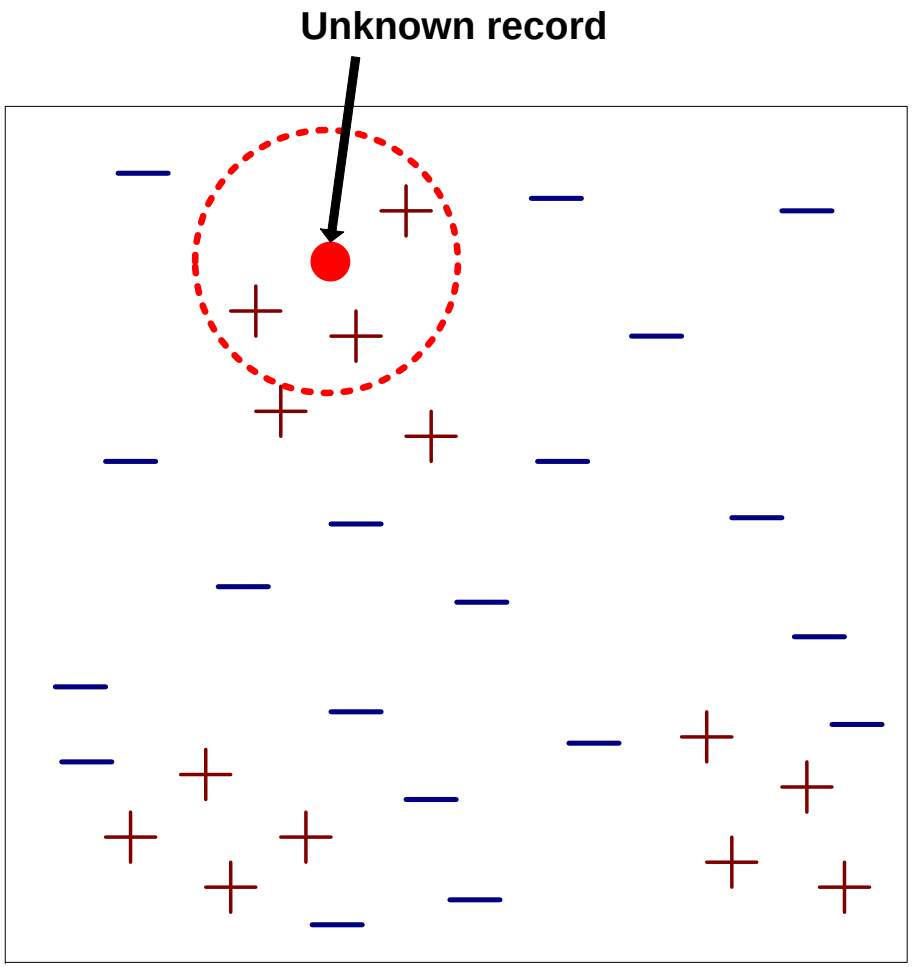
Introduction to Data Mining

Data Mining
Classification: Alternative Techniques

Recap: Nearest-Neighbor Classifiers



Main issues:

- Scalability issue

  Each unknown record must be compared to all of the known records.

  Distances need to be sorted from closest to furthest.

- How to chose k?

  If k is too small then we become sensitive to noise. If k is too large then we may get the wrong class. We can not know a-priory the best value for k.

Learning Vector Quantization (LVQ)

LVQ addresses the shortcomings of the k-nearest neighbor.

- Given n input/target pairs $\{\mathbf{x}_1, \mathbf{c}_1\}, \{\mathbf{x}_2, \mathbf{c}_2\}, \ldots, \{\mathbf{x}_n, \mathbf{c}_n\}$
- Initialize k-codebook vectors $\mathbf{m}_1, \ldots, \mathbf{m}_k$ (by randomly choosing k input/target samples)
- Repeat for a pre-set number of iterations:
  - Select an input/target pair $\{\mathbf{x}_i, \mathbf{c}_i\}$
  - Compute best matching codebook $s = \arg\min_k \left\| x_i - m^k \right\|$
  - Update the winning codebook

$$m^s(t+1) := \begin{cases} m^s(t) + \alpha(t)[x_i - m^s(t)] & \text{if class of } m^s \text{ same as class of } x \\ m^s(t) - \alpha(t)[x_i - m^s(t)] & \text{else} \end{cases}$$

