INF5390 - Kunstig intelligens

Neural Networks and Support Vector Machines

Roar Fjellheim

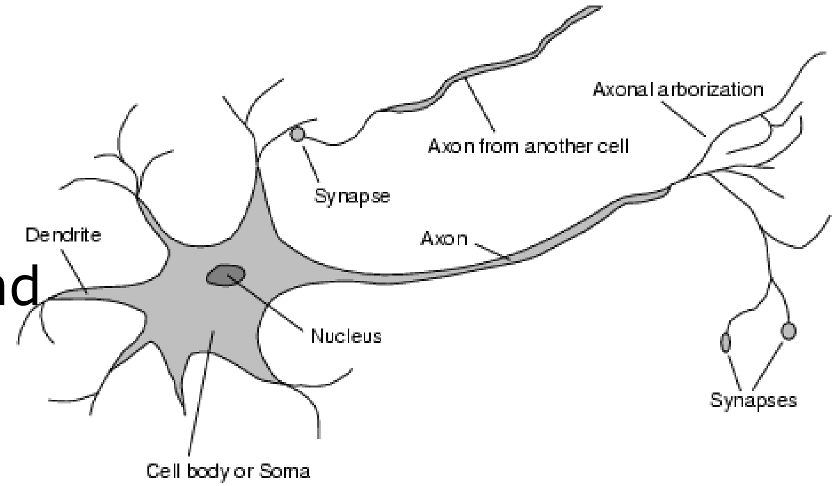
Outline

- Neural networks
- Perceptrons
- Neural networks
- Support vector machines
- Summary

AIMA Chapter 18: Learning from Examples

Neural networks in AI

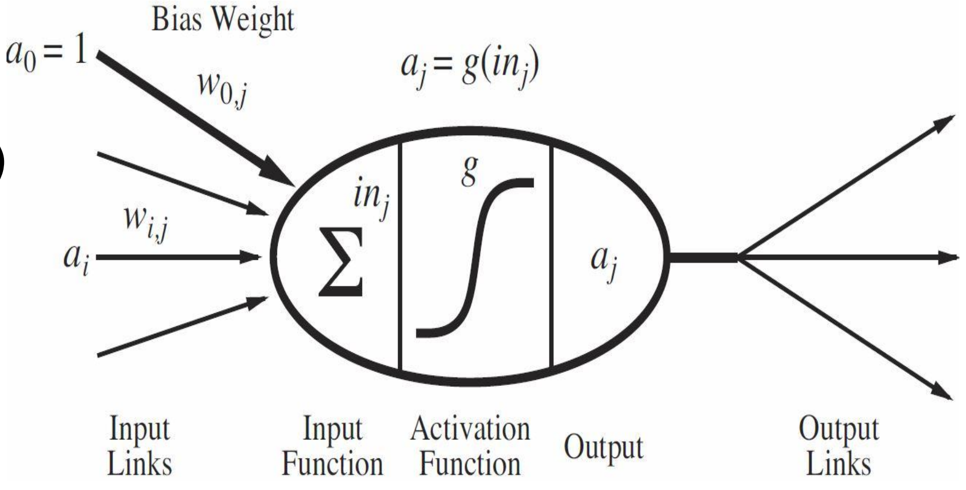
- The human brain is a huge network of *neurons*
  - A neuron is a basic processing unit that collects, processes and disseminates electrical signals



- Early AI tried to imitate the brain by building *artificial neural networks* (ANN)
  - Met with theoretical limits and ”disappeared”
- In the 1980-90’es, interest in ANNs resurfaced
  - New theoretical development
  - Massive industrial interest&applications

