Introduction to Data Mining

Data Mining
Classification: Advanced Techniques

Overview

1. Brief recap
2. Computational power of MLP
   - Linear and non-linear separability
3. Dealing with more complex data
   - Sequences
   - Tree structured data
4. Conclusion on Classification

Linear separability

How does a (trained) MLP perform classification?

Lets consider a simple MLP consisting of a single layer. In this case (i.e. no hidden layers), the network produces an output as follows:

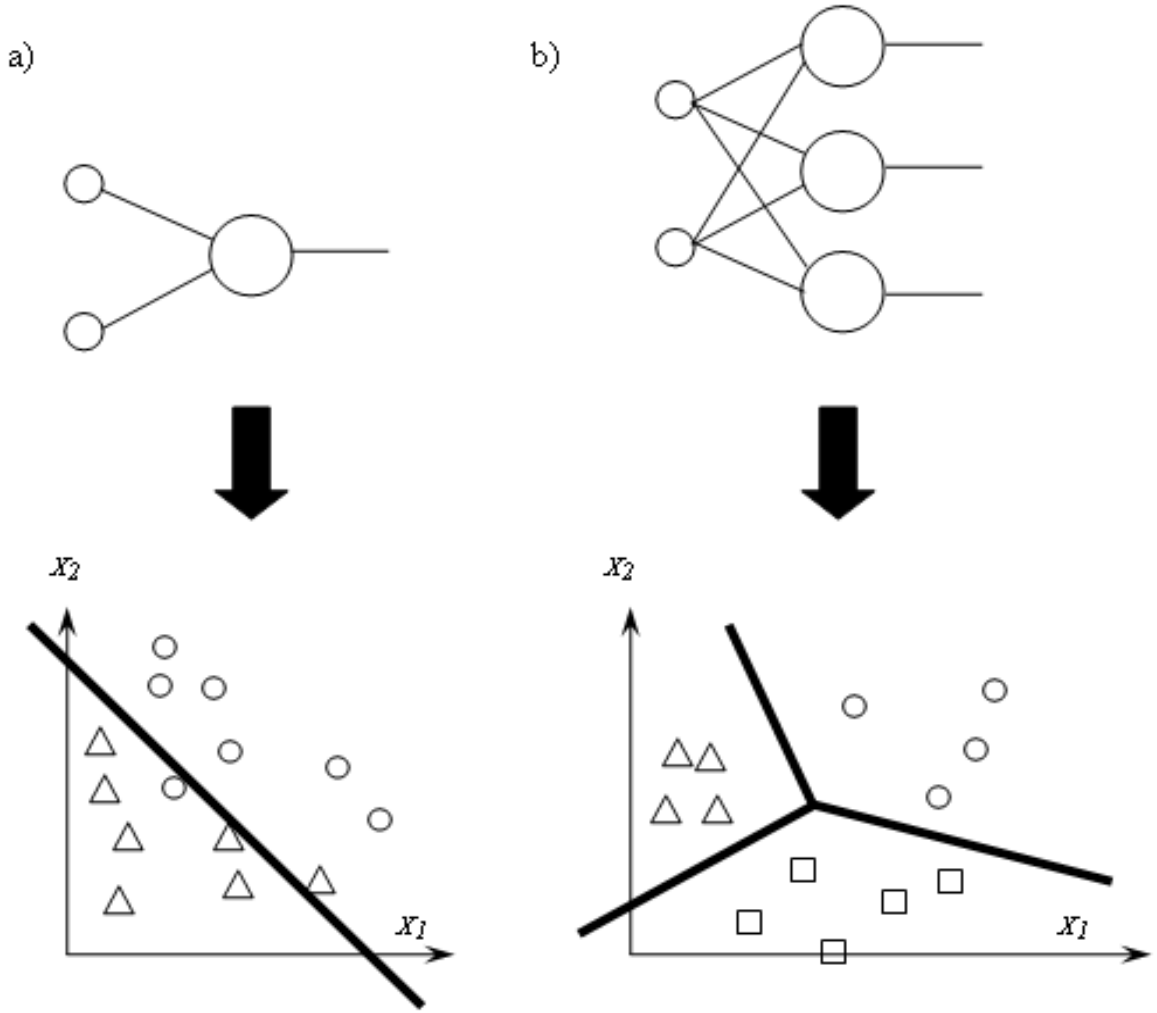
$$o_i = \sum_{j=1}^{n} w_{ji} x_j$$

The MLP is a realization of a parameterized function *f(x)*. In case of a one layer MLP, this function produces *k* hyperplanes of dimension n-1, where *k* is the number of output neurons.

