Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. Engg. 91 (2016) 38613-38615

ISSN: 2229-712X

Pattern Recognition in Neural Networks

Kanika Bajaj
Department of Computer Science & Applications JMIT, Radaur.

ARTICLE INFO

Article history:
Received: 17 May 2012;
Received in revised form:
23 February 2016;
Accepted: 27 February 2016;

Keywords
Pattern recognition,
MLP,
Training set,
Classification,
Feed forward ANN.

ABSTRACT
In this paper, we review some pattern recognition learning methods and the models published in recent years. After giving the general description of pattern recognition, we discuss the Multi Layer Perceptron algorithm for classification in pattern recognition. Lastly, the example describing the implementation of MLP .The objective of this paper is to summarize and compare some of the methods for pattern recognition, and future research issues which need to be resolved and investigated further are given along with the new trends and ideas

Introduction

Pattern recognition can also be seen as a classification process. Its ultimate goal is to optimally extract patterns based on certain conditions and is to separate one class from the others. Pattern recognition was often achieved using linear and quadratic discriminants , the k-nearest neighbor classifier or the Parzen density estimator, template matching and Neural Networks. These methods are basically statistic. Neural Networks (Multi Layer Perceptron) have gained prominence in the field of pattern classification.

This paper is organized as follows. We first introduce some learning methods of 1. Pattern recognition[1] and basic models in section 2 .MLP in section 3 followed by implementation example in section 4. Conclusions are made in section 5.

Learning Methods & Models

Algorithms for pattern recognition depend on whether the learning process is supervised or unsupervised and it also depends on whether the algorithm is statistical or non-statistical in nature. Statistical algorithms[2] can further be categorized as discriminative or generative.

Supervised learning

It incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. It assumes that a set of training set has been provided, consisting of a set of instances that have been properly labeled by hand with the correct output. Usually, supervised learning is performed off-line. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.

Unsupervised learning

It uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. It assumes training data that has not been hand-labeled, and attempts to find inherent patterns in the data that can then be used to determine the correct output value for new data instances. A neural network learns on-line if it learns and operates at the same time. Unsupervised learning is performed on-line. Paradigms of unsupervised learning []are Hebbian learning and competitive learning.

Generative Model

Generative models are typically more flexible than discriminative models in expressing dependencies in complex learning tasks.

Examples of generative models include:

- Gaussian mixture model and other types of mixture model
- Hidden Markov model
- Naive Bayes
- AODE
- Latent Dirichlet allocation
- Restricted Boltzmann Machine

Discriminative Model

These class of models are used in machine learning for modeling the dependence of an unobserved variable *y* on an observed variable *x*. For classification discriminative models generally yield superior performance. These models belong to supervised learning .

Examples of discriminative models used in machine learning include:

- Logistic regression
- Linear discriminant analysis
- Support vector machines
- Boosting
- Conditional random fields
- Linear regression
- Neural networks(Multi Layer Perceptron)

Multi Layer Perception

The single layer neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron (figure 1) is the McCulloch and Pitts model (MCP). The difference from the single layer model is that the inputs are ‘weighted’; the effect that each input has at decision making is dependent on the weight of the particular input. The weight of

Tele:
E-mail addresses: bajajkanika555@gmail.com

an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

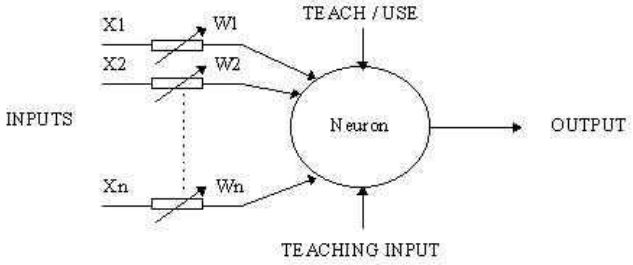


Figure 1. An MCP neuron

In mathematical terms, the neuron fires if and only if;

$$X1W1 + X2W2 + X3W3 + ... > T$$

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks. MLP utilizes a supervised learning[3] technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron , which can distinguish data that is not linearly separable.