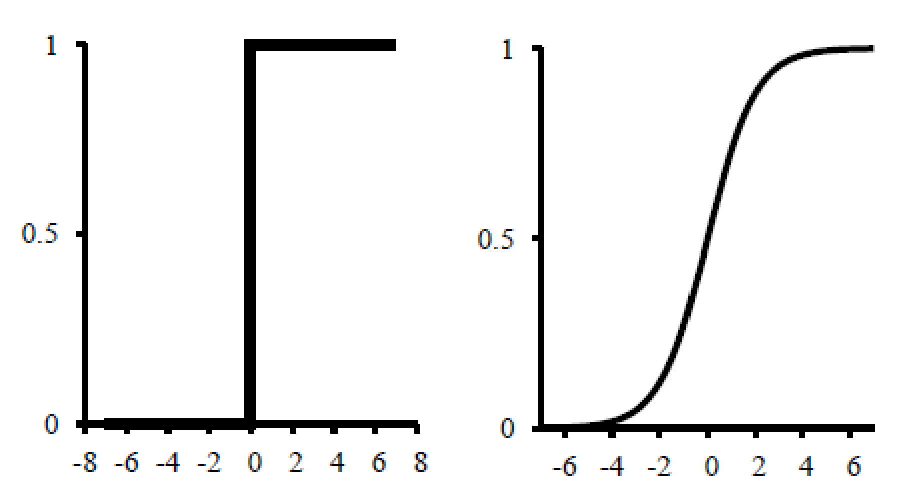
The basic unit of neural networks

- The network consists of *units* (nodes, "neurons") connected by *links*
  - √ Carries an *activation* a_i from unit i to unit j
  - √ The link from unit i to unit j has a *weight* $W_{i,j}$
  - √ *Bias* weight $W_{0,j}$ to fixed input $a_0 = 1$
- *Activation* of a unit j
  - √ Calculate input $in_j = \sum W_{i,j}\, a_i \quad (i=0..n)$
  - √ Derive output $a_j = g(in_j)$ where g is the *activation function*



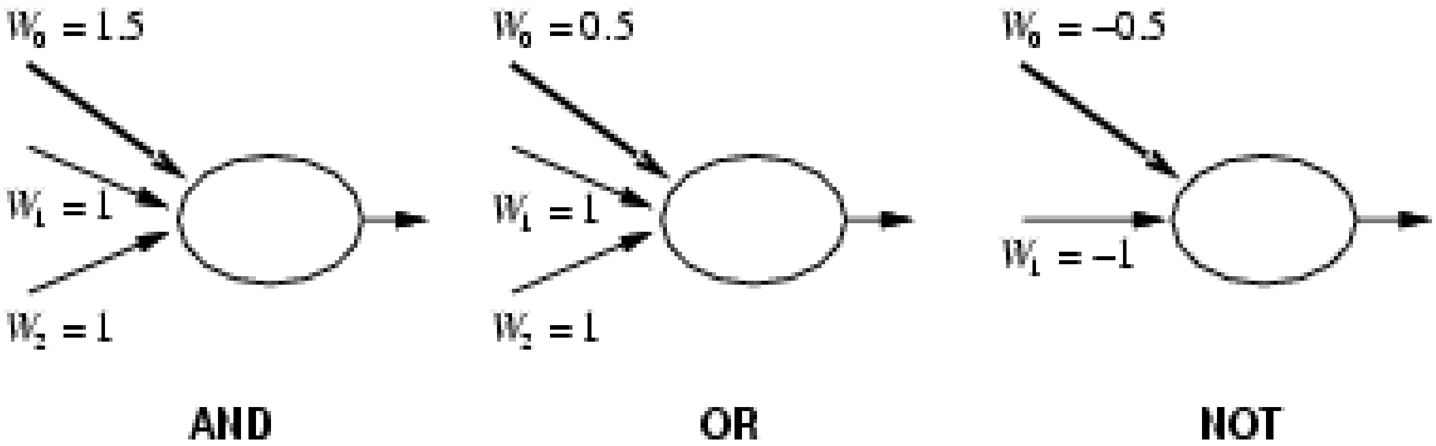
Activation functions

- Activation function should separate well
  - √ ”Active” (near 1) for desired input
  - √ ”Inactive” (near 0) otherwise
- It should be *non-linear*
- Most used functions
  - √ *Threshold* function
  - √ *Sigmoid* function



Neural networks as logical gates

- With proper use of *bias weight* W_0 to set thresholds, neural networks can compute standard logical gate functions