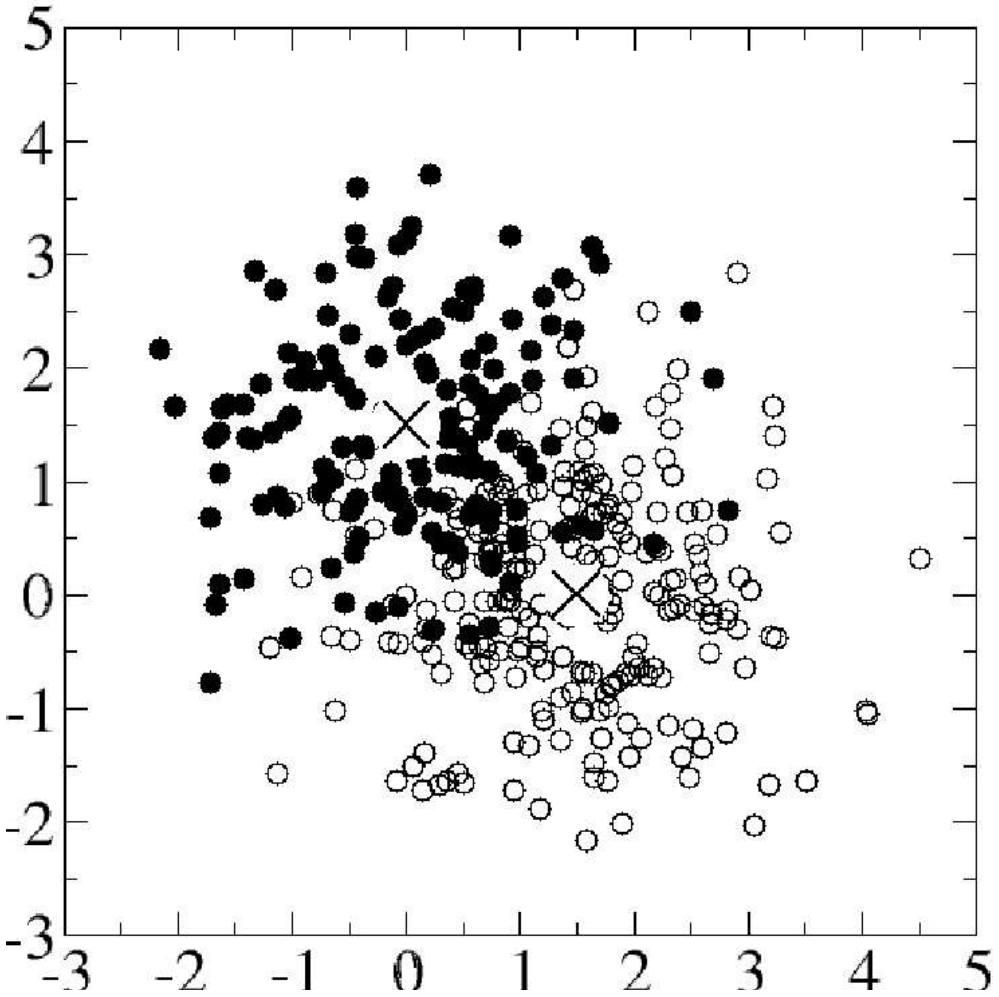
Learning Vector Quantization (LVQ)

Thus, LVQ trains a set of prototype vectors.

These prototypes are then a representation of the various classes of a given dataset.

This reduces the dimensionality of the problem when obtaining a class membership of a given test pattern.

Learning Vector Quantization (LVQ)



Example: Two prototype vectors (the crosses) in a two dimensional space defined by data belonging to two different classes.

To find the class membership of a new pattern, we simply compute the nearest prototype vector.

Learning Vector Quantization (LVQ)

The before mentioned algorithm is known as LVQ1. There are refinements which aim at enhancing the accuracy of the system. These are named LVQ2.1 and LVQ3.

LVQ is a simple **machine learning** approach to achieve **classification**. LVQ is said to engage a **competitive** , winner takes all, learning scheme.

Later we will introduce another competitive learning scheme called Self-Organizing Maps (in chapter on clustering)

Machine Learning

Machine Learning is a science concerned with the development of algorithms that can learn from data.

Commonly used machine learning algorithms are:

1. LVQ
2. K-means
3. Self-Organizing Maps
4. Support Vector Machines.
5. Multi-layer perceptron networks.

…and many others. The before mentioned algorithms are suitable for many Data Mining Exercises.

