Available online at http://www.journalijdr.com

International Journal of DEVELOPMENT RESEARCH

ISSN: 2230-9926

International Journal of Development Research
Vol. 6, Issue, 04, pp. 7417-7420, April, 2016

Full Length Research Article

PERCEPTRON MODEL AND ITS APPLICATION IN INTELLIGENT SYSTEMS

*Arnab Acharyya, Suranjan Dhar and Sayani Bose

Computer Science and Technology, Technique Polytechnic Institute, Panchrokhi, Sugandhya, Hooghly, Pin-712102, West Bengal, India

ARTICLE INFO

Article History:
Received 27th January, 2016
Received in revised form
16th February, 2016
Accepted 19th March, 2016
Published online 27th April, 2016

Key Words:

Perceptron, Machine learning, Neural network, Pattern, Cluster, Intelligent system.

ABSTRACT

This paper is basically a review paper on Perceptron Model in Machine Learning and Intelligent Systems. This paper expands on the formative years of neural networks, going back to the pioneering work of McCulloch and Pitts in 1943. It also focuses on the perceptron convergence theorem. This theorem proves convergence of the perceptron as a linearly separable pattern classifier in finite number time-steps. We have also covered the basic area of machine learning, different learning techniques and application of perceptron in different problem solving.

INTRODUCTION

To solve a problem on a computer, we need an algorithm. An algorithm is a sequence of instructions that should be carried out to transform the input to output. For example, one can design an algorithm for sorting. The input is a set of unorganized numbers and the output is their ordered list. For the same task, there may be various algorithms and we may be interested in finding the most efficient one, requiring the least number of instructions or memory or both. For some tasks, however, we do not have an algorithm for example, to tell spam emails from legitimate emails. We know what the input is: an email document that in the simplest case is a file of characters. We know what the output should be: a yes/no output indicating whether the message is spam or not. We do not know how to transform the input to the output. What can be considered spam changes in time and from individual to individual. What we lack in knowledge, we make up for in data. We can easily compile thousands of example messages some of which we know to be spam and what we want is to "learn" what constitutes spam from them. In other words, we would like the computer (machine) to extract automatically the algorithm for this task. There is no need to learn to sort numbers, we already have algorithms for that; but there are many applications for which we do not have an algorithm but do have example data. With advances in computer technology, we currently have the ability to store and process large amounts of data, as well as to access it from physically distant locations over a computer network. Most of the application in now days are database oriented. These applications take the data input from a structured data set and process it and store the data in the same structured database. But machine learning is not just a database problem; it is a part of artificial intelligence. To be intelligent, a system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations. Machine learning helps us to find solutions in many problems like vision, speech recognition, and robotics. In this paper we are trying to explore the area of application of a perceptron in the field of machine learning and intelligent systems. Therefore we have to start with the basic ANN model to gather the prerequisite knowledge about a perceptron and how it works.

***Corresponding author: Arnab Acharyya,**
Computer Science and Technology, Technique Polytechnic Institute, Panchrokhi, Sugandhya, Hooghly, Pin-712102, West Bengal, India.

Basics of artificial neural network

A neural network is basically a model structure and an algorithm for fitting the model to some given data. The network approach uses a generic nonlinearity and allows all the parameters to be adjusted. In this way it can deal with a wide range of nonlinearities. Learning is the procedure of training a neural network to represent the dynamics of the problem, for instance in accordance with Fig. 1.

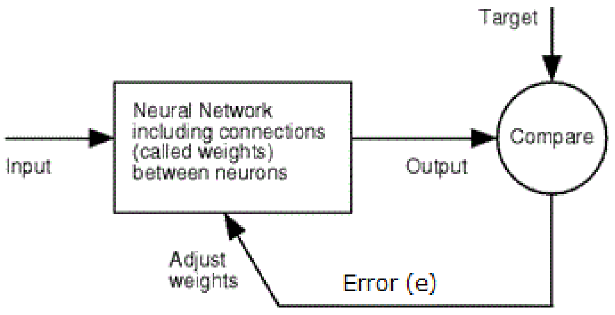
Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights
Error (e)

Fig. 1. A Block Diagram of ANN with learning capability

The inputs enter in to the neuron of the neural network and it generates an output. Between the generated output of the system and the target outputs, the error e, is used as the training signal. Neural networks have a potential for intelligent control systems because they can learn and adapt, they can approximate nonlinear functions, they are suited for parallel and distributed processing, and they naturally model multivariable systems.

Learning process of an intelligent system

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both. Learning process can be classified the following three categories. They are – Supervised Learning, Unsupervised Learning and Reinforcement Learning.

A. Supervised Learning

In supervised learning process the set of data (training data) consists of a set of input data and correct responses corresponding to every piece of data. Based on this training data, the algorithm has to generalize such that it is able to correctly (or with a low margin of error) respond to all possible inputs. The algorithm should produce sensible outputs for inputs that weren't encountered during training. This type of learning can be applied to solve the following two types of problems.