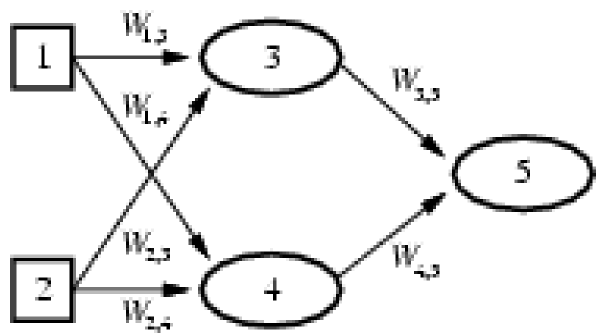


Neural network structures

- Two main structures
  - *Feed-forward* (acyclic) networks
    - Represents a function of its inputs
    - No internal state
  - *Recurrent* network
    - Feeds outputs back to inputs
    - May be stable, oscillate or become chaotic
    - Output depends on initial state
- Recurrent networks are the most interesting and ”brain-like”, but also most difficult to understand

Feed-forward networks as functions

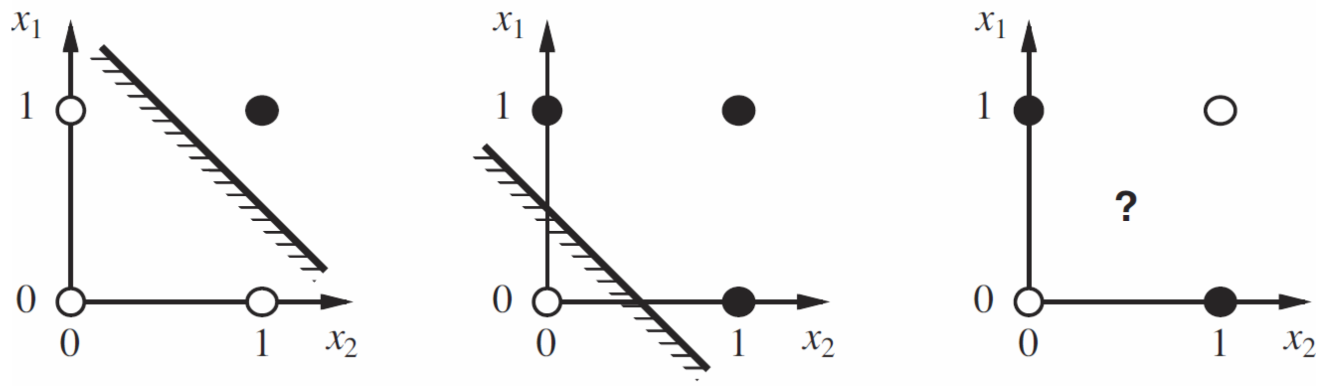
- A FF network calculates a *function* of its inputs
- The network may contain *hidden* units/layers



- By changing #layers/units and their weights, different functions can be realized
- FF networks are often used for *classification*

Perceptrons

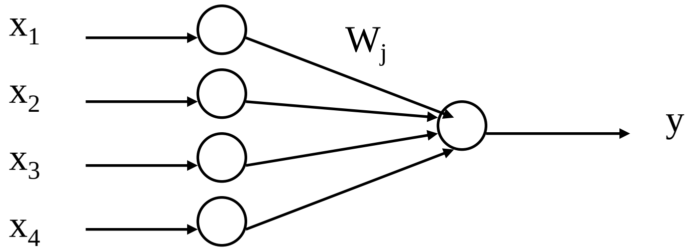
- Single-layer feed-forward neural networks are called *perceptrons*, and were the earliest networks to be studied
- Perceptrons can only act as *linear separators*, a small subset of all interesting functions
  - √ This partly explains why neural network research was discontinued for a long time



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

Perceptron learning algorithm

- How to train the network to do a certain function (e.g. *classification*) based on *a training set* of input/output pairs?