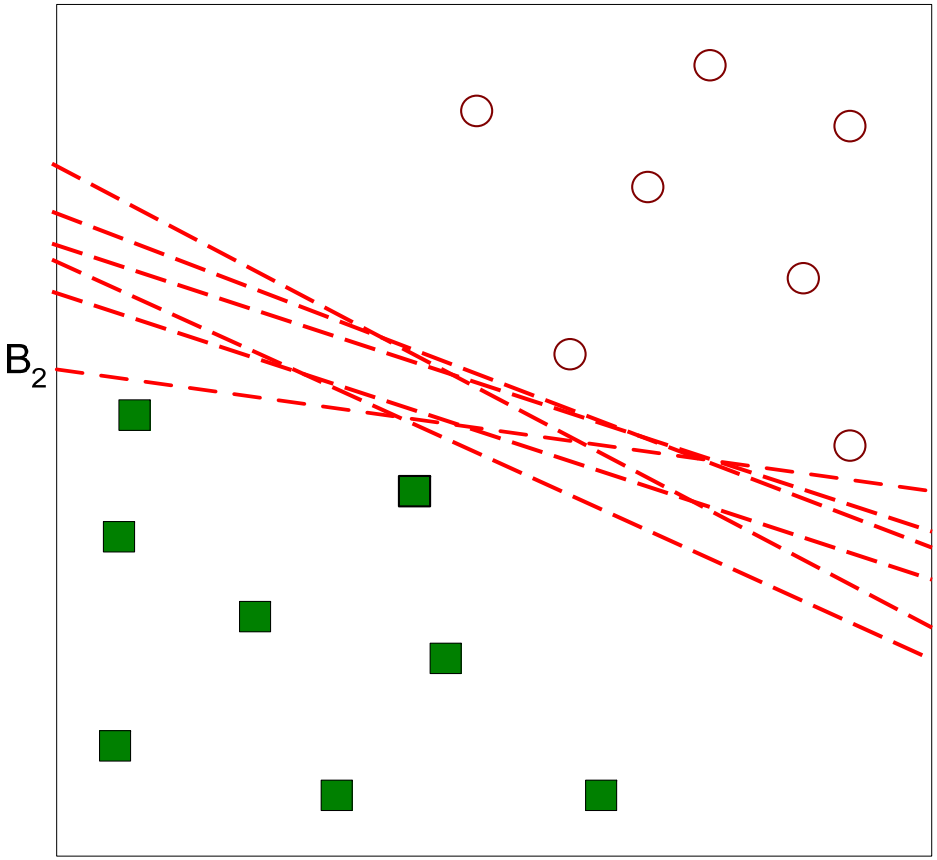
Example: In case of k=1; n=2, then we have

$$f(x) = w_{11}x_1 + w_{12}x_2$$

Linear separability



Linear separability



Note that the function learned may not be unique (to a given learning problem) as several planes may separate the classes in a training set equally well.

Linear separability

Problem: The hyperplanes always pass through zero. For example, the line defined by:

$$f(x) = w_{11}x_1 + w_{12}x_2$$

Passes through zero. It would be unable to separate two classes which are both located on on one side of the coordinate system.

Solution: Every neuron in an MLP receives an additional weighted input called **bias**. The bias provides a constant input which is weighted. The bias allows us to “move” the hyperplane freely across the output space. Thus:

$$o_i = \sum_{j=1}^{n} w_{ji} x_j + b_i$$

Linear separability

Thus, a single layer MLP can only solve problems which are linearly separable. Formally, a single layer MLP can solve problems for which there exists an ε such that:

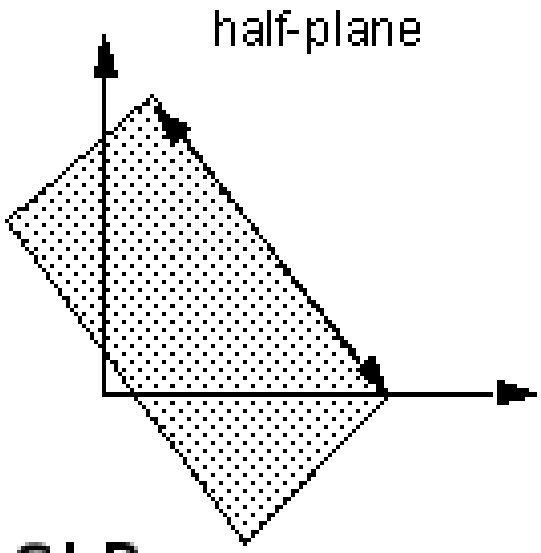
$$\boldsymbol{y}_i \cdot (< \boldsymbol{w}, \boldsymbol{x}_i > + \boldsymbol{b}) > \varepsilon$$

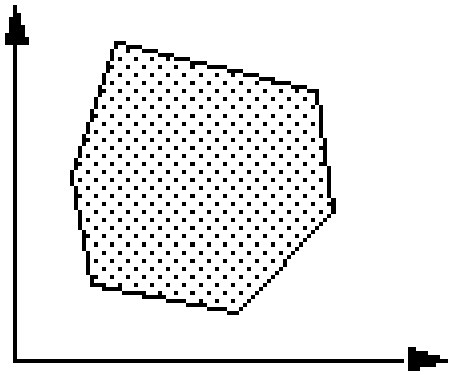
, where $\boldsymbol{b}$ is the bias, and $\boldsymbol{x}_i$,$\boldsymbol{y}_i$ refer to the i-th input/output pair.

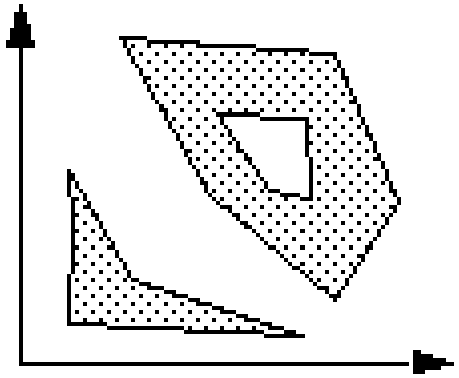
There are many problems which are not linearly separable. For example, the XOR problem can not be solved since only one hyperplane per unit can be produced to the separate two classes .

MLPs with two or more hidden layers can solve such problems because the function (the MLP) can realize non-linear **decision surfaces**.

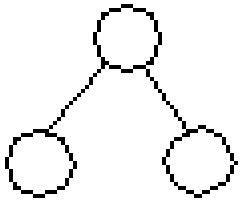
Computational power



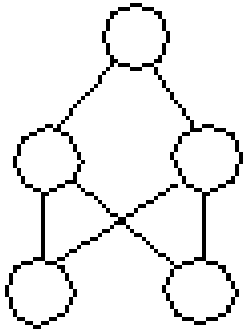


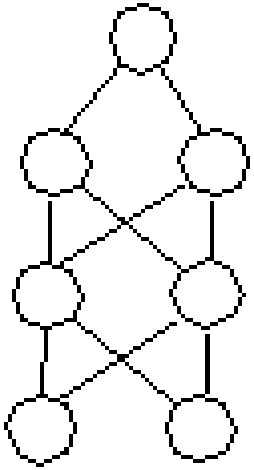


SLP



MLP





Computational power

Online simulations of MLP are available which demonstrate the ability of a MLP:

http://www.nullpointer.ch/machine-learning/supervised-learning/MLP/

http://freeisms.com/MLPAppletItself.html

Also supported in WEKA:

http://www.cs.waikato.ac.nz/ml/weka/

Example: The Iris data set: onsists of 50 samples from each of three species of *Iris* flowers (the target classes). Four features were measured from each sample, they are the length and the width of sepal and petal.

Computational power

We have seen that the MLP learns a parameterized function, and that the number of parameters affect the capability of the MLP.
In general, the more parameters we have, the better the (training) performance of the MLP at the cost of more computation time.
However, there is a limit to how many parameters an MLP can have. The limit is caused by the problem domain.
For example, an MLP capable of solving the XOR problem is limited to having a 2-dimensional input, and a 1-dimensional output. This causes a 2-layer MLP to have at most 8 weights/parameters (plus the biases). While this suffices to solve the XOR problem, there are learning problems where this restriction causes a limitation in the computational abilities.
Solution: Add more hidden layers, add direct forward links, or binarization of targets.

