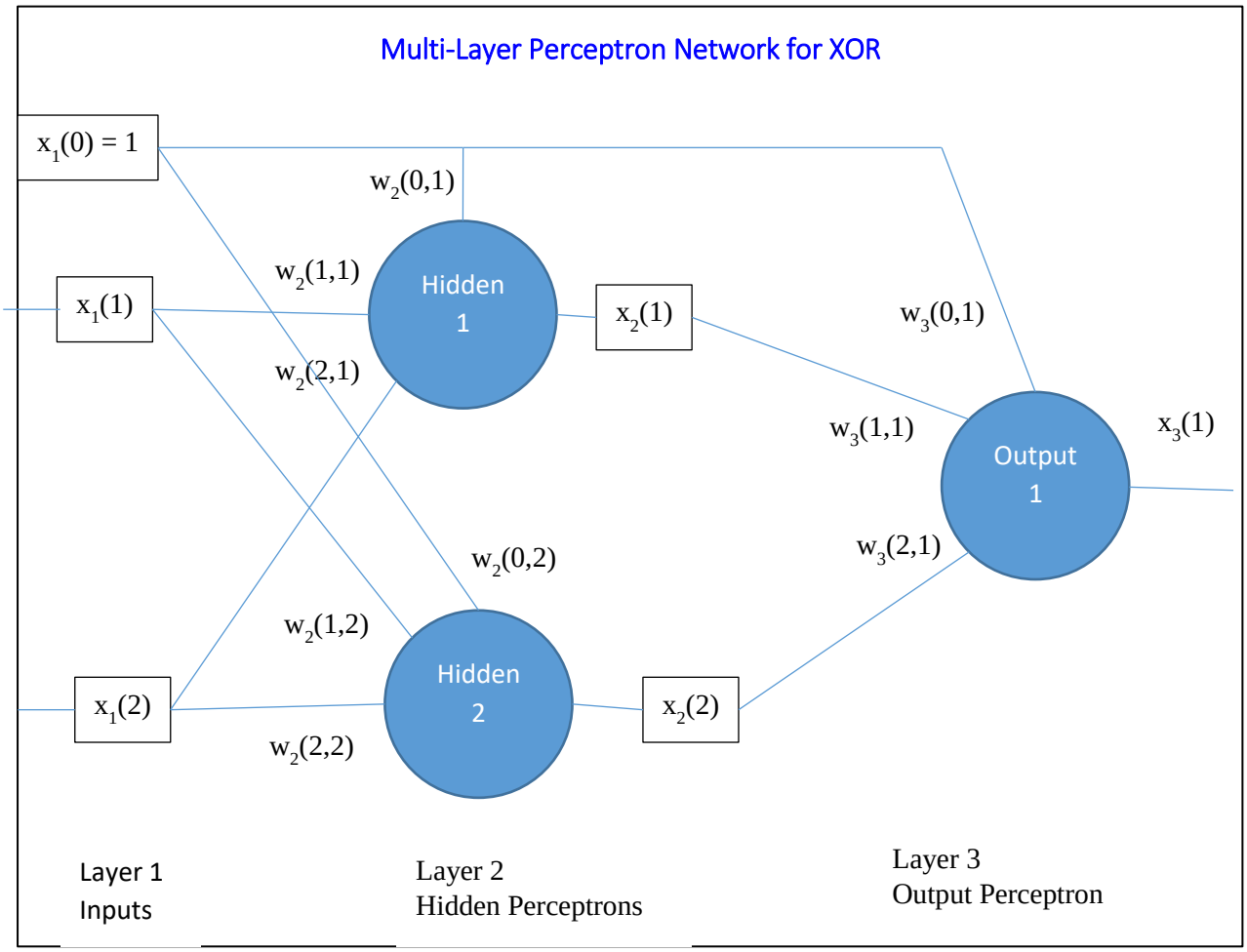
Multilayer Perceptron Notes

Example ANN for XOR and BP formulas

Remark: For each perceptron we have an extra input $x_1(0)$ which is always equals 1. The weights of this input to each of the three perceptrons are $w_2(0,1)$, $w_2(0,2)$ and $w_3(0,1)$ and they are equivalent to the threshold Θ that we used for a single percepton.



Backpropagation learning formulas are:

Assign random numbers from -1 to +1 to all 9 weights.

