Example: Using Backpropagation algorithm to train a two layer MLP for XOR problem.

Input vector $\mathbf{x}_n$	Desired response t_n
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	0

The two layer network has one output

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} h\Big(w_j^{(2)} h\Big(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\Big)\Big)$$

where $M = D = 2$. The output activation function and the hidden units $h(a)$ have sigmoidal activation function given by

$$h(a) = \frac{1}{1 + \exp(-5a)} \tag{1}$$

A useful feature of this function is that its derivative can be expressed in a simple form

$$h'(a) = 5h(a)(1 - h(a)) \tag{2}$$

The summary of the algorithm

- Outer loop (training epoch)
  - Inner loop (data samples)
  - Backpropagation to find $\nabla E_n(\mathbf{w}^{(\tau)})$.
  - Update weight vector

    $$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

    where η is preset learning rate.
  - End inner loop, until the last data sample.

- End outer loop, until a predetermined number of training epoches has reached.

We detail the Backpropagation step as below.

For each training pattern in the training set, we first perform a forward propagation using

$$a_j^{(1)} = \sum_{i=0}^{2} w_{ji}^{(1)} x_i$$

$$z_j = h(a_j^{(1)}) = \frac{1}{1 + \exp(-5a_j^{(1)})}$$

$$a_j^{(2)} = \sum_{j=0}^{2} w_j^{(2)} z_j$$

$$y = h(a_j^{(2)}) = \frac{1}{1 + \exp(-5a_j^{(2)})}$$

Next we compute the δ for each output unit using

$$\delta = y - t. \tag{3}$$

Then we backpropagate to obtain $\delta' s$ for the hidden units using

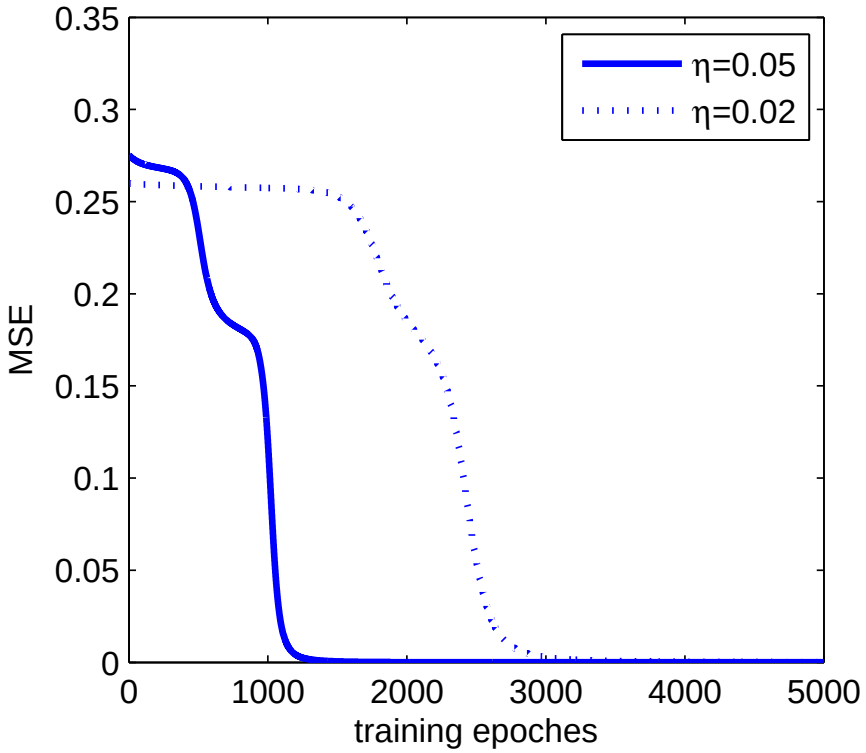
$$\delta_j = 5z_j(1 - z_j)w_j^{(2)}\delta. \tag{4}$$

Finally the derivatives with respect to the first layer and second layer weights are given by