MLP, alternate notation

For the following, it is more convenient to use a matrix notation as follows:

$$\boldsymbol{o} = F_p(C\boldsymbol{y})$$
$$\boldsymbol{y} = F_n(A\boldsymbol{x}),$$

where $\boldsymbol{x}$ is an input vector, $\boldsymbol{y}$ is the output of the hidden layer, $\boldsymbol{o}$ is the output of the network, A is a matrix of weights connecting the input with the hidden layer, C is a matrix of weights connecting the hidden with the output layer, and F_n is an n-dimensional vector with all elements set to $f(\cdot)$.

Elman Network

An MLP can only deal with fixed sized vectors. But there are many problem domains for which the input vectors are not fix in size. For example, time series information such as in speech processing, financial forecasting, text processing, and many more.

Simply put, an Elman Network is an extension of the MLP which is trained by using a shifting (fixed sized) window over the input sequence. This window serves as an input to the network. In addition, the output of the hidden layer is forwarded to the input layer when processing the next time window.

Elman Network

Formally, an Elman Network can be expressed as follows:

$$\boldsymbol{o} = F_p(C\boldsymbol{y})$$
$$\boldsymbol{y} = F_n(A\boldsymbol{x} + Bq^{-1}\boldsymbol{y}),$$

where y^{-1} is a shift operator such that $q^{-1}\mathbf{y}$ denotes the availability of the hidden layer output **y** from the previous iteration. In other words, it denotes the relationship of the current input with the previous input.

Elman Network

The Elman Network processes an input sequence iteratively. The same gradient error descent method can be applied to train the system. A difference is that the gradient needs to be propagated back iteratively. The longer the sequence, the longer the path for which the gradient needs to be propagated back. Frasconi, Baldi proved that this can lead to **long term dependency** problems. This remains a largely unsolved problem.

The learning algorithm is gradient based as for MLP. But since the gradient needs to be passed back through the sequence, and hence, the algorithm is also known as BPTT.

Elman Network

$q^{-1}\mathbf{y}$ was the network's internal response to the processing of the previous time window. In other words, $q^{-1}\mathbf{y}$ refers to the state of the network from a previous time instance. Hence, we will refer to this simply as state.
The Elman Network can only deal with directed sequences. There are problem domains which deal with more complex structures such as in document processing and document parsing, robot navigation, chinese character recognition, etc. In these cases, the data is represented as a tree data structure.

Observation: A tree data structure with a maximum out-degree of 1 is equivalent to a directed sequence. Hence, directed sequences are a special case of trees.

Back Propagation Through Structure

A subsequent extension of Elman's idea to tree structured data is possible if the maximum out-degree *c* is known. In this case

Formally, BPTS can be described as follows:

$$\boldsymbol{o} = F_p(C\boldsymbol{y})$$
$$\boldsymbol{y} = F_n(A\boldsymbol{x} + \boldsymbol{B}q^{-1}\boldsymbol{z}),$$

Where $\boldsymbol{x}$ refers to the numeric label attached to a node in a tree $\boldsymbol{B}q^{-1}\boldsymbol{z}$ is a shorthand notation of the following:

$$\boldsymbol{B}q^{-1}\boldsymbol{z} \overset{\Delta}{=} \begin{bmatrix} B_1 & B_2 & \dots & B_c \end{bmatrix} \begin{bmatrix} q_1^{-1}\boldsymbol{y}_1 \\ q_2^{-1}\boldsymbol{y}_2 \\ \vdots \\ q_c^{-1}\boldsymbol{y}_c \end{bmatrix}$$

Frasconi et.al. proposed this approach. It was later named after its learning algorithm BPTS.

Back Propagation Through Structure

Thus, BPTS processes nodes in a tree, one at a time, and takes as input the networks' state for all of the nodes' children. It can be seen clearly that if $c = 1$ then BPTS will be an equivalent to an Elman Network.

Since the state of all children needs to be computed first when processing a given node, this dictates the processing of nodes in the inverse topological order (from leaf-to-root nodes). This is a recursive procedure, and hence, this class of networks is referred to as **Recursive Neural Networks**.