Calculate weighted sums for both perceptrons in hidden layer $X_2(\mathrm{j}) = \sum_{i=0}^{2} x_1(i)\, w_2(i,\mathrm{j})$

and hence the hidden outputs $x_2(j) = 1/\left(1 + e^{-X_2(j)}\right)$

Repeat for output layer with $X_3 = \sum_{j=0}^{2} x_2(j)\, w_3(j,1)$ and $x_3 = 1/\left(1 + e^{-X_3)}\right)$

Note: since there is only one neuron in the output layer, we label it simply as x_3 instead of $x_3(j)$ where j = 1, 2,3 .. n.

Weight Adjustments

Back propagate weight corrections, starting in output layer,

$\delta_3 = (d - x_3)x_3(1 - x_3) \qquad \Delta w_3(j,1) = \alpha\delta_3 x_2(j) \qquad j = 1,2$

then proceed to the hidden layer

$\delta_2(j) = x_2(j)\left(1 - x_2(j)\right)w_3(j,1)\delta_3 \qquad \Delta w_2(i,j) = \alpha\delta_2(j)x_1(i) \qquad i = 1,2 \; and \; j = 1,2$

Some Comments

The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble.

Between 2009 and 2012, the recurrent neural networks and deep feedforward neural networks developed significantly. Today, the backpropagation algorithm is the workhorse of learning in neural networks.

Multi-layer perceptron

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function.

The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds for a wide range of activation functions, e.g. for the sigmoidal functions.

Multi-layer networks use a variety of learning techniques, the most popular being back-propagation. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the network calculates the derivative of the error function with respect to the network weights (partial derivatives, multivariate calculus), and changes the weights such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions such as sigmoid but not the step function.

In general, the problem of teaching a network to perform well, even on samples that were not used as training samples, is a quite subtle issue that requires additional techniques. This is especially important for cases where only very limited numbers of training samples are available. The danger is that the network overfits the training data and fails to capture the true statistical process generating the data. Computational learning theory is concerned with training classifiers on a limited amount of data. In the context of neural networks a simple heuristic, called early stopping, often ensures that the network will generalize well to examples not in the training set.

Other typical problems of the back-propagation algorithm are the speed of convergence and the possibility of ending up in a local minimum of the error function. Today there are practical methods that make back-propagation in multi-layer perceptrons the tool of choice for many machine learning tasks.

Activation function

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then it is easily proved with linear algebra that any number of layers can be reduced to the standard two-layer input-output model (see perceptron).

What makes a multilayer perceptron different is that some neurons use a nonlinear activation function which was developed to model the frequency of action potentials, or firing, of biological neurons in the brain. This function is modelled in several ways.

The two main activation functions used in current applications are both sigmoids:

$sigmoid(X) = 1/\left(1 + e^{-X)}\right)$ and

$tanh(X)$, the hyperbolc tangent function.

Learning through backpropagation

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation, a generalization of the least mean squares algorithm in the linear perceptron.

This depends on the change in weights of the nodes in the output layer. So to change the hidden layer weights, we must first change the output layer weights according to the derivative of the activation function, and so this algorithm represents a backpropagation of the activation function.

Applications

Multilayer perceptrons (MLPs) using a backpropagation algorithm are the standard algorithm for any supervised learning pattern recognition process and the subject of ongoing research in computational neuroscience and parallel distributed processing. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to get approximate solutions for extremely complex problems like fitness approximation.

MLPs are universal function approximators as showed by Cybenko's theorem, so they can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical, MLPs are also good classifier algorithms.

MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as speech recognition, image recognition, and machine translation software, but have since the 1990s faced strong competition from the much simpler (and related) support vector machines. More recently, there has been some renewed interest in backpropagation networks due to the successes of deep learning.

Supervised Learning

Supervised learning is the machine learning task of inferring a function from labelled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples.

Regression models predict a value of the Y variable given known values of the X variables. Prediction within the range of values in the dataset used for model-fitting is known informally as interpolation. Also known as function fitting. Prediction outside this range of the data is known as extrapolation.

Function fitting is the process of training a neural network on a set of inputs in order to produce an associated set of target outputs. Once the neural network has fit the data, it forms a generalization of the input-output relationship and can be used to generate outputs for inputs it was not trained on.

