International Journal of Neural Networks
ISSN: 2249-2763 & E-ISSN: 2249-2771, Volume 2, Issue 1, 2012, pp.-20-24.
Available online at http://www.bioinfopublication.org/jouarchive.php?opt=&jouid=BPJ0000238

SIMPLE PERCEPTRON MODEL (SPM) FOR EVOLVING STREAMS IN MASSIVE DATA MINING (MDM)

SRIMANI P.K.[1] AND PATIL M.M.[2]*

[1]R&D Division, B.U., D.S.I., Bangalore-560 078, Karnataka, India.
[2]Dept. of ISE, JSSATE, Bangalore-560 060, Karnataka, India.
*Corresponding Author: Email- patilmalini31@yahoo.com

Received: October 25, 2012; Accepted: November 06, 2012

Abstract- Data sets which arrive continuously and rapidly grow over time are referred to as data streams. The nature of data streams is massive, infinite, very high speed and fast changing. For example, data generated by a satellite mounted remote sensor. Other examples include network monitoring data, user clicks in world wide web and usage of credit cards. Mining of such data streams is referred to as massive data mining. Traditional data mining techniques require multiple scans and are infeasible for mining data streams because of their nature. In the case of a model developed for a data stream the arrival speed of the data would be very high and the algorithm must process the data stream under very strict constraints of space and time. Massive Online Analysis (MOA) is a frame work used to mine data streams. The paper aims at developing a Simple Perceptron Learning model using different classification evaluators on evolving data streams in a MOA framework.

Keywords- Data streams, Perceptron model, Learning rate, Data Mining, Data generator, Massive Online Analysis.

Citation: Srimani P.K. and Patil M.M. (2012) Simple Perceptron Model (SPM) for Evolving Streams in Massive Data Mining (MDM). International Journal of Neural Networks, ISSN: 2249-2763 & E-ISSN: 2249-2771, Volume 2, Issue 1, pp.-20-24.

Introduction

On-line transactions in the financial market or retail industry, real-time surveillance systems, communication networks, electric power grids, internet traffic, industry production processes, scientific and engineering experiments, remote sensors, and other dynamic environments are best examples to understand the definition of data streams. The above mentioned dynamic environments generate potentially infinite volumes of data. In comparison with the traditional data sets, the stream data flows in and out of a computer system continuously with varying update rates. It is impossible to store all the data from the data stream. From the data stream its not possible to store all the data but only small summaries of data streams could be computed and stored. Data streams have to be processed online because of their arrival speed. Because of its tremendous volume it may be impossible to store an entire data stream or to scan through it multiple times. Moreover, stream data tend to be of a rather low level of abstraction. The distribution may change over time. The literature review reveals that it is possible to discover knowledge [3,4] or patterns from data streams. But the method of knowledge discovery should be different from the traditional data mining methods. In order to develop knowledge or patterns from data streams it is absolutely necessary to develop analysis methods comprising of single-scan, on-line, multilevel, multidimensional and stream processing. Massive online Analysis is one such frame work developed. Paper aims at developing a simple perceptron model using massive online analysis frame work.

Paper is organized as follows. Section II is about the biological history of Neural Network approach of which perceptron model is studied; Section III is about the related work; Section IV about the methodology to develop the perceptron model; Results and discussions are covered in section V; Finally conclusions are presented in section VI.

History of Neural Networks

Data mining is a confluence of many disciplines including artificial neural network (ANN). It is a learning technique which uses training and test data. It uses a technology that tries to produce intelligent behavior. It mimics the structure and function of human nervous system. The human brain consists of a network of neurons. Each neuron is made up of a number of nerve fibers called *dendrites* which are connected to the cell body where the cell nucleus is located. The *axon* is a long single fiber that originates from the cell body and branches near its end into a number of strands. At the ends of these strands there exist the transmitting ends of the synapse which are responsible for the other biological neurons. Actually the receiving ends of the synapse which are found on the dendrites as well as the cell body of biological neurons connect the biological neurons. A single axon typically makes thousands of synapses with other neurons.

Evolution of NN

The evolution of NN as a new computational model originates from the work done by McCulloch and Pitts in the year 1943. The model suggested by these scientists is basically referred to as a simple model of neuron which computes weighted sum of the inputs to the neuron and an output of 1 or 0. A '0' output would correspond to the inhibitory state of the neuron, while a '1' output would correspond to the excitory state of neuron.

From the above discussion it is apparent that the *nervous system in general can be abstracted as a weighted directed graph in which the nodes are neurons (nerve cells) and the edges between them are the connectors.* The type and strength of the interaction between the adjacent neurons is indicated by the weight on the respective edge. This leads to the simplest design of neural network which is called as perceptron.

The Perceptron

As discussed earlier the simplest kind of neural network is a perceptron that consists of a single neuron having several real-valued or binary inputs and a binary output. Actually a perceptron is trained to respond to certain inputs with certain desired outputs. The perceptron is shown in [Fig-1].