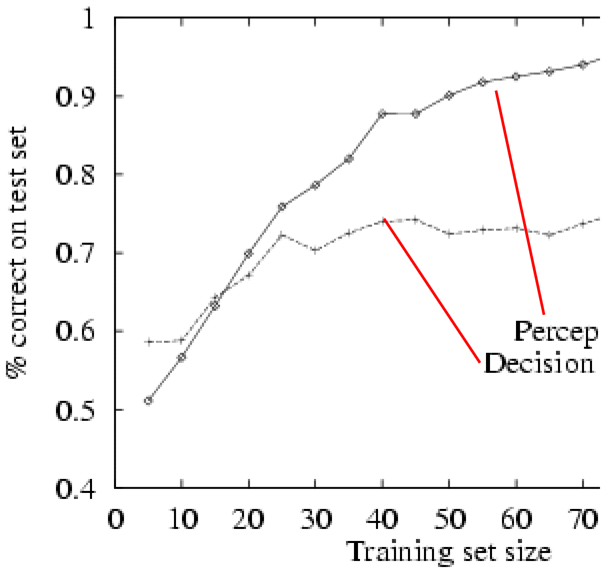
- Basic idea
  - √ Adjust network link weights to minimize some measure of the error on the training set
  - √ Adjust weights in direction that minimizes error

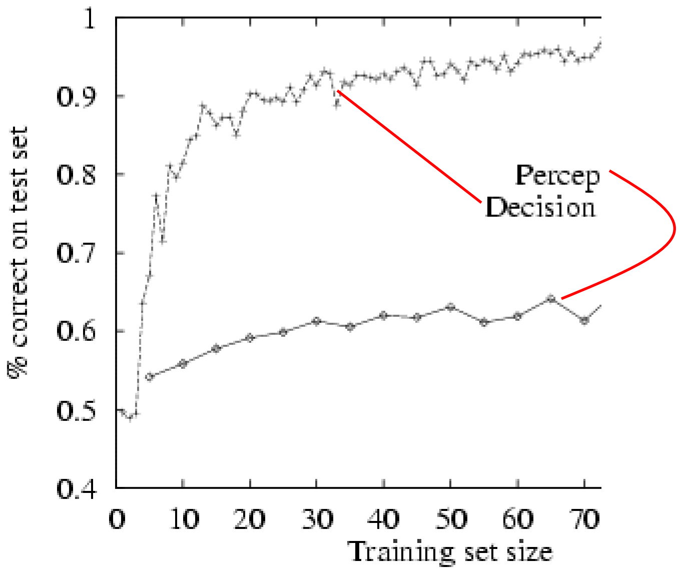
Perceptron learning algorithm (cont.)

function PERCEPTRON-LEARNING(*examples, network*)
returns a perceptron hypothesis
inputs: *examples*, a set of examples, each with inputs x_1, x_2 .. and output y
network, a perceptron with weights W_j and act. function g
repeat
for each e **in** *examples* **do**
$in = \sum W_j \, x_j[e]$ j=0 .. n
$Err = y[e] - g(in)$
$W_j = W_j + \alpha \, Err \, x_j[e]$ α - the *learning rate*
until some stopping criterion is satisfied
return NEURAL-NETWORK-HYPOTHESIS(*network*)

Performance of perceptrons vs. decision trees

- Perceptrons better at learning separable problem
- Decision trees better at ”restaurant problem”