Artificial Neural Networks (ANN)

Artificial Neural Networks:

- Learn from data
- Simulate biological counterpart (the brain)
- Are massively parallel systems
- Tolerant to faults (within the system) and noise
- NN consist of Neurons and Axons, and Synapses
- ANN consist of Neurons and Weights

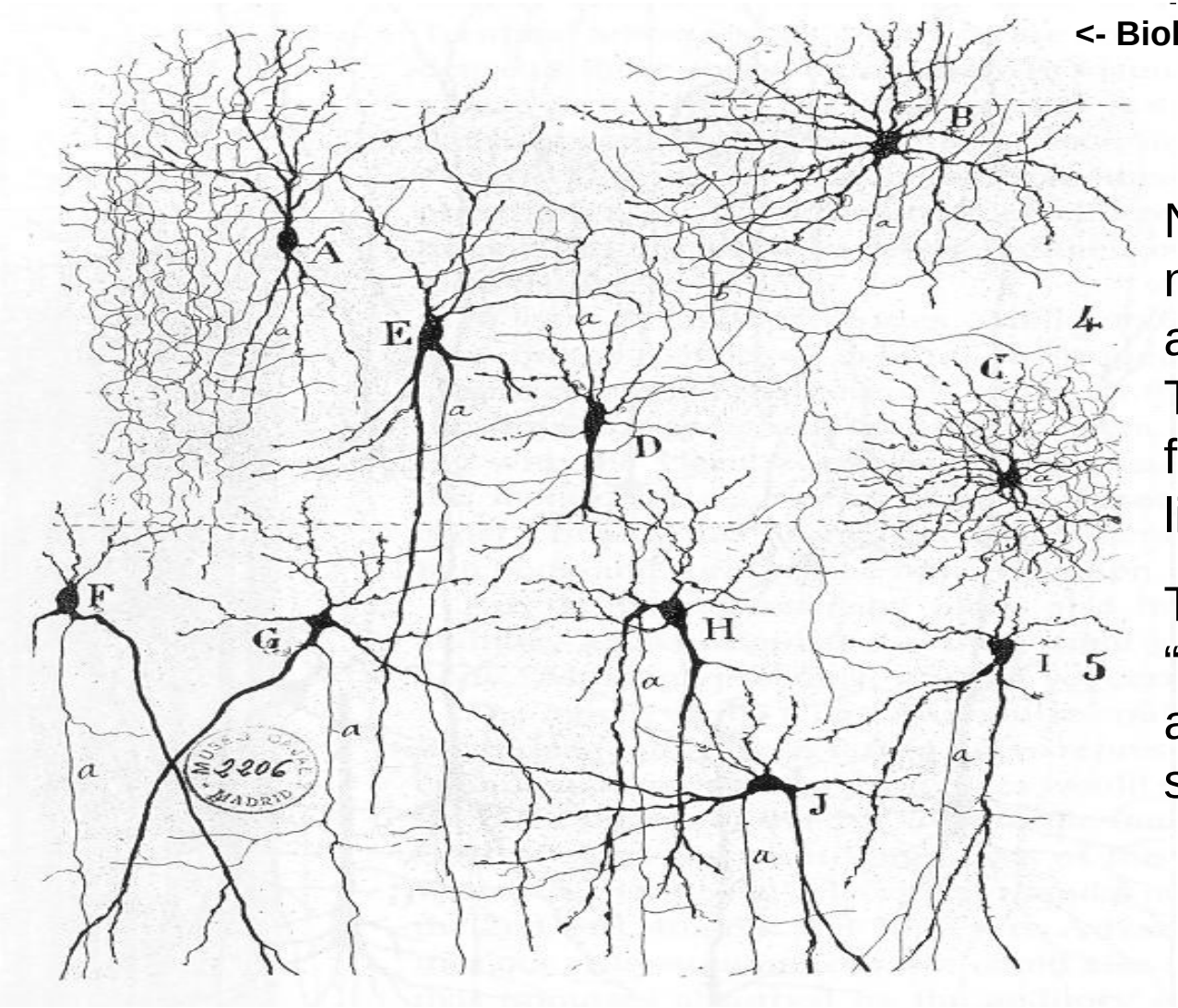
Artificial Neural Networks

An example: The human brain consists of a large number of **neurons**. These neurons allow us to "learn", "understand", and "memorize". But how do the neurons achieve this?

Your task: Watch the video presentation, then answer the following questions:

* How many neurons does the human brain have?
* How many connections does each neuron have?
* How do neurons communicate?
* How are connections between neurons formed?