This dataset can be used to demonstrate how a neural network can be trained to estimate the relationship between two sets of data.

Some Examples

Housing Dataset

Regression example. This dataset can be used to train a neural network to estimate the median house price in a neighbourhood based on neighbourhood statistics.

Description

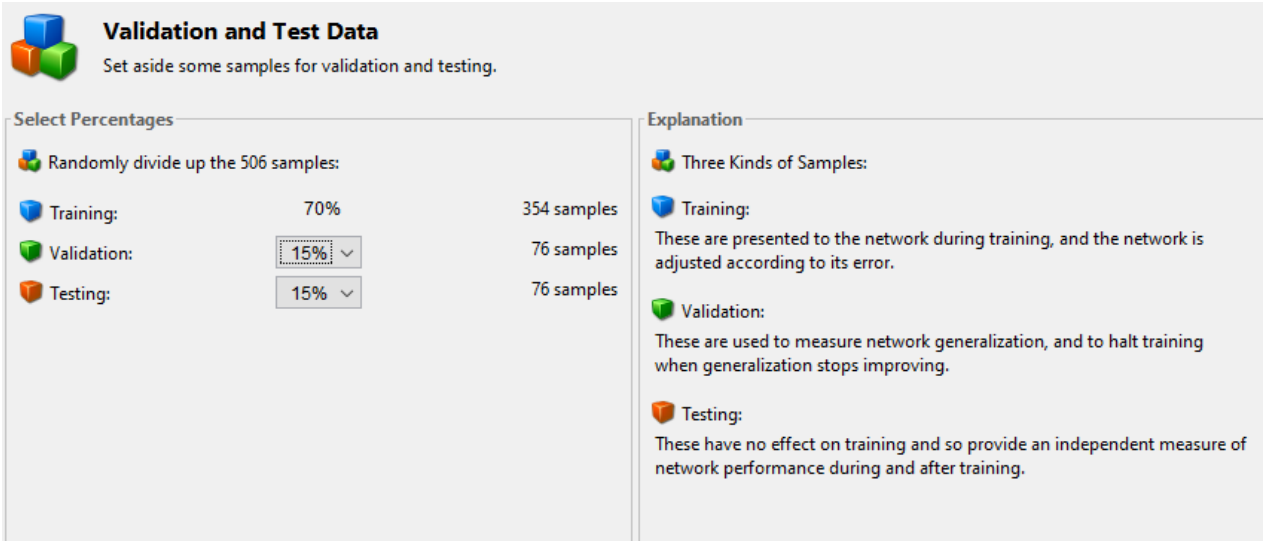
houseInputs - a 13x506 matrix defining thirteen attributes of 506 different neighborhoods.

1. Per capita crime rate per town
2. Proportion of residential land zoned for lots over 25,000 sq. ft.
3. proportion of non-retail business acres per town
4. 1 if tract bounds Charles river, 0 otherwise
5. Nitric oxides concentration (parts per 10 million)
6. Average number of rooms per dwelling
7. Proportion of owner-occupied units built prior to 1940
8. Weighted distances to five Boston employment centres
9. Index of accessibility to radial highways
10. Full-value property-tax rate per $10,000
11. Pupil-teacher ratio by town
12. 1000(Bk - 0.63)^2, where Bk is the proportion of blacks by town
13. Percent lower status of the population