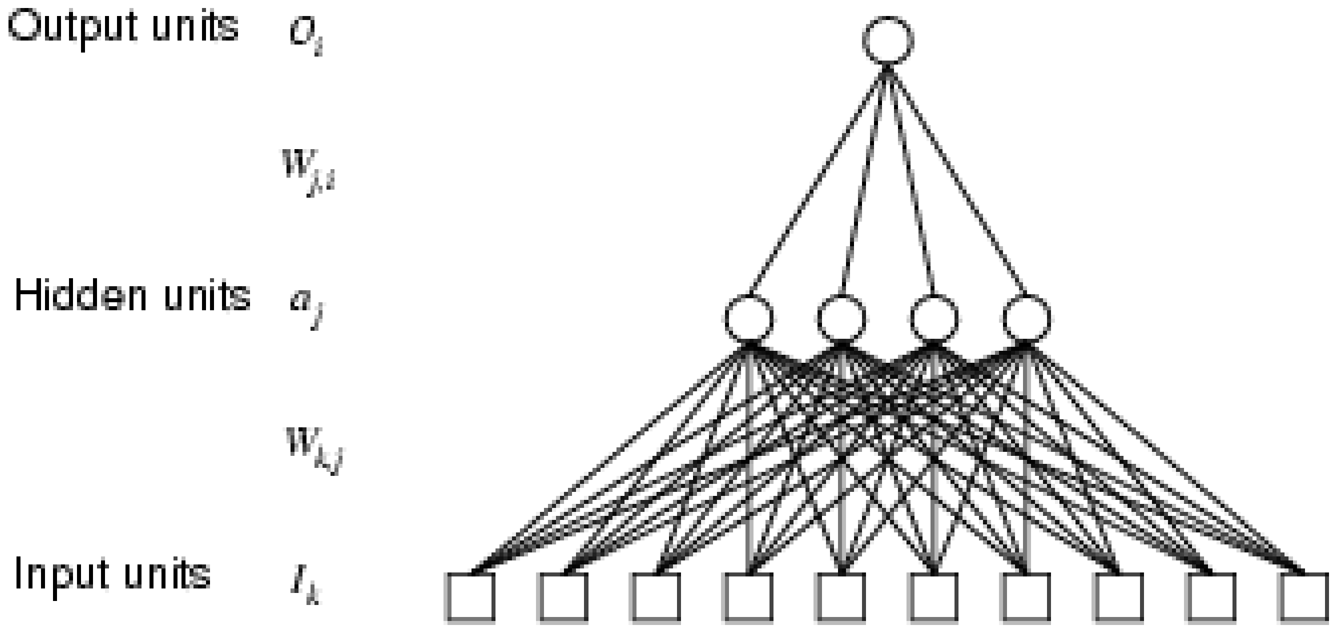


Multi-layer feed-forward networks

- Adds *hidden* layers
  - The most common is one extra layer
  - The advantage is that more function can be realized, in effect by combining several perceptron functions
- It can be shown that
  - A feed-forward network with a single sufficiently large hidden layer can represent any *continuous* function
  - With two layers, even *discontinuous* functions can be represented
- However
  - Cannot easily tell which functions a particular network is able to represent
  - Not well understood how to choose structure/number of layers for a particular problem

Example network structure

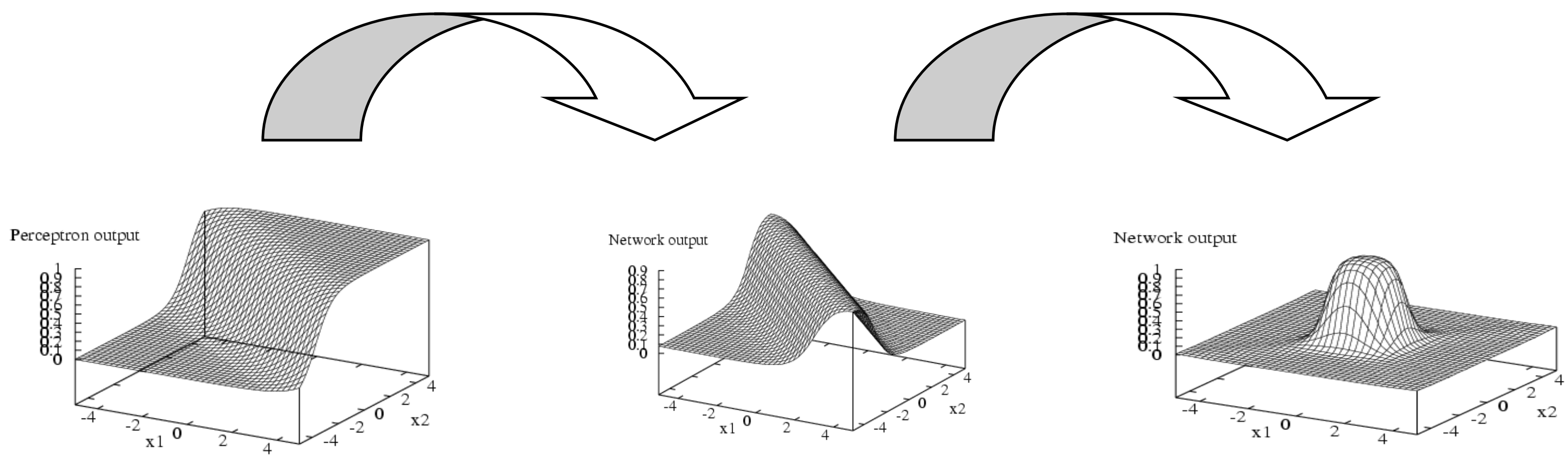
- Feed-forward network with 10 inputs, one output and one hidden layer – suitable for ”restaurant problem”