Artificial Neural Networks



<- Biological Neural Network

Neurons consist of a nucleus (cell body), an axon, and dentrites. The axon and dedrites form communication lines between neurons.

The nucleus is a “threshold unit”. It fires a spike if excited sufficiently.

Artificial Neural Networks

Observation: The human brain can learn by establishing suitable communication lines between neurons. Hence, it is not the neurons that cause “intelligence” it is the connections between the neurons. There are about 5,000,000,000,000 connections in an adult brain!

-> Too many to simulate on a computer.

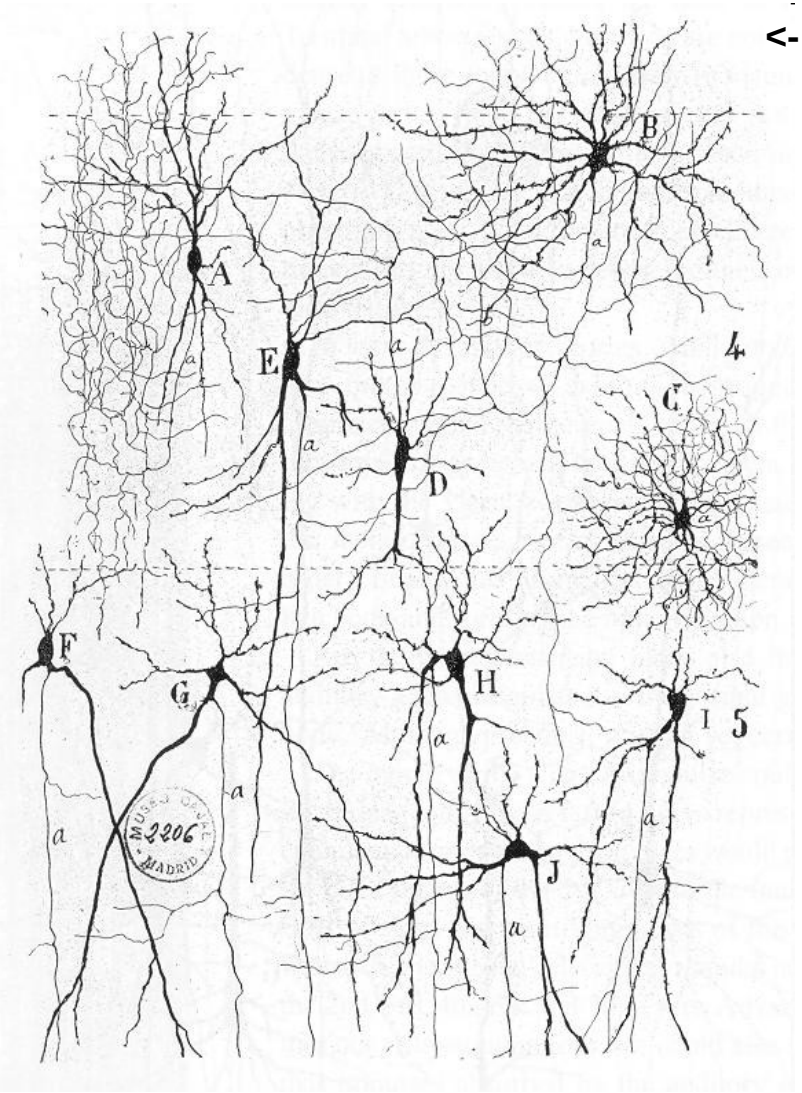
But we can simulate a smaller number of neurons and their connections in an attempt to achieve the ability for a machine to “learn”, and to “memorize” basic tasks.