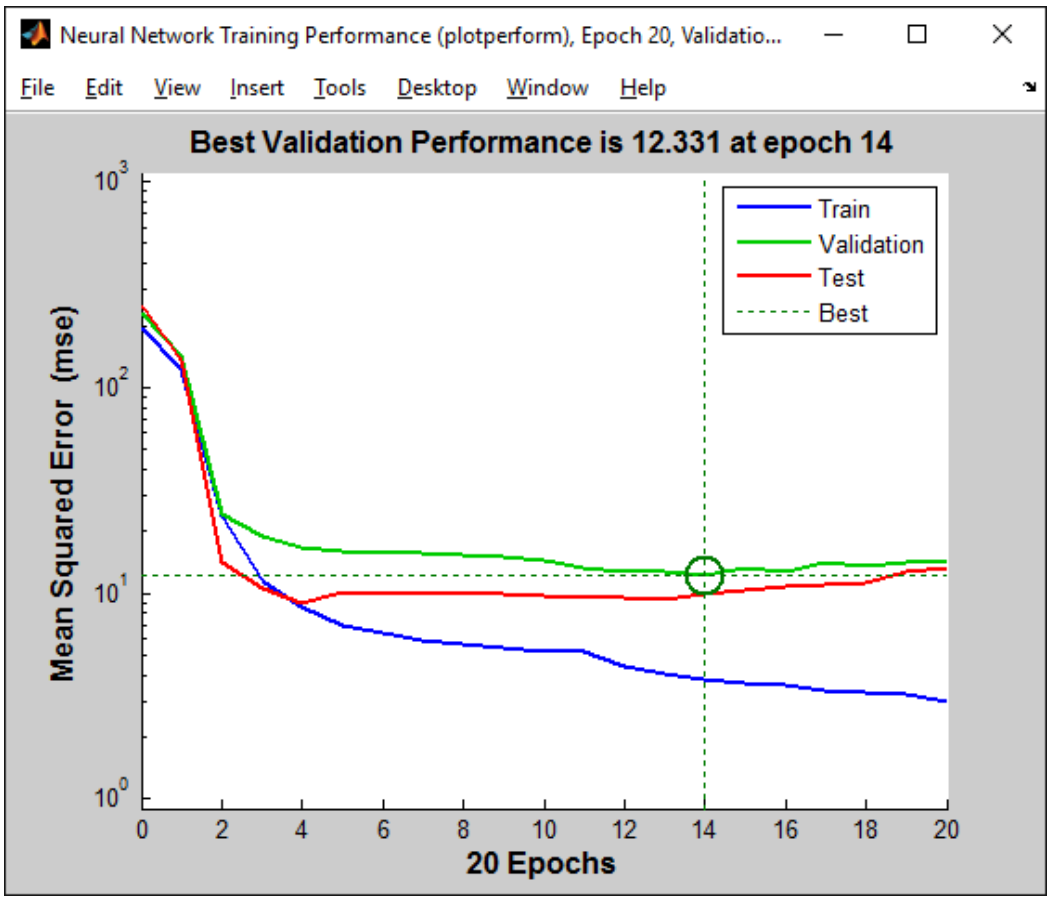
houseTargets - a 1x506 matrix of median values of owner-occupied homes in each neighborhood in 1000's of dollars.

[X,T] = house_dataset loads the inputs and targets into

Validation and Test Data in Supervised Learning



Here, learning happened quickly, 20 iteration sufficed. Weights established on 14th iteration are optimal.



Bodyfat

Regression example. This dataset can be used to train a neural network to estimate the bodyfat of someone from various measurements.

Description

bodyfatInputs - a 13x252 matrix defining thirteen attributes for 252 people.

1. Age (years)
2. Weight (lbs)
3. Height (inches)
4. Neck circumference (cm)
5. Chest circumference (cm)
6. Abdomen 2 circumference (cm)
7. Hip circumference (cm)
8. Thigh circumference (cm)
9. Knee circumference (cm)
10. Ankle circumference (cm)
11. Biceps (extended) circumference (cm)
12. Forearm circumference (cm)
13. Wrist circumference (cm)

bodyfatTargets - a 1x252 matrix of associated body fat percentages, to be estimated from the inputs.

[X,T] = bodyfat_dataset loads the inputs and targets into variables of your own choosing.

Classification (another type of supervised learning)

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

Breast Cancer Example

This dataset can be used to design a neural network that classifies cancers as either benign or malignant depending on the characteristics of sample biopsies.