Training the BPTS

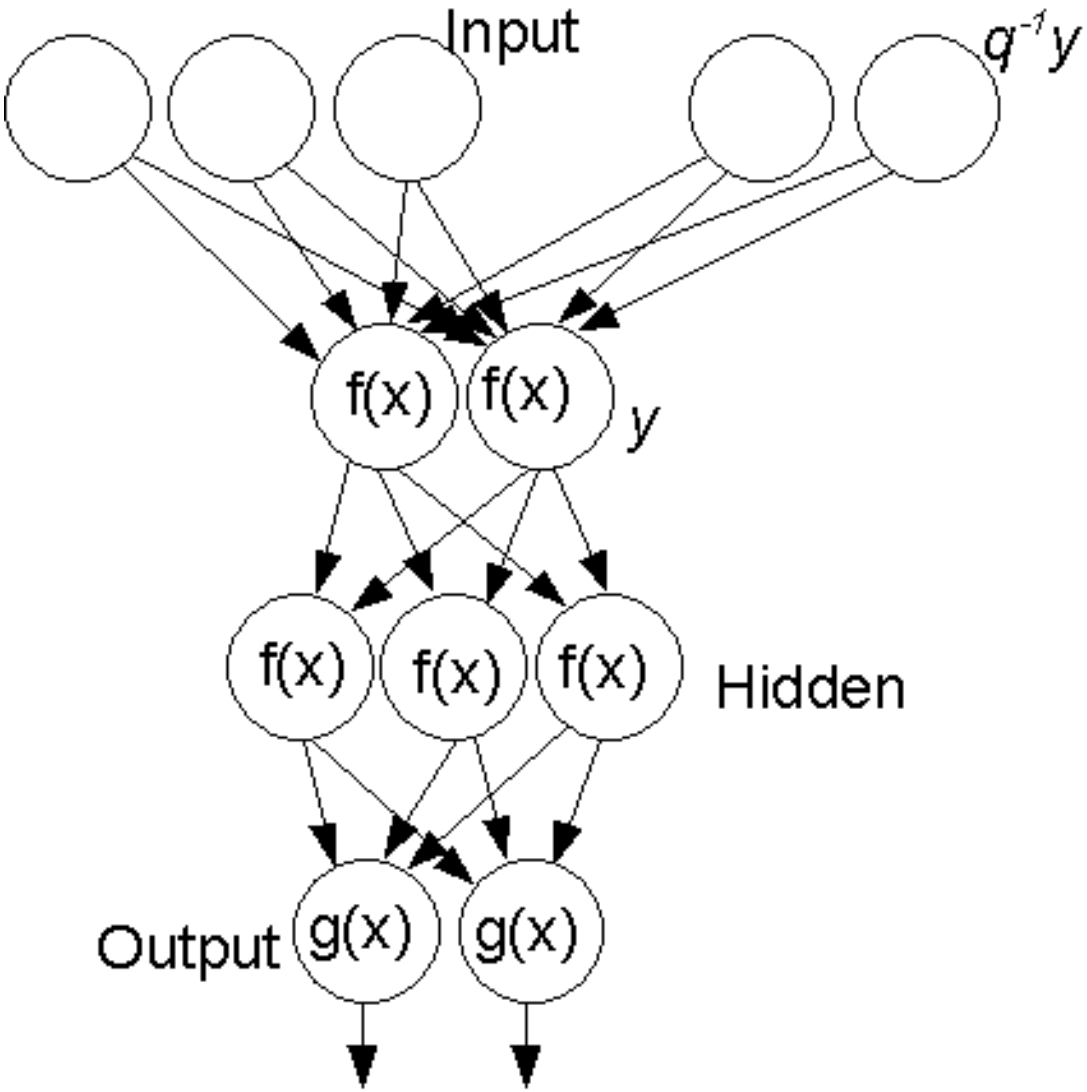
The principle of minimizing the cost function by following the gradient descent method remains unchanged. However, the gradient needs to be propagated back recursively through the same network again and again. This lead to the formation of the term **unfolding architecture**.