Neural Networks for Mining Data

Time

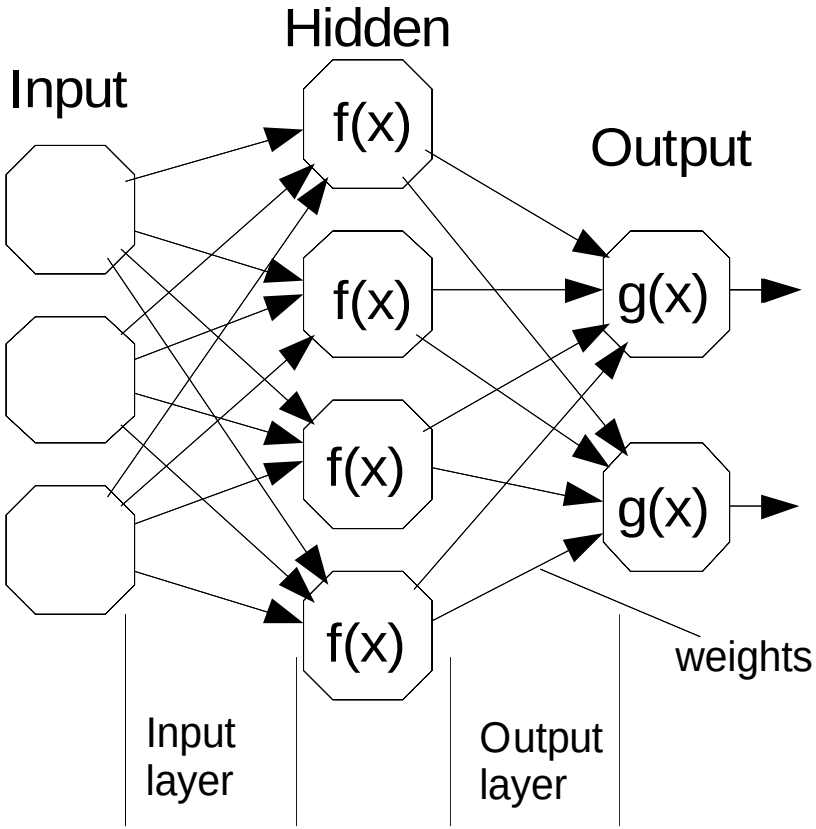
- 2009 Unsupervised learning of graphs (Hagenbuchner et.al.)
- 2008 Supervised learning of graphs (Hagenbuchner et.al.)
- 2002 Graph-matching editSOM (Bunke et.al.)
- 1999 Unsupervised learning of tree-data(Hagenbuchner et.al.)
- 1996 Supervised learning of tree-data (Frasconi et.al.)
- 1989 Supervised learning of sequences (Elman)
- 1986 Supervised learning of fixed vectors (PDP).
- 1986 Un-Supervised learning of fixed vectors (T.Kohonen)
- 1982 Un-Supervised learning of fixed vectors (Hopfield)
- The Minsky and Papert (1969) hole
- 1957 Supervised learning of fixed vectors (Rosenfeld)

Artificial Neural Networks



<- Biological Neural Network

Artificial Neural Network



Artificial Neural Networks

There are two forms of Neural Networks

- Supervised (target values are available for some data from the domain)
  - Classification
  - Regression
  - Projection
- Unsupervised (target values are not available)
  - Clustering
  - Projection (dimension reduction)

There is a special class of NN which is both, supervised and unsupervised: The Auto-associative memory.

Artificial Neural Networks

Thus these models cover the following ability of the human brain:

- To learn with the help of a teacher.
- To organize itself into dedicated regions.
- To memorize.

One of the easiest ANNs is the Hopfield model. It can memorize data in a fault tolerant fashion.

(The Hofield model is a whiteboard exercise; not part of an assessment item for this subject)

Multi-layer Perceptron Networks

- An MLP consist of neurons (called perceptrons) which are organized in layers.
- The layers are fully connected by weighted connections (the weights).
- MLP are trained in a supervised fashion in two phases:
  1. Forward phase
  2. Backward phase

Multi-layer Perceptron Networks

The neuron:

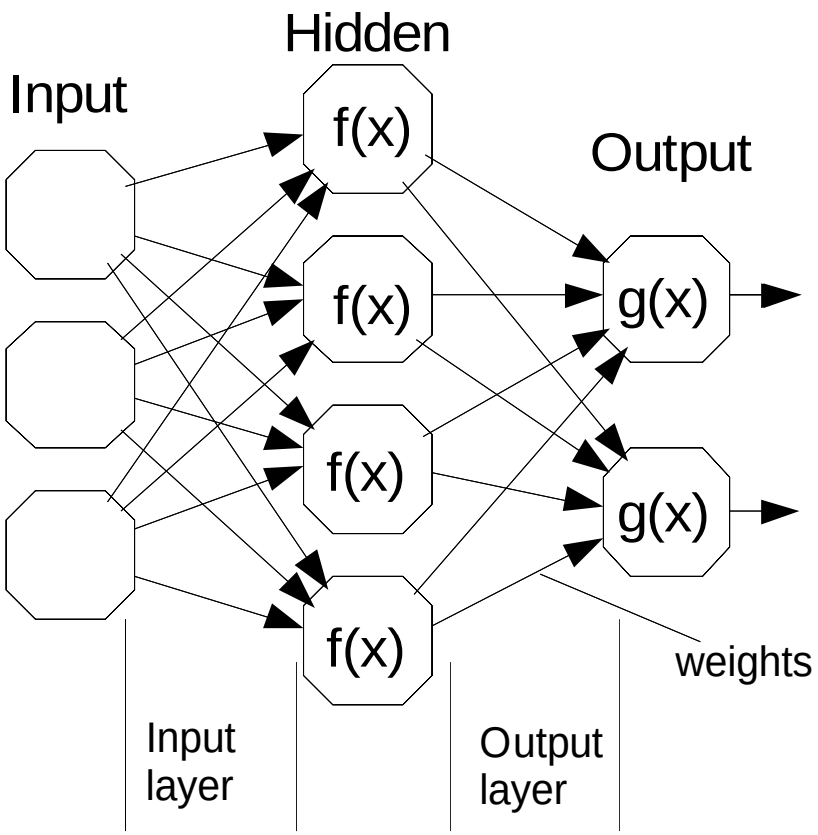
- Is connected to by other neurons. The connections are weighted.
- Computes a weighted sum of all inputs, then produces a response.
- Uses an “activation” function to compute the response. Common activation functions are:

Linear: f(x):=x

Non-linear: *f(x):=tanh(x) or*

$$f(x):=1/(1+e^{-x})$$

Artificial Neural Networks



A layered neural network with 6 neurons organized in two layers. The layers are fully connected.

Multi-layer Perceptron Networks

Forward phase:

For each neuron, starting with the input layer, and working towards the output layer, compute its output:

$$x_j^n = f\left(\sum_{i=0}^{m} w_{ji} x_i^{n-1}\right)$$

,where f(.) is said to be the transfer function (or activation function), w_{ij} is the weight connecting the current neuron with the i-th neuron, and x_i^{n-1} is the output of the i-th neuron in layer n-1.

The output at the output layer is the output of the network.

Multi-layer Perceptron Networks

Backward phase:

The output of the network is compared with a given target value. The error is computed by using the Euclidian.

The network is trained with the aim to minimize a “Cost” function. For example,

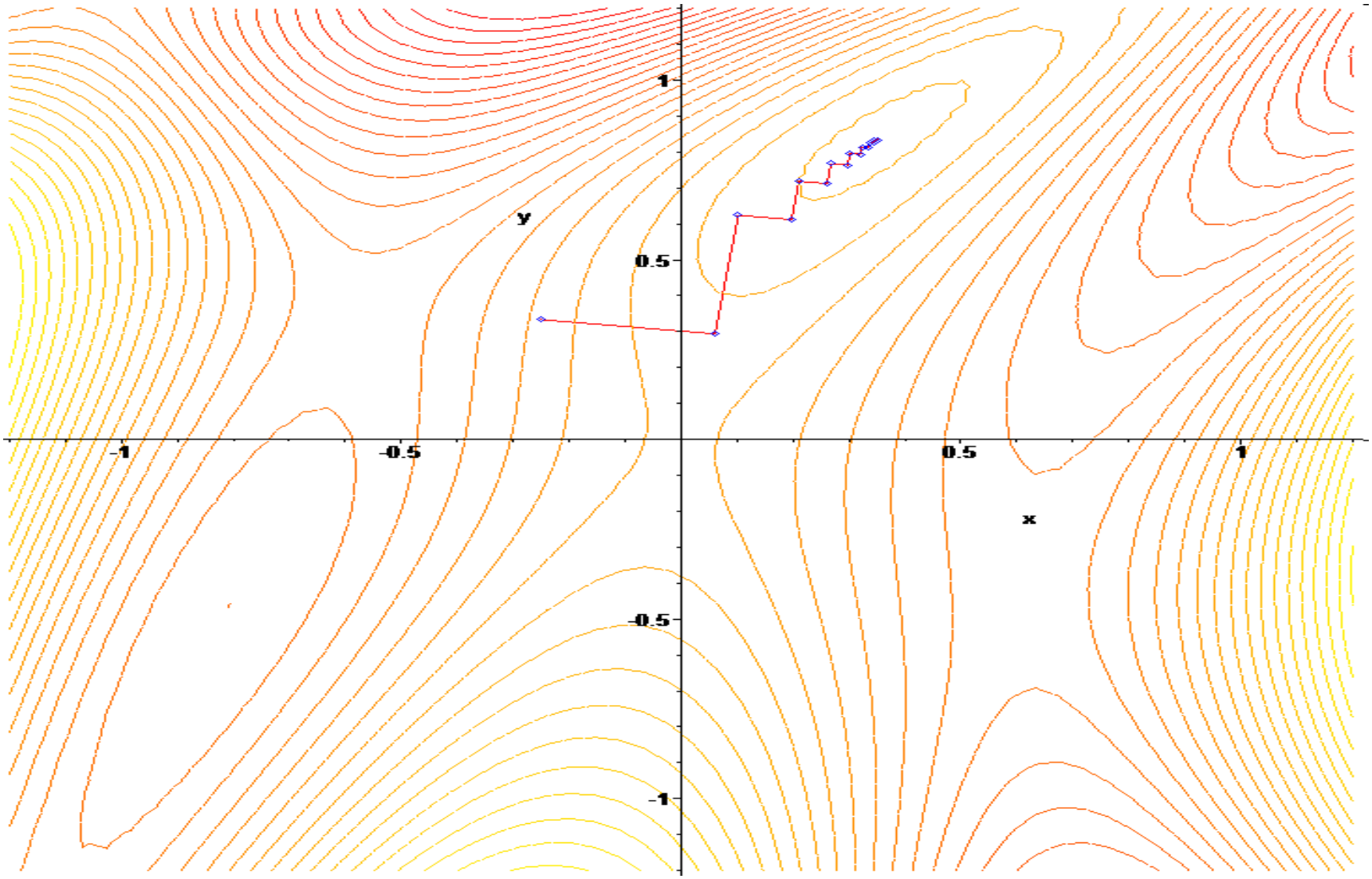
$$E = \frac{1}{2}\|x - t\|$$

, where t is a given target.

This can be done by using a *gradient descent* method:

For each weight in the network, compute the derivative of function E (with respect to this weight), then change the size of this weight into the negative direction of the gradient.

Gradient descent learning



Multi-layer Perceptron Networks

The MLP is a parameterized model. The weight in this model can be adjusted.

The purpose of the Error Back-propagation Algorithm is to minimize the error of a given cost function by adjusting the weights of the MLP.

We do not know a-priori the amount by which each weight needs to be changed in order to minimize the error. But we can know the direction in which each weight needs to be changed so as to reduce the error.

The direction of a weight change is computed by using the sign of the gradient of the cost function with respect to each of the weights.

Multi-layer Perceptron Networks

In practice:

- The output neurons almost always use a linear activation function.
- Hidden neurons always use a non-linear activation function.
