Multilayer perceptron

A **multilayer perceptron** (**MLP**) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean *any* feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation); see § Terminology. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.[1]

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.[2][3] Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.[4]

Contents

Theory

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 linear algebra shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a *nonlinear* activation function that was developed to model the frequency of action potentials, or firing, of biological neurons.

The two historically common activation functions are both sigmoids, and are described by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}.$$

In recent developments of deep learning the rectifier linear unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the sigmoids.

The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function, which is similar in shape but ranges from 0 to 1. Here y_i is the output of the ith node (neuron) and v_i is the weighted sum of the input connections. Alternative activation functions have been proposed, including the

rectifier and softplus functions. More specialized activation functions include radial basis functions (used in radial basis networks, another class of supervised neural network models).

Layers

The MLP consists of three or more layers (an input and an output layer with one or more *hidden layers*) of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain weight w_{ij} to every node in the following layer.

Learning

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.

We can represent the degree of error in an output node j in the nth data point (training example) by $e_j(n) = d_j(n) - y_j(n)$, where d is the target value and y is the value produced by the perceptron. The node weights can then be adjusted based on corrections that minimize the error in the entire output, given by

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n).$$

Using gradient descent, the change in each weight is

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

where y_i is the output of the previous neuron and η is the *learning rate*, which is selected to ensure that the weights quickly converge to a response, without oscillations.

The derivative to be calculated depends on the induced local field v_j, which itself varies. It is easy to prove that for an output node this derivative can be simplified to

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n))$$

where ϕ' is the derivative of the activation function described above, which itself does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi'(v_j(n))\sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n).$$

This depends on the change in weights of the kth nodes, which represent the output layer. So to change the hidden layer weights, the output layer weights change according to the derivative of the activation function, and so this algorithm represents a backpropagation of the activation function.[5]

Terminology

The term "multilayer perceptron" does not refer to a single perceptron that has multiple layers. Rather, it contains many perceptrons that are organized into layers. An alternative is "multilayer perceptron network". Moreover, MLP "perceptrons" are not perceptrons in the strictest possible sense. True perceptrons are formally a special case of artificial neurons that use a threshold activation function such as the Heaviside step function. MLP perceptrons can employ arbitrary activation functions. A true perceptron performs *binary* classification, an MLP neuron is free to either perform classification or regression, depending upon its activation function.

The term "multilayer perceptron" later was applied without respect to nature of the nodes/layers, which can be composed of arbitrarily defined artificial neurons, and not perceptrons specifically. This interpretation avoids the loosening of the definition of "perceptron" to mean an artificial neuron in general.

Applications

MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation.

MLPs are universal function approximators as shown by Cybenko's theorem,[4] 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 make 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,[6] but thereafter faced strong competition from much simpler (and related[7]) support vector machines. Interest in backpropagation networks returned due to the successes of deep learning.

References

1. Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.
2. Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961
3. Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation (https://apps.dtic.mil/dtic/tr/fulltext/u2/a164453.pdf)". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.
4. Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function *Mathematics of Control, Signals, and Systems*, 2(4), 303–314.
5. Haykin, Simon (1998). *Neural Networks: A Comprehensive Foundation* (2 ed.). Prentice Hall. ISBN 0-13-273350-1.
6. Neural networks. II. What are they and why is everybody so interested in them now?; Wasserman, P.D.; Schwartz, T.; Page(s): 10-15; IEEE Expert, 1988, Volume 3, Issue 1
7. R. Collobert and S. Bengio (2004). Links between Perceptrons, MLPs and SVMs. Proc. Int'l Conf. on Machine Learning (ICML).

External links

- Weka: Open source data mining software with multilayer perceptron implementation (http://www.cs.waikato.ac.nz/ml/weka/).
- Neuroph Studio documentation, implements this algorithm and a few others (http://neuroph.sourceforge.net/).