BPTS

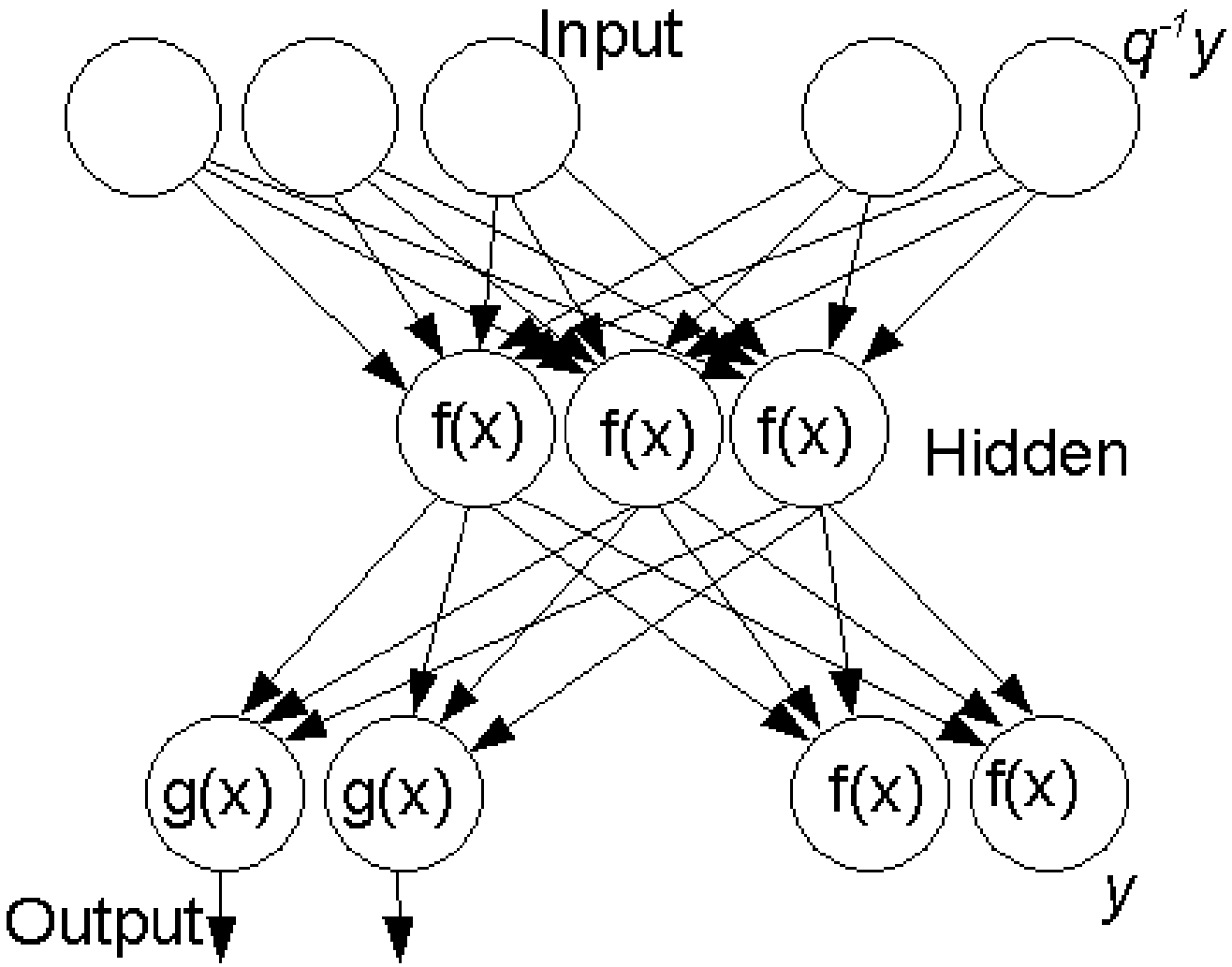
There are several variants of BPTS. For example:

- The output MLP architecture
- The state MLP architecture

The state MLP architecture



The output MLP architecture



Some Limitations of the BPTS

•Can only encode causal relationships.

•Can “only” deal with tree data structures. But what about problems involving chemical molecules, image processing applications, the world wide web, etc. are a few examples of problem domains which are represented by cyclic directed or undirected graphs.

•The BPTS can not deal with cyclic dependencies between
•nodes in a graph.

Summary

Supervised learning can be used for Data Mining tasks

- If the problem requires the **classification** of data
- If the problem can be described by a function.
- If some **ground truth** information is **available**
- If an extraction of rules is not required

A main advantage of machine learning is that expert knowledge is normally not required.

MLP provides a generic (one-size-fits-all) algorithm to classification.

Note that an MLP is hardware which is almost always simulated by software realizations.