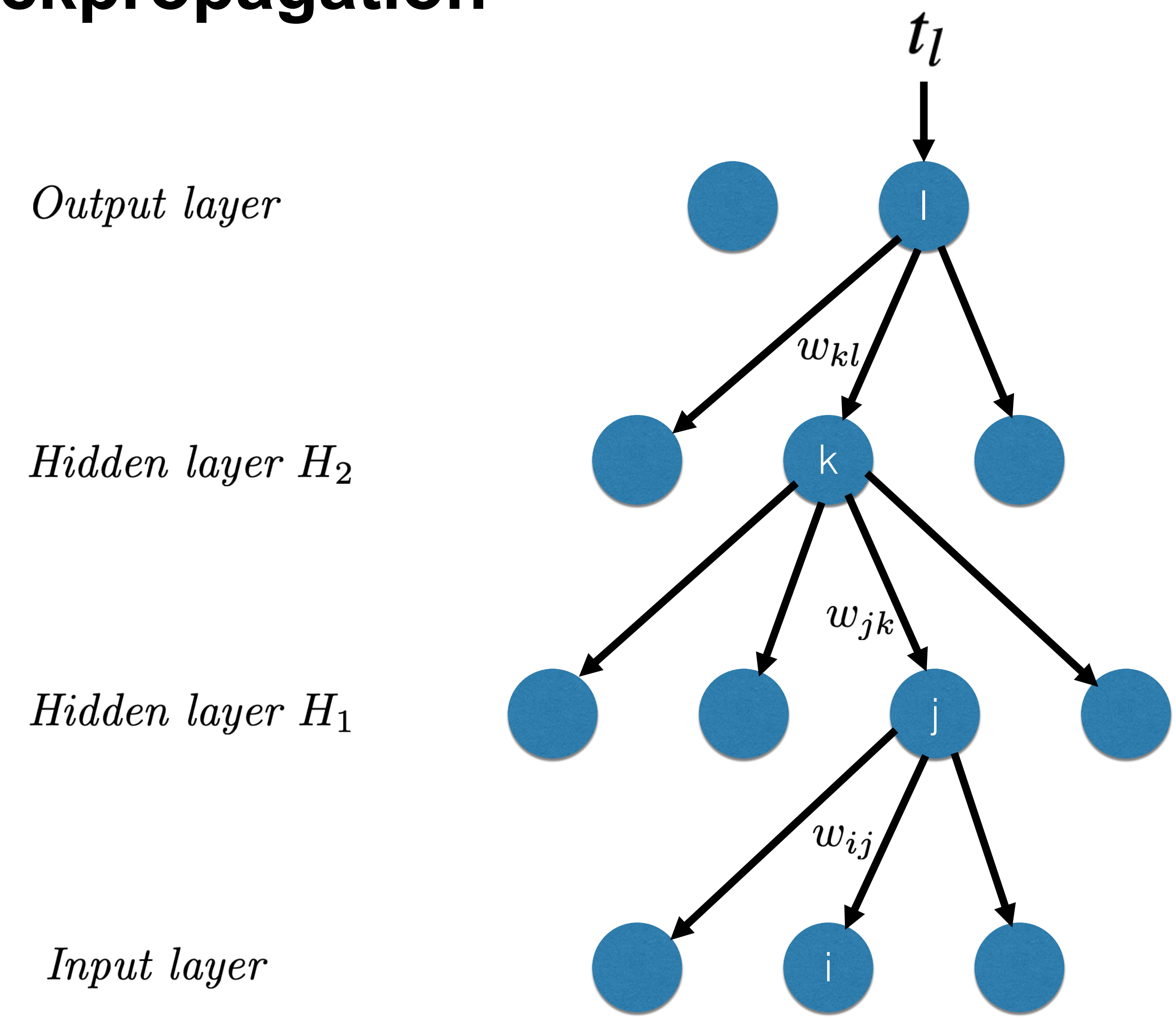
$$y_k = f(z_k)$$

$$z_k = \sum_{j \in H_1} w_{jk} x_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \in Input} w_{ij} x_i$$