Description

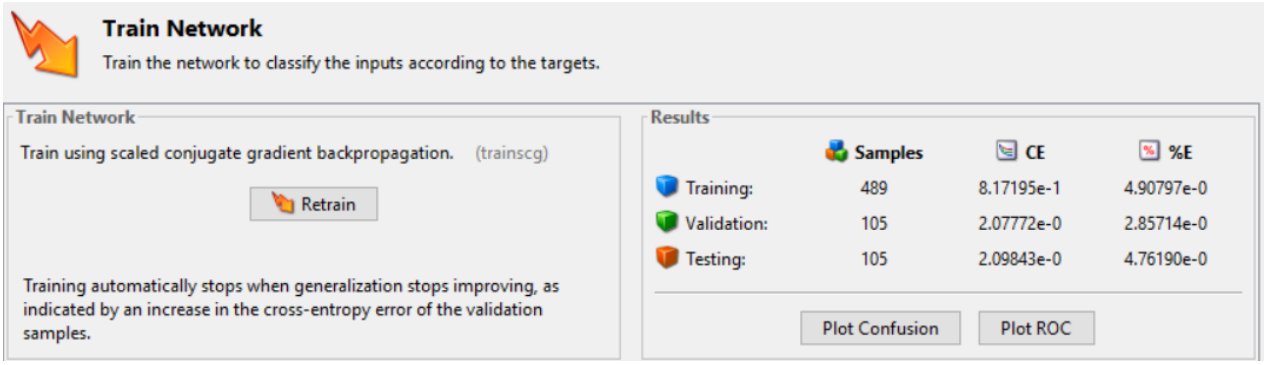
cancerInputs - a 9x699 matrix defining nine attributes of 699 biopsies.

1. Clump thickness
2. Uniformity of cell size
3. Uniformity of cell shape
4. Marginal Adhesion
5. Single epithelial cell size
6. Bare nuclei
7. Bland chomatin
8. Normal nucleoli
9. Mitoses

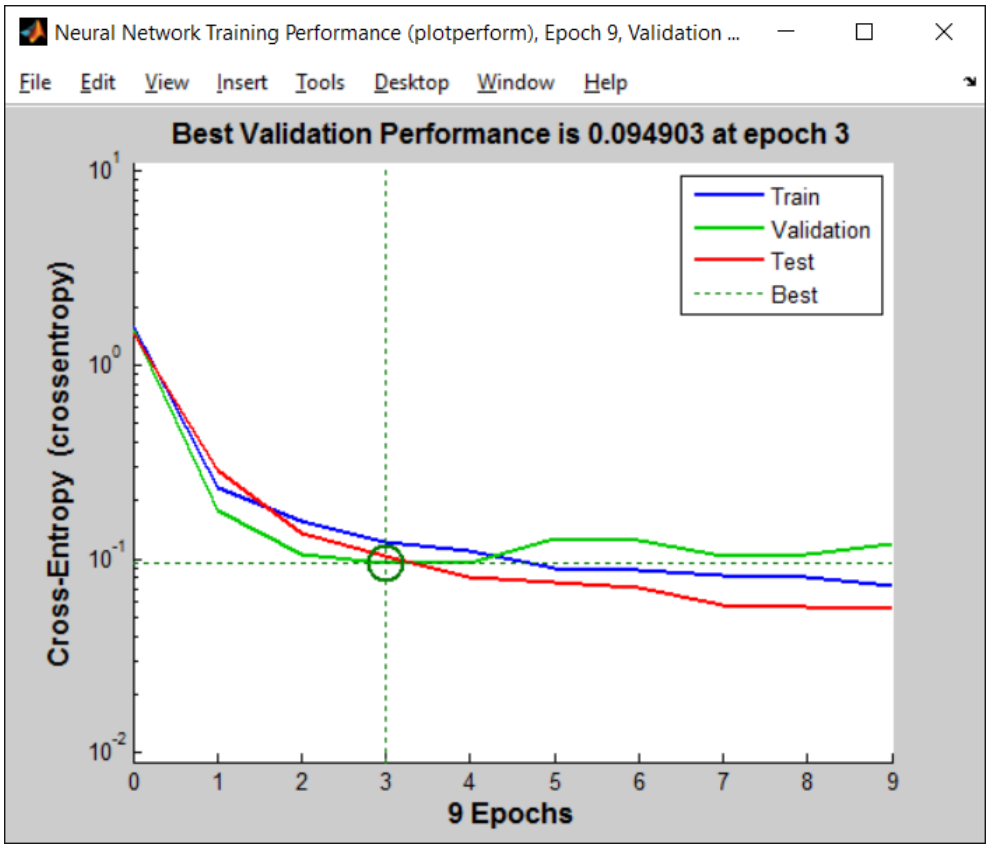
cancerTargets - a 2x966 matrix where each column indicates a correct category with a one in either element 1 or element 2.

1. Benign
2. Malignant

[X,T] = cancer_dataset loads the inputs and targets into variables of your own choosing.



You can see the weights settle down quickly, 9 iterations is enough to find the optimal weights which were those found on the third iteration.



Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.