An Example

The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

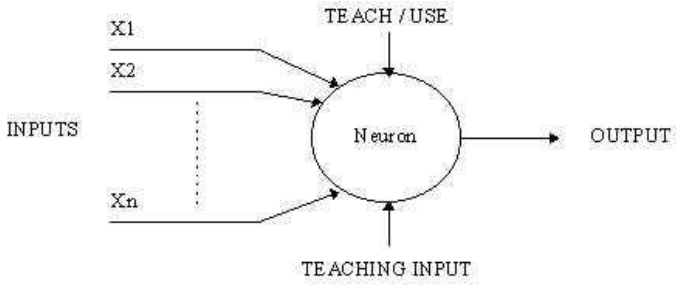


Fig 2. A simple neuron

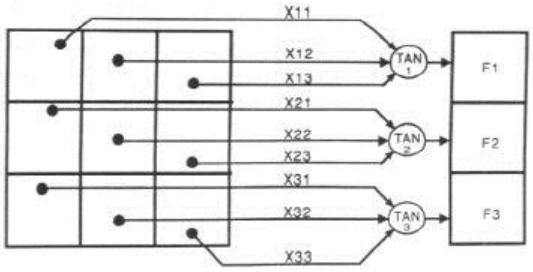
Firing Rule

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained. A simple firing rule can be implemented by using Hamming distance technique. The rule states:

"Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state".

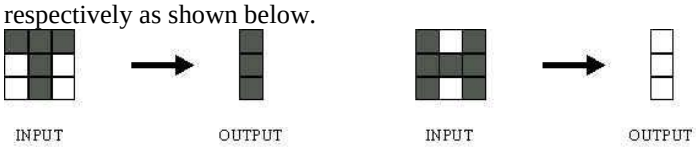
Implementation

An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward (figure 1) neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.



For example:

The network of figure 1 is trained to recognize the patterns T and H. The associated patterns are all black and all white respectively as shown below.



If we represent black squares with 0 and white squares with 1 then the truth tables for the 3 neurons after generalization are;

X11:	0	0	0	0	1	1	1	1
X12:	0	0	1	1	0	0	1	1
X13:	0	1	0	1	0	1	0	1
OUT:	0	0	1	1	0	0	1	1

Top neuron

X21:	0	0	0	0	1	1	1	1
X22:	0	0	1	1	0	0	1	1
X23:	0	1	0	1	0	1	0	1
OUT:	1	0/1	1	0/1	0/1	0	0/1	0

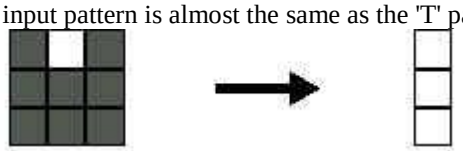
Middle neuron

X21:	0	0	0	0	1	1	1	1
X22:	0	0	1	1	0	0	1	1
X23:	0	1	0	1	0	1	0	1
OUT:	1	0	1	1	0	0	1	0

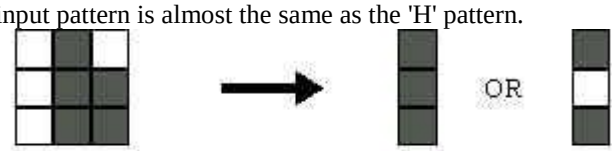
Bottom neuron

From the tables it can be seen the following associations can be extracted:

In this case, it is obvious that the output should be all blacks since the input pattern is almost the same as the 'T' pattern.



Here also, it is obvious that the output should be all whites since the input pattern is almost the same as the 'H' pattern.



Here, the top row is 2 errors away from the T and 3 from an H. So the top output is black. The middle row is 1 error away from both T and H so the output is random. The bottom row is 1 error away from T and 2 away from H. Therefore the output is black. The total output of the network is still in favor of the T shape.

Conclusion

The computing world has gained a lot from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain. The basic idea we get is: the more relevant patterns at your process, the better features subsets you obtain, the more simple your classifier will be applied, finally the better your decisions will be. Finally, I would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.. In summary, to get a better way for our final goal we should attempt to design a hybrid system combining with multiple models.

References

[1.] R. Callan. The Essence of Neural Networks. Prentice Hall, 1999.

[2.] Meijer, B.R.; “Rules and algorithms for the design of templates for template matching”, Pattern Recognition, 1992. Vol.1. Conference A: Computer Vision and Applications, 11th IAPR International Conference on, pp: 760 – 763, Aug. 1992.

[3.] Hush, D.R.; Horne, B.G.; “Progress in supervised neural networks”, Signal Processing Magazine, IEEE, Vol. 10, Issue: 1, pp:8 – 39, Jan. 1993

[4.] Girolami, M.; Chao He; “Probability density estimation from optimally condensed data samples” Pattern Analysis and Machine Intelligence, IEEE Transactions on, Volume: 25, Issue: 10 , pp:1253 – 1264,Oct. 2003.

[5.] Vapnik, V., The Nature of Statistical Learning Theory, Springer, 1995.

[6.] G.F. Luger. Arti_cial Intelligence. Addison Wesley, 2005.

[7.]http://comjnl.oxfordjournals.org/content/43/3/177.abstract