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

$$\frac{\partial E_n}{\partial w_j^{(2)}} = \delta z_i$$

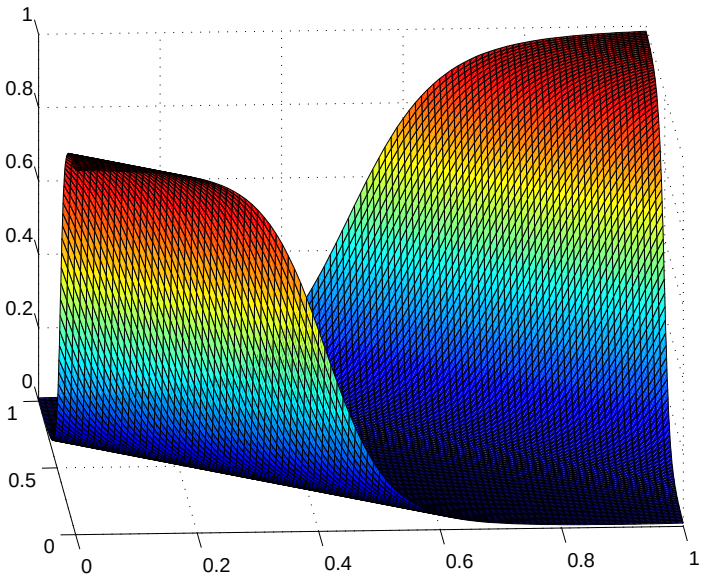
The mean square error (MSE) for $\eta = 0.02$ and $\eta = 0.05$ were shown below.



For the final model, the weights vectors converge to

First hidden node	1.5136	-1.0179	-1.0182
Second hidden node	0.5684	-1.3677	-1.3732
Output node	0.9968	-2.1416	2.2291

Input vector $\mathbf{x}_n$	Desired response t_n	Network output y_n
(0, 0)	0	0.9919
(0, 1)	1	0.0090
(1, 0)	1	0.0091
(1, 1)	0	0.9859



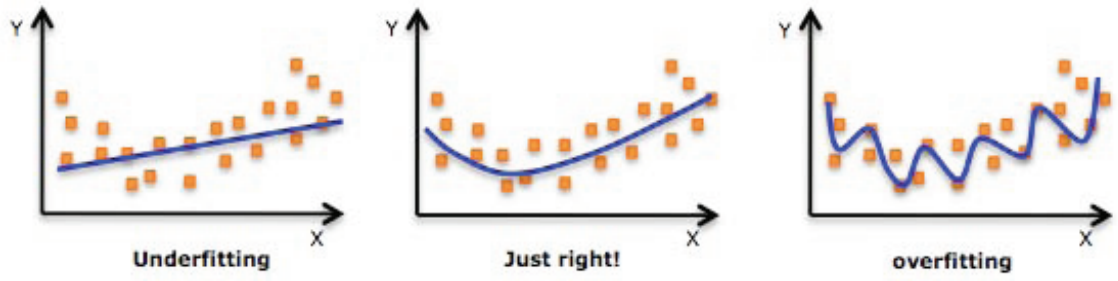
The resulting network prediction surface.

Regularization in neural networks: The number of input and output units in a neural network is generally determined by the dimensionality of the data set, where the number of hidden units is a free parameter that can be adjusted to give the better performance.

Overfitting occurs when the model or the algorithm fits the data too well, which is often a result of an excessively complicated model.

Underfitting occurs when the model or the algorithm does not fit the data well enough, often a result of an excessively simple model.

Both overfitting and underfitting lead to poor predictions on new data sets.



Generalization error is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data. So cross validation on a test data set (data that has not been used for training) can be used to choose the number of hidden units.

Alternatively regularization is a technique used to avoid overfitting for network with a large size. The simplest regularizer is to use

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} \qquad (5)$$

where $\lambda > 0$ is the regularization parameter. λ can be adjusted based on the performance of a test data set. Early stopping is a form of regularization used to avoid overfitting.