1) ***Regression Problem:*** Given some data, we assume that those values come from some sort of function and try to find out what the function is. We try to fit a mathematical function that describes a curve, such that the curve passes as close as possible to all the data points. So, regression is essentially a problem of function approximation or interpolation.

2) ***Classification Problem:*** Consists of taking input vectors and deciding which of the N classes they belong to, based on training from exemplars of each class. It is discrete (most of the time) in nature. That is an example belongs to precisely one class, and the set of classes covers the whole possible output space. To solve this problem we find 'decision boundaries' that can be used to separate out the different classes. Given the features that are used as inputs to the classifier, we need to identify some values of those features that will enable us to decide which class the current input belongs to.

B. Unsupervised Learning

The aim of unsupervised learning is to find clusters of similar inputs in the data without being explicitly told that some data points belong to one class and the other in other classes. The algorithm has to discover this similarity by itself.

C. Reinforcement Learning

Basically it lies in the middle of supervised and unsupervised learning. The algorithm is provided information about whether or not the answer is correct but not how to improve it. The reinforcement learner has to try out different strategies and see which works best. The algorithm searches over the state space of possible inputs and outputs in order to maximize a reward.

Perceptron Model

The perceptron is the basic processing element. It has inputs that may come from the environment or may be the outputs of other perceptrons. Associated with each input, $x_j \in R$, j = 1, . . . , d, is a connection weight, or synaptic weight $w_j \in R$, and the output, y, in the simplest case is a weighted sum of the inputs.

$$y = \sum_{j=1}^{d} x_j w_j + w_0 \qquad \text{...} (1)$$

w_0 is the intercept value to make the model more general; it is generally modeled as the weight coming from an extra bias unit, x_0, which is always +1. We can write the output of the perceptron as a dot product

$$y = w^T x \qquad \text{...} (2)$$

Where, $w=[w_0,w_1, . . . , w_d]^T$ and $x=[1, x_1, . . . , x_d]^T$ are augmented vectors to include also the bias weight and input. During testing, with given weights, w, for input x, we compute the output y. To implement a given task, we need to learn the weights w, the parameters of the system, such that correct outputs are generated given the inputs.

When d = 1 and x is fed from the environment through an input unit, we have $y = wx + w_0$ which is the equation of a line with w as the slope and w0 as the intercept. Thus this perceptron with one input and one output can be used to implement a linear fit. With more than one input, the line becomes a (hyper) plane, and the perceptron with more than one input can be used to implement multivariate linear fit. Given a sample, the parameters w_j can be found by regression.

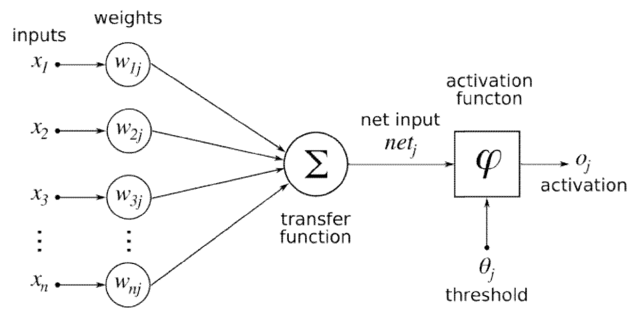

Fig. 2. Diagram of a perceptron model

The perceptron as defined in equation (1) defines a hyperplane and as such can be used to divide the input space into two: the half-space where it is positive and the half-space where it is negative. By using it to implement a linear discriminate function, the perceptron can separate two classes by checking the sign of the output. If we define f(.) as threshold function, then we can write

$$f(\theta) = \begin{cases} 1, & \theta > 0 \\ 0, & ot\ erwise \end{cases} \quad \text{...} (3)$$

Multilayer perceptron model

A perceptron that has a single layer of weights can only approximate linear functions of the input and cannot solve problems like the XOR, where the discrimininant to be estimated is nonlinear. Similarly, a perceptron cannot be used for nonlinear regression. This limitation does not apply to feedforward networks with intermediate or hidden layers between the input and the output layers. If used for classification, such multilayer perceptrons (MLP) can implement nonlinear discriminants and, if used for regression, can approximate nonlinear functions of the input. Input x is fed to the input layer (including the bias), the "activation" propagates in the forward direction, and the values of the hidden units z_h are calculated. Each hidden unit is a perceptron by itself and applies the nonlinear sigmoid function to its weighted sum:

$$z_h = \text{sigmoid}(w_h^T x) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj}x_j + w_{h0}\right)\right]}, \ h = 1, \ldots, H$$

The output y_i are perceptrons in the second layer taking the hidden units as their inputs

$$y_i = v_i^T z = \sum_{h-1}^{H} v_{ih} z_h + v_{i0}$$

where there is also a bias unit in the hidden layer, which we denote by z_0, and v_{i0} are the bias weights. The input layer of x_j is not counted since no computation is done there and when there is a hidden layer, this is a two-layer network. As usual, in a regression problem, there is no nonlinearity in the output layer in calculating y. In a two-class discrimination task, there is one sigmoid output unit and when there are K > 2 classes, there are K outputs with softmax as the output nonlinearity. If the hidden units' outputs were linear, the hidden layer would be of no use: linear combination of linear combinations is another linear combination. Sigmoid is the continuous, differentiable version of thresholding. We need differentiability because the learning equations we will see are gradient-based. Another sigmoid (S-shaped) nonlinear basis function that can be used is the hyperbolic tangent function, tanh, which ranges from 1 to +1, instead of 0 to +1. In practice, there is no difference between using the sigmoid and the tanh. Still another possibility is the Gaussian, which uses Euclidean distance instead of the dot product for similarity. The output is a linear combination of the nonlinear basis function values computed by the hidden units. It can be said that the hidden units make a nonlinear transformation from the d-dimensional input space to the H-dimensional space spanned by the hidden units, and, in this space, the second output layer implements a linear function. One is not limited to having one hidden layer, and more hidden layers with their own incoming weights can be placed after the first hidden layer with sigmoid hidden units, thus calculating nonlinear functions of the first layer of hidden units and implementing more complex functions of the inputs. In practice, people rarely go beyond one hidden layer since analyzing a network with many hidden layers is quite complicated; but sometimes when the hidden layer contains too many hidden units, it may be sensible to go to multiple hidden layers, preferring "long and narrow" networks to "short and fat" networks.

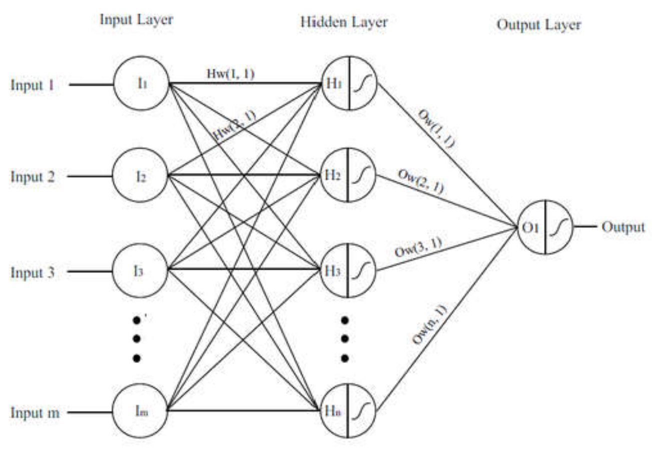

Fig. 3 Multilayer Perceptron Model

Conclusion

In the above discussion we have tried to cover the basic area of perceptron model in the field of machine learning. It is a survey paper based on the study of different learning methods in machine learning. We are currently working on the multilayer perceptron model to predict the weather conditions by input of some given data. We will publish our experiment results data along with the perceptron model in the upcoming research paper.

Acknowledgement

We are thankful to the Research & Development Cell of Technique Polytechnic Institute for their help and support. Our departmental faculty members and our departmental In-Charge Mr. P.K. Mitra helped us a lot to carry on this project. We also express our heartiest gratitude to Prof. Dr. Bimal Kr. Datta, H.O.D, CSE of BBIT for his excellent seminar on Artificial Neural Network which motivates us for this project.

REFERENCES

Akaike, H. 1974. 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control*, Vol. 19, No. 6, pp. 716–723.

Cavanaugh, J. E. 1997. 'Unifying the Deriviations for the Akaike and Corrected Akaike Information Criteria', *Statistics & Probability Letters*, Vol. 33, pp. 201–208.

Goldberg, D.E. 2001. "Genetic algorithms in search, optimization, and machine learning", Pearson Education.

Gupta, R. K., A. K. Bhunia, 2006. "An Application of real-coded Genetic Algorithm for integer linear programming", *AMO-Advanced Modeling and Optimization*, Volume 8, Number 1.

Hornik, K.., M. Stnchcombe and H White, 1989. Multilayer Feedforward Networks are Universal Approximators, Neural Network, 2:pg359- 366.

Hurvich, C. M. and C. Tsai, 1989. 'Regression and Time Series Model Selection in Small Samples', *Biometrika*, Vol. 76, pp. 297–307.

Kullback, S., and R. A. Leibler, 1951. 'On Information and Sufficiency', *The Annals of Mathematical Statistics*, Vol. 22, No. 1, pp. 79–86.

Linhart, H. and W. Zucchini, 1986. Model Selection, John Wiley and Sons.

Mak, B. L., H. Sockel, 2001. Info. & Mgmt, A confirmatory factor analysis of IS employee motivation and retention.

Man Mohan, Gupta P.K. 1992. Operations research, Methods and Solutions.Reading, Sultan Chand and Sons.

Poonam Garg, 2009. Advanced in Computer Science & Engineering, MacMillan Publication.

Srinivas, V., G.L. Thompson, 1973. "Benefit-cost Analysis of coding techniques for the Primal Transportation Algorithm.", *Journal of the association for computing machinery*, Vol.20, April, pp194-213

Vijyalakshmi Pai, G. A. and Rajasekaran, S. 2004. Neural networks, fuzzy logic and genetic algorithms, Synthesis and applications. Reading, Prentice-Hall of India.

****