More complex activation functions

- Multi-layer networks can combine simple (linear separation) perceptron activation functions into more complex functions

(combine 2) *(combine 2)*



Learning in multi-layer networks

- In principle as for perceptrons – adjusting weights to minimize error
- The main difference is what ”error” at internal nodes mean – nothing to compare to
- Solution: *Propagate* error at output nodes back to hidden layers
  - Successively propagate backwards if the network has several hidden layers
- The resulting *Back-propagation algorithm* is the standard learning method for neural networks

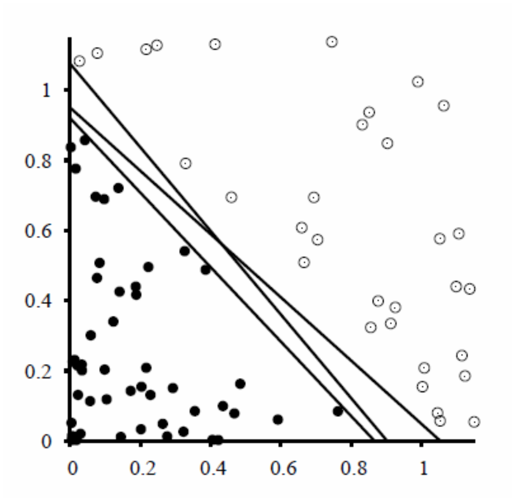
Learning neural network structure

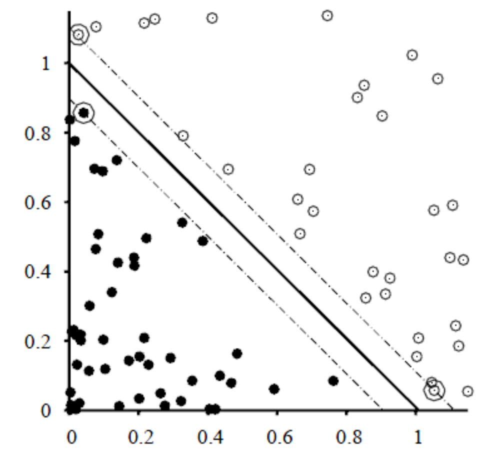
- Need to learn network structure
  - Learning algorithms have assumed fixed network structure
  - However, we do not know in advance what structure will be necessary and sufficient
- Solution approach
  - Try different configurations, keep the best
  - Search space is very large (# layers and # nodes)
  - ”Optimal brain damage”: Start with full network , remove nodes selectively (optimally)
  - ”Tiling”: Start with minimal network that covers subset of training set, expand incrementally

Support Vector Machines (SVM)

- Currently the most popular approach for supervised learning
  - √ Does not require any prior knowledge
  - √ Scales to very large problems
- Attractive features of SVM
  - √ Constructs a *maximum margin separator*, decision boundary with max. possible distance to examples
  - √ Creates linear separators, but can embed data in higher dimensions (*kernel trick*)
  - √ A *non-parametric* method, i.e. may retain examples (instances, in addition to parameters as in NN), thus be able to express more complex functions

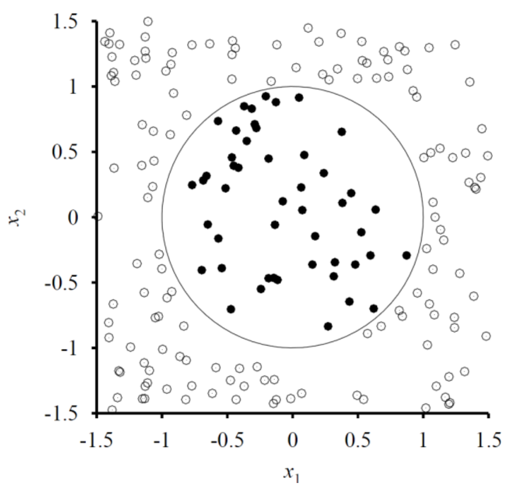
Classification by SVM

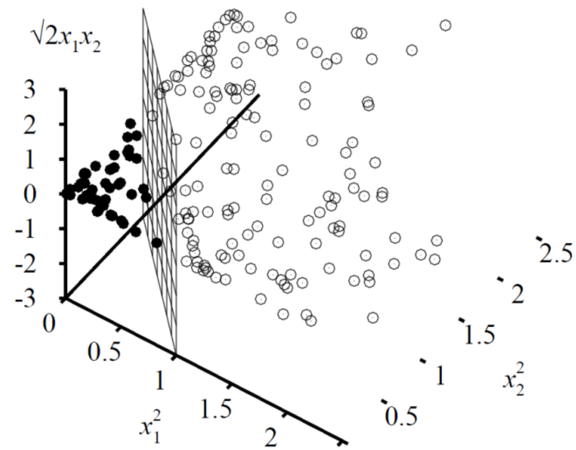




- SVM finds the separator with *maximum margin* between examples
- The example points nearest the separator are called *support vectors*

The kernel trick in SVM

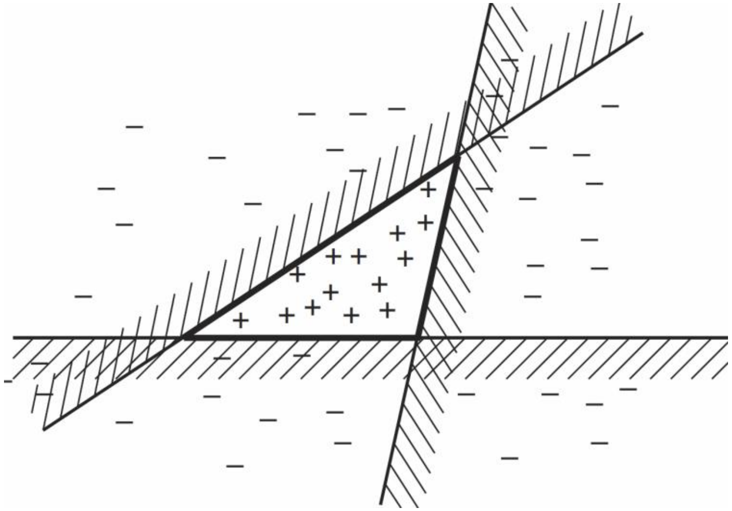




- What if the examples are not linearly separable?
- SVM maps each example to a new vector in a higher dimension space, using *kernel functions*
- In the new space, a linear maximum separator may be found (the *kernel trick*)

Ensemble learning (EM)

- In *ensemble learning* predictions from a *collection* of hypotheses are combined
- Example: Three linear separators are combined such that all separators must return positive for the overall classification to be positive
- The combined classifier is more expressive without being much more complex
- *Boosting* is a widely used ensemble learning method



Summary

- Neural networks (NN) are inspired by human brains, and are complex nonlinear functions with many parameters learned from noisy data
- A perceptron is a feed-forward network with no hidden layers and can only represent linearly separable functions
- Multi-layer feed-forward NN can represent arbitrary functions, and be trained efficiently using the back-propagation algorithm
- Support vector machines (SVM) is an effective method for learning classifiers in large data sets
- Ensemble learning (EM) combines several simpler classifiers in a more complex function