- An MLP can consist of many layers though two- and three layer architectures are the most common.

Multi-layer Perceptron Networks

Computing the weight change: At the output layer

$$\Delta w_{ij} = \alpha \delta_i x_j$$

, where $0 < \alpha < 1$ is a learning rate, and δ_i is the network error at (output) neuron *i*. Note that a weight change is computed. The weights are not changed immediately.

This is obtained by computing

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\frac{1}{2} \sum_k \left(x_k - t_k \right)^2 \right)$$

Multi-layer Perceptron Networks

Computing the weight change: At the hidden layer

$$\Delta w_{ij} = \alpha \delta_i x_j$$

, where α is a learning rate, and $\delta_i = f'(x) \sum_j w_{ji} \delta_j$

is the network error propagated back through the previous (output) layer. Hence the name error-back propagation algorithm.

This is also obtained by computing

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta}{\delta w_{ij}} \left(\frac{1}{2} \sum_k \left(x_k - t_k \right)^2 \right)$$

Multi-layer Perceptron Networks

The training algorithm:

Initialize the weights in the network with random values

Repeat

For each example $\boldsymbol{x}$ in the training set

$\boldsymbol{x}_o$ = neural-net-output(network, x) ; (forward pass)

$\boldsymbol{e}$ = Calculate error ($\boldsymbol{t}$ - $\boldsymbol{x}_o$).

Compute Δw for all weights in the hidden layer and output layer by using the gradient descent method;

Update the weights in the network

Until network error does no longer decrease.

Multi-layer Perceptron Networks

The forward phase and backward phase are repeated for a number of iterations until the network error does no longer decrease.

A trained network is said to be a “model” which “encodes” the problem domain.

The model can then be applied to unseen data. The network will then produce an output in accordance to the input pattern.

This requires only one pass through the forward phase.

-> Linear computational complexity

However, the MLP is a “black box”: We do not now (nor understand) how the model has encoded the information (there is no rule which we can extract).

Multi-layer Perceptron Networks

Limitations of MLP:

- Asymptotic algorithm; error reduction slows down with the approach to a minimum.
- Converges to a local minimum which is not necessarily the global minimum.
- Requires setting of network parameters (number of hidden layers, number of neurons in each layer, initialization of weights)

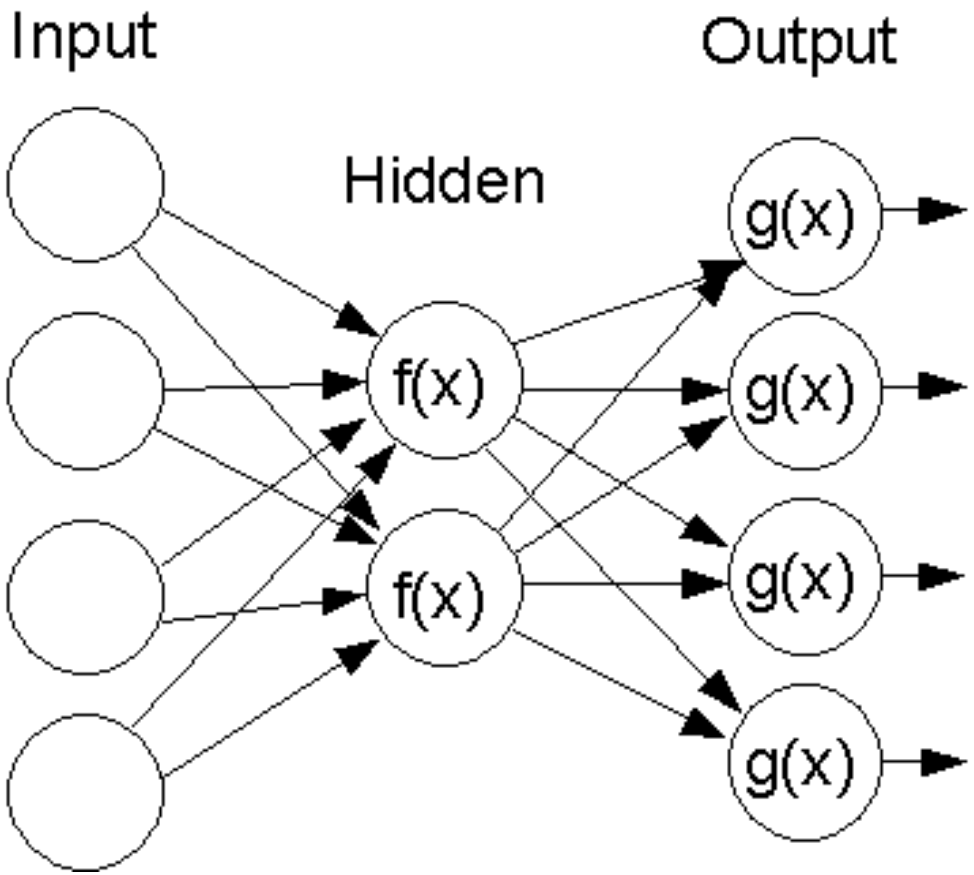
These limitations can be addresses by:

- Using a momentum term or adaptive learning rates.
- Careful network initialization (trial and error is OK)
- Use of Cascade correlation networks.

Auto-association

A special instance of ANN realizes auto-association.

Consider the following architecture:



What will happen if such an architecture is trained on data whose target is the same as the input?

Auto-association

In fact, an AAM realized by an MLP is seen as an equivalent to the Support Vector Machine approach. Both can be used to help reduce the input dimension of a given learning problem by computing a linear encoding of input features such that the error in re-constructing is minimal.

Both, MLP and SVM are proven to perform such a task optimally.

Multi-layer Perceptron Networks

- MLPs are widely used for classification and regression problems.
- Are proven to solve **any** continuously differentiable function to **any** precision.
- Can be used as a convenient way to perform dimension reduction (by using an associative memory architecture)
- Have some limitations for “knowledge discovery” tasks (they are trained supervised, and hence, some a-priory knowledge is required).
- Unsupervised machine learning methods are more suitable for knowledge discovery tasks. For example: Self-Organizing Maps.

Multi-layer Perceptron Networks

- MLPs can deal with vectorial information.
- What about more complex data structures such as sequences (i.e. time series information), trees (i.e. parse trees), and graphs (i.e. chemical molecules)?
- A classic approach for dealing with complex structures is to squash those into a vectorial form before feeding them to an MLP.
- Next week: Classification of complex data structures