Backpropagation



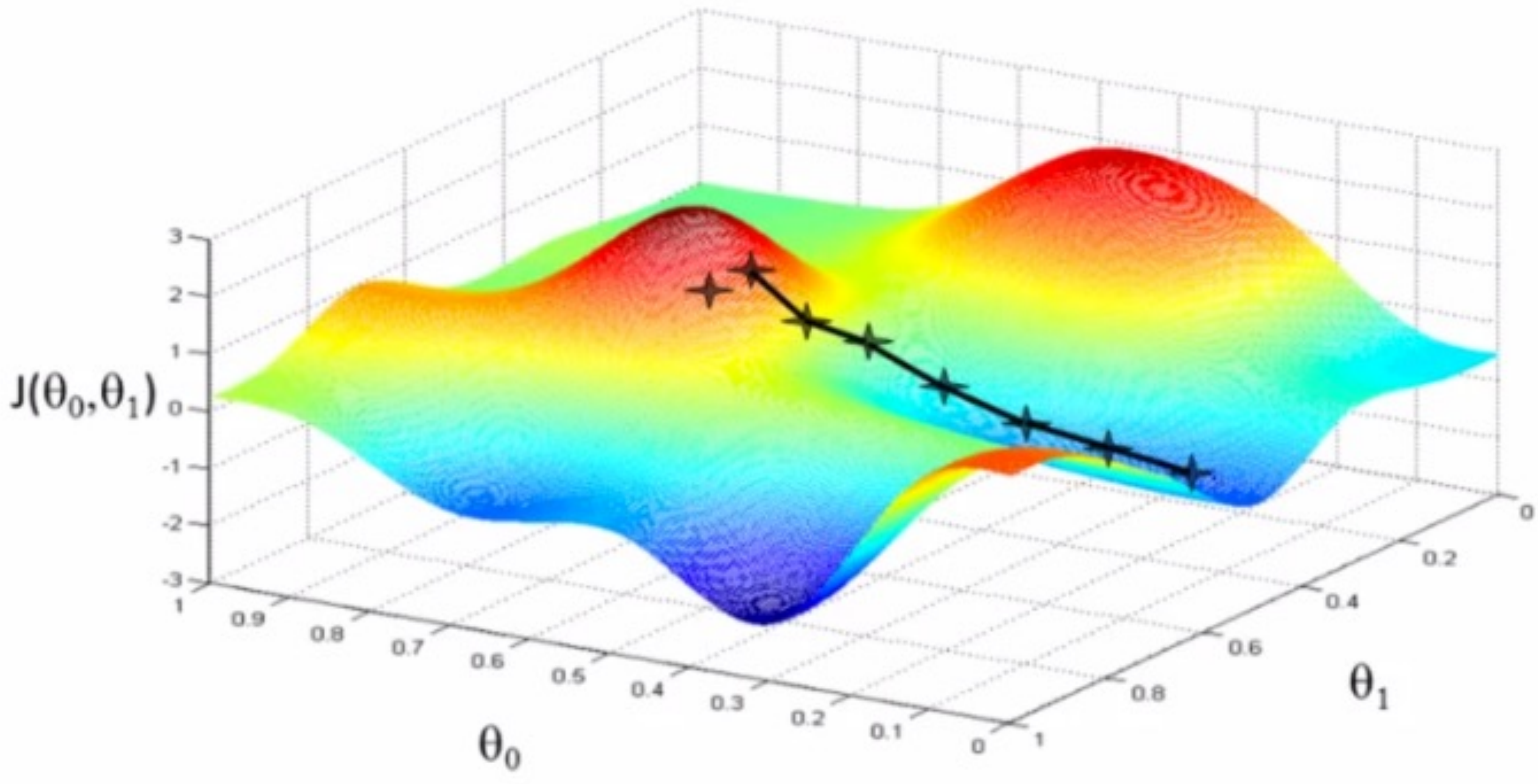
Error function (sample-by-sample measure of difference between output and target value):

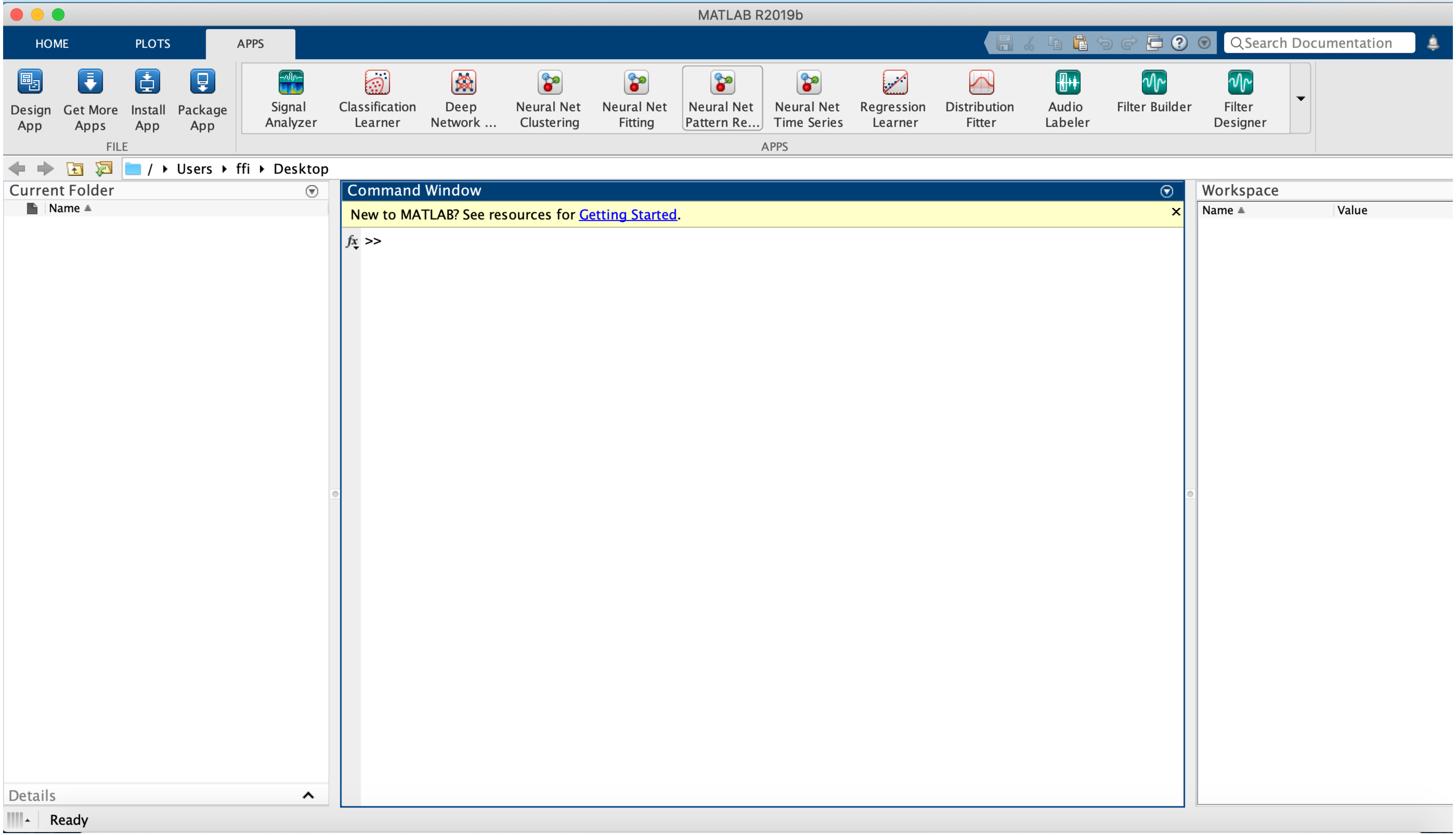
$$E(\boldsymbol{w}) = \sum_l (y_l - t_l)^2$$

Backpropagation:

- Compute the gradient of the error function with respect to the weights, using the chain rule.
- Adjust weights (gradient descent) level-by-level.

Gradient descent




MATLAB R2019b
HOME
PLOTS
APPS
Search Documentation
Design App
Get More Apps
Install App
Package App
FILE
Signal Analyzer
Classification Learner
Deep Network ...
Neural Net Clustering
Neural Net Fitting
Neural Net Pattern Re...
Neural Net Time Series
Regression Learner
Distribution Fitter
Audio Labeler
Filter Builder
Filter Designer
APPS
/ Users ffi Desktop
Current Folder
Name
Command Window
New to MATLAB? See resources for Getting Started.
>>
Workspace
Name
Value
Details
Ready

Neural Pattern Recognition (nprtool)

Welcome to the Neural Network Pattern Recognition app.

Solve a pattern-recognition problem with a two-layer feed-forward network.

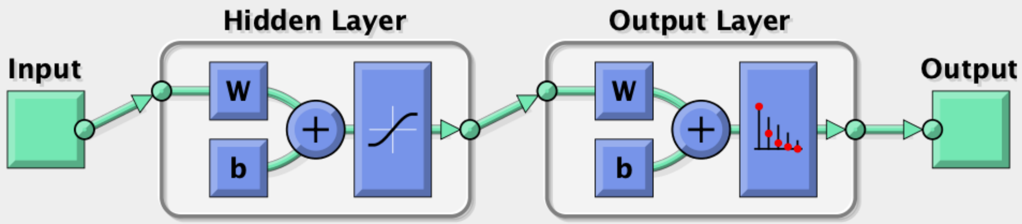
Introduction

In pattern recognition problems, you want a neural network to classify inputs into a set of target categories.

For example, recognize the vineyard that a particular bottle of wine came from, based on chemical analysis (wine_dataset); or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis (cancer_dataset).

The Neural Pattern Recognition app will help you select data, create and train a network, and evaluate its performance using cross-entropy and confusion matrices.

Neural Network

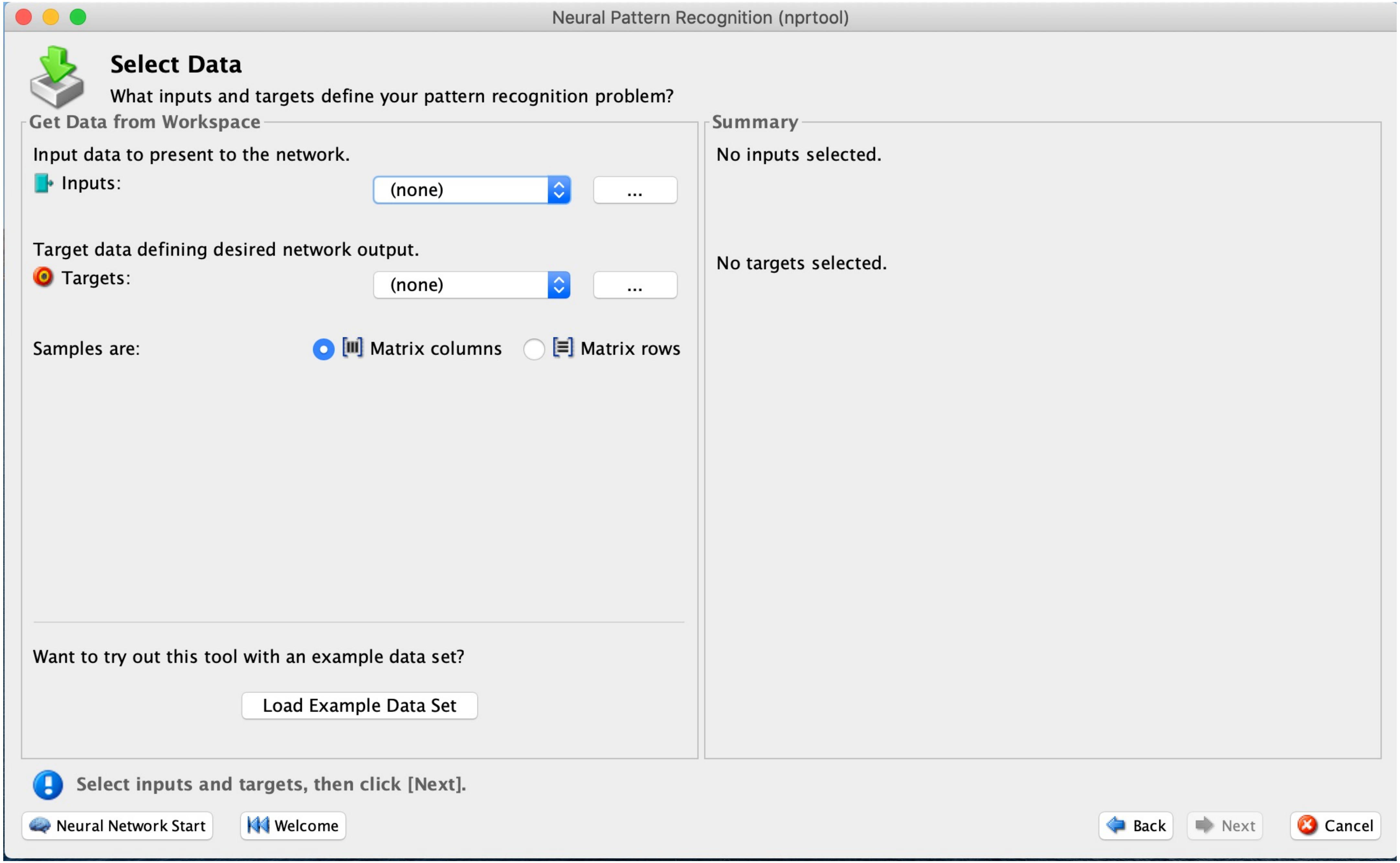


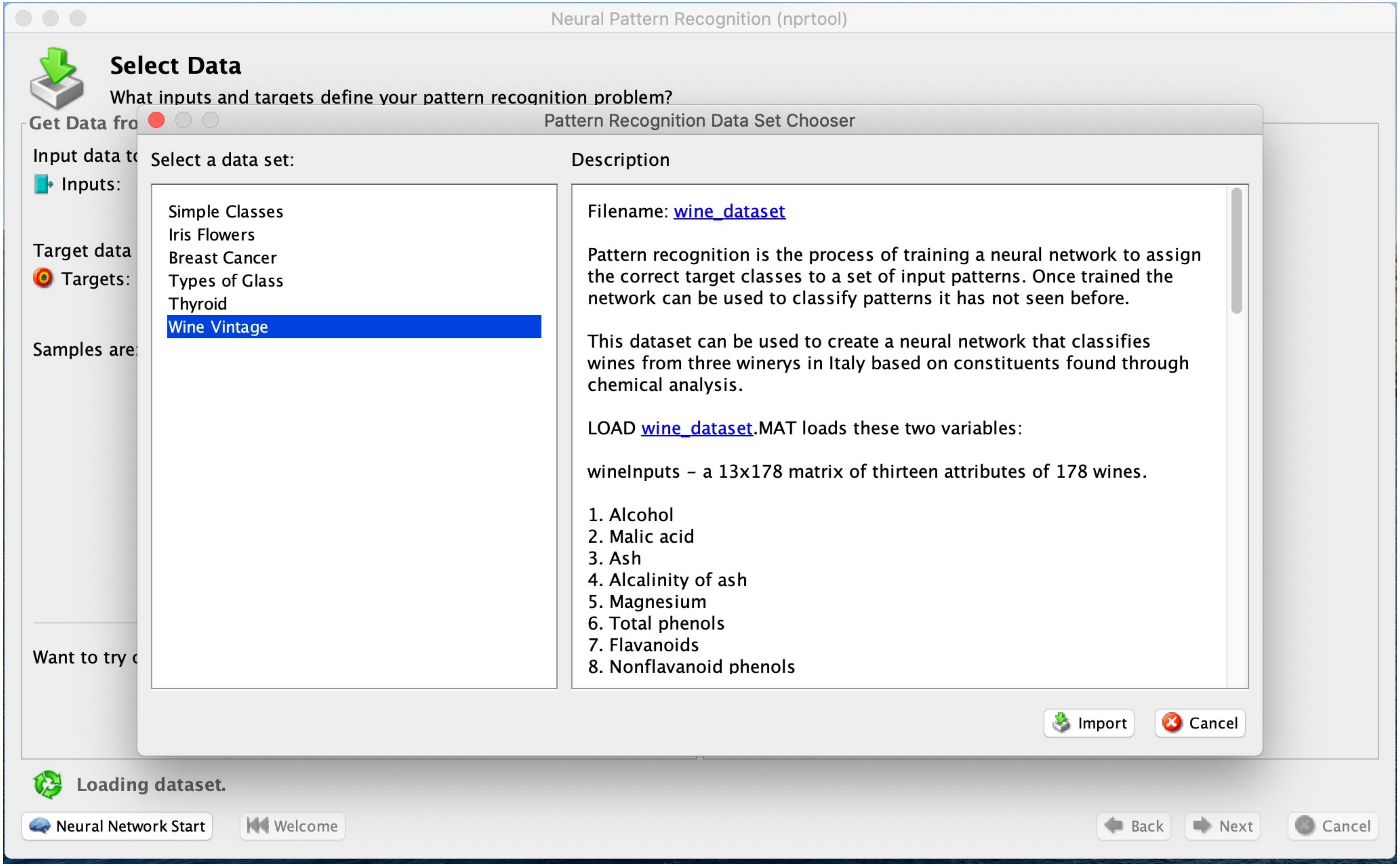
A two-layer feed-forward network, with sigmoid hidden and softmax output neurons (patternnet), can classify vectors arbitrarily well, given enough neurons in its hidden layer.

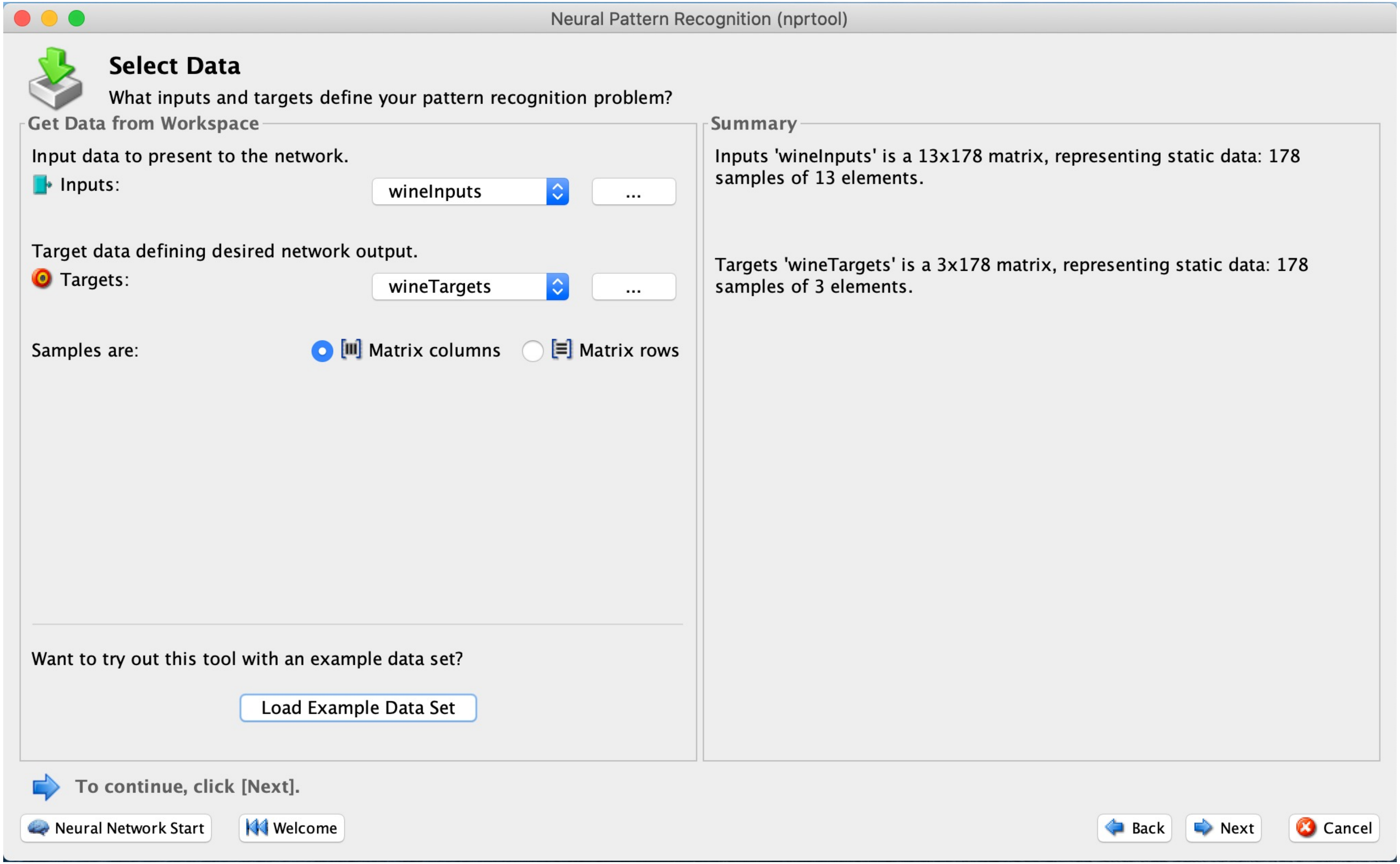
The network will be trained with scaled conjugate gradient backpropagation (trainscg).

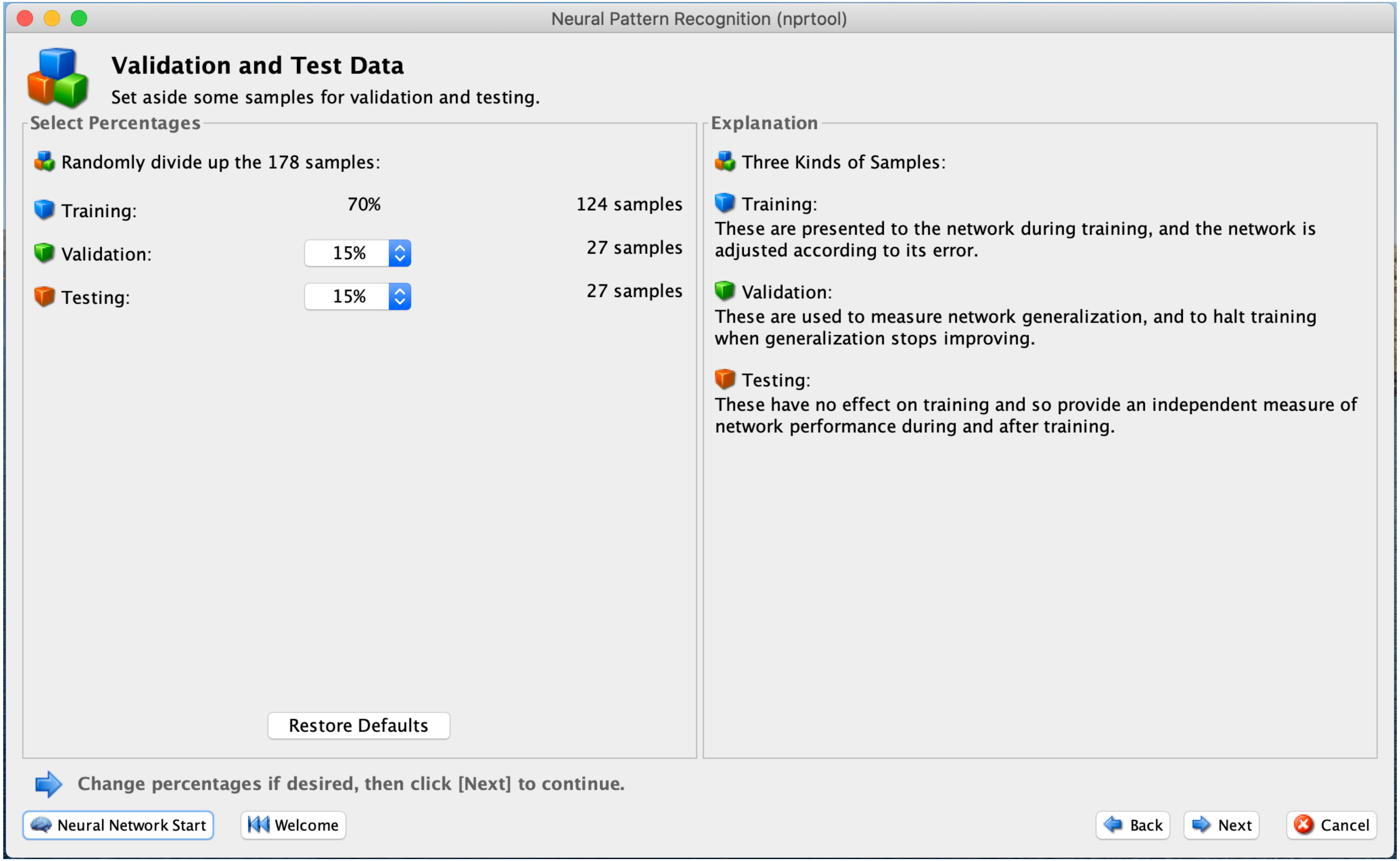
To continue, click [Next].

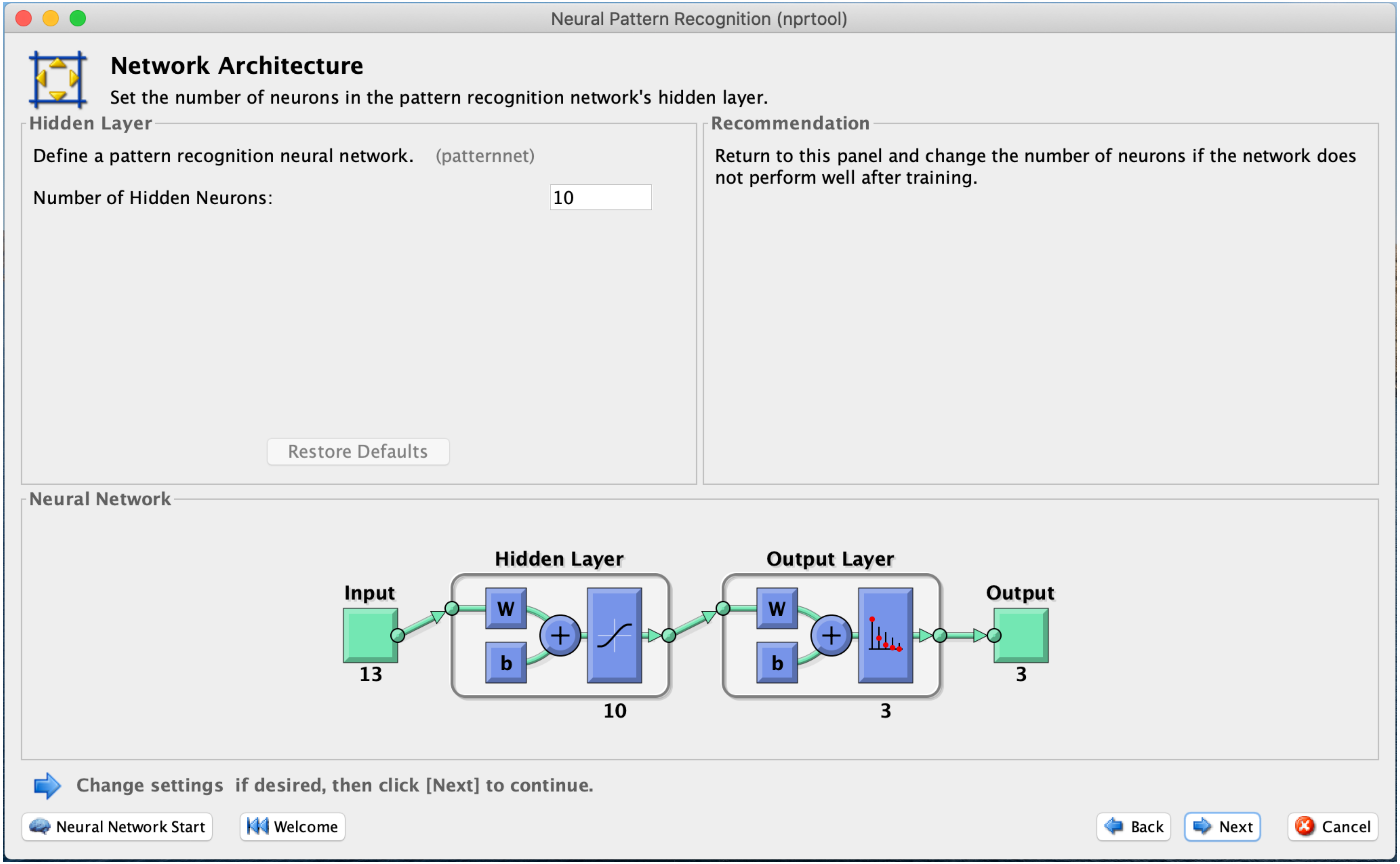
Neural Network Start | Welcome | Back | Next | Cancel

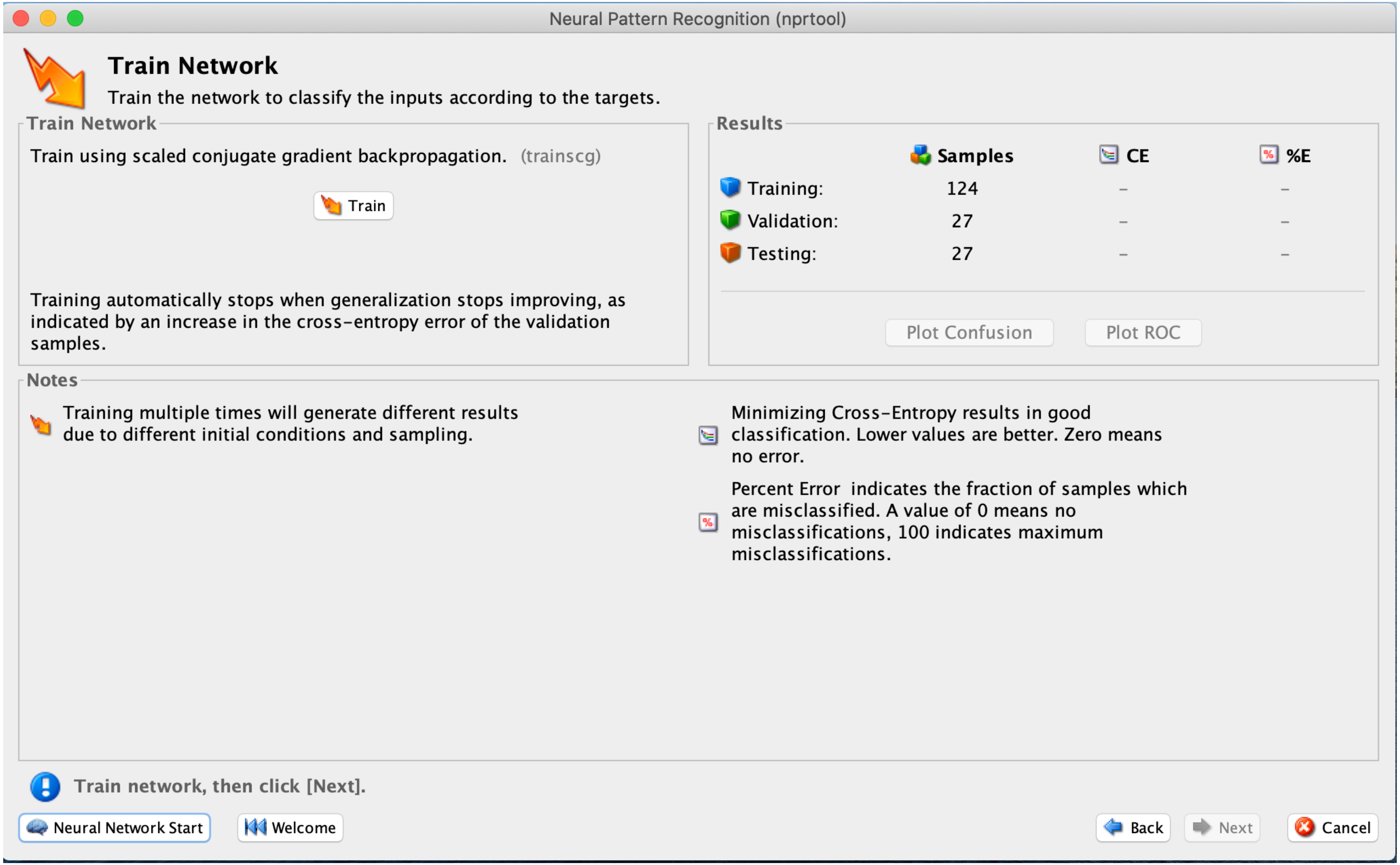

Neural Pattern Recognition (nprtool)
Select Data
What inputs and targets define your pattern recognition problem?
Get Data from Workspace
Input data to present to the network.
Inputs:
(none)
...
Target data defining desired network output.
Targets:
(none)
...
Samples are:
Matrix columns
Matrix rows
Want to try out this tool with an example data set?
Load Example Data Set
Summary
No inputs selected.
No targets selected.
Select inputs and targets, then click [Next].
Neural Network Start
Welcome
Back
Next
Cancel


Neural Pattern Recognition (nprtool)
Select Data
What inputs and targets define your pattern recognition problem?
Get Data fro
Input data to
Inputs:
Target data
Targets:
Samples are:
Want to try
Pattern Recognition Data Set Chooser
Select a data set:
Simple Classes
Iris Flowers
Breast Cancer
Types of Glass
Thyroid
Wine Vintage
Description
Filename: wine_dataset
Pattern recognition is the process of training a neural network to assign the correct target classes to a set of input patterns. Once trained the network can be used to classify patterns it has not seen before.
This dataset can be used to create a neural network that classifies wines from three winerys in Italy based on constituents found through chemical analysis.
LOAD wine_dataset.MAT loads these two variables:
wineInputs - a 13x178 matrix of thirteen attributes of 178 wines.
1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
Import
Cancel
Loading dataset.
Neural Network Start
Welcome
Back
Next
Cancel


Neural Pattern Recognition (nprtool)
Select Data
What inputs and targets define your pattern recognition problem?
Get Data from Workspace
Input data to present to the network.
Inputs:
wineInputs
...
Target data defining desired network output.
Targets:
wineTargets
...
Samples are:
Matrix columns
Matrix rows
Want to try out this tool with an example data set?
Load Example Data Set
Summary
Inputs 'wineInputs' is a 13x178 matrix, representing static data: 178 samples of 13 elements.
Targets 'wineTargets' is a 3x178 matrix, representing static data: 178 samples of 3 elements.
To continue, click [Next].
Neural Network Start
Welcome
Back
Next
Cancel


Neural Pattern Recognition (nprtool)
Validation and Test Data
Set aside some samples for validation and testing.
Select Percentages
Randomly divide up the 178 samples:
Training: 70% 124 samples
Validation: 15% 27 samples
Testing: 15% 27 samples
Restore Defaults
Explanation
Three Kinds of Samples:
Training:
These are presented to the network during training, and the network is adjusted according to its error.
Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.
Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.
Change percentages if desired, then click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel


Neural Pattern Recognition (nprtool)
Network Architecture
Set the number of neurons in the pattern recognition network's hidden layer.
Hidden Layer
Define a pattern recognition neural network. (patternnet)
Number of Hidden Neurons: 10
Restore Defaults
Recommendation
Return to this panel and change the number of neurons if the network does not perform well after training.
Neural Network
Hidden Layer
Output Layer
Input
Output
W
b
W
b
13
10
3
3
Change settings if desired, then click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel



Neural Pattern Recognition (nprtool)
Train Network
Train the network to classify the inputs according to the targets.
Train Network
Train using scaled conjugate gradient backpropagation. (trainscg)
Train
Training automatically stops when generalization stops improving, as indicated by an increase in the cross-entropy error of the validation samples.
Results
Samples
CE
%E
Training: 124 – –
Validation: 27 – –
Testing: 27 – –
Plot Confusion
Plot ROC
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Minimizing Cross-Entropy results in good classification. Lower values are better. Zero means no error.
Percent Error indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.
Train network, then click [Next].
Neural Network Start
Welcome
Back
Next
Cancel


Neural Pattern Recognition (nprtool)
Train Network
Train the network to classify the inputs according to the targets.
Train Network
Train using scaled conjugate gradient backpropagation. (trainscg)
Retrain
Training automatically stops when generalization stops improving, as indicated by an increase in the cross-entropy error of the validation samples.
Results
Samples
CE
%E
Training: 124 9.86550e-1 0
Validation: 27 2.74688e-0 11.11111e-0
Testing: 27 2.66457e-0 0
Plot Confusion
Plot ROC
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Minimizing Cross-Entropy results in good classification. Lower values are better. Zero means no error.
Percent Error indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.
Open a plot, retrain, or click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help
Training Confusion Matrix
43 34.7%
0 0.0%
0 0.0%
100% 0.0%
0 0.0%
47 37.9%
0 0.0%
100% 0.0%
0 0.0%
0 0.0%
34 27.4%
100% 0.0%
100% 0.0%
100% 0.0%
100% 0.0%
100% 0.0%
Output Class
Target Class
Validation Confusion Matrix
5 18.5%
2 7.4%
0 0.0%
71.4% 28.6%
0 0.0%
13 48.1%
0 0.0%
100% 0.0%
0 0.0%
1 3.7%
6 22.2%
85.7% 14.3%
100% 0.0%
81.2% 18.8%
100% 0.0%
88.9% 11.1%
Test Confusion Matrix
11 40.7%
0 0.0%
0 0.0%
100% 0.0%
0 0.0%
8 29.6%
0 0.0%
100% 0.0%
0 0.0%
0 0.0%
8 29.6%
100% 0.0%
100% 0.0%
100% 0.0%
100% 0.0%
100% 0.0%
All Confusion Matrix
59 33.1%
2 1.1%
0 0.0%
96.7% 3.3%
0 0.0%
68 38.2%
0 0.0%
100% 0.0%
0 0.0%
1 0.6%
48 27.0%
98.0% 2.0%
100% 0.0%
95.8% 4.2%
100% 0.0%
98.3% 1.7%

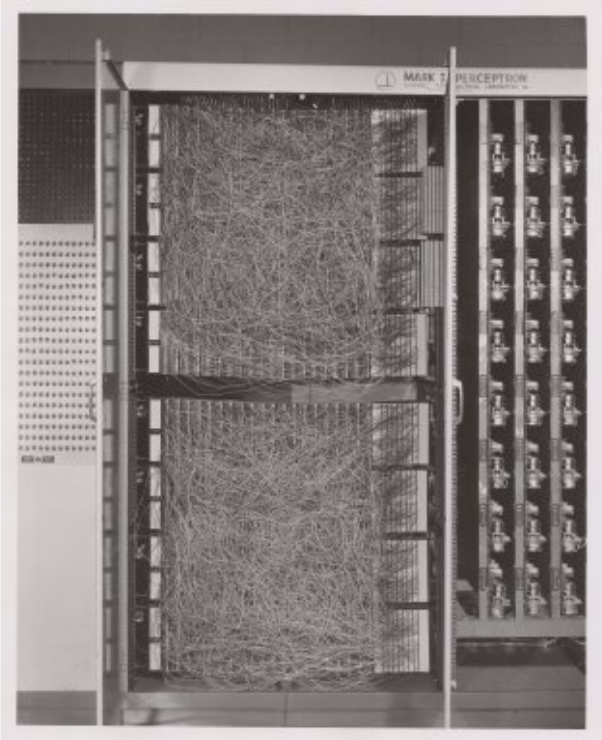
Summary

Machine learning:
- Pattern classification
- Training of classifiers (supervised learning)
- Parametric and non-parametric methods
- Discriminant functions
- Quadratic and linear classifiers
- Neural Networks.

Recommended reading:
- Szeliski 5.1 - 5.3

Additional reading:
- Szeliski 6.1 - 6.3
- R. O. Duda, P. E. Hart, D. G. Stork (2001). *Pattern classification* (2nd ed.). Wiley, New York. ISBN 0-471-05669-3.

Mark 1 Perceptron



(Credit: Cornell Aeronautical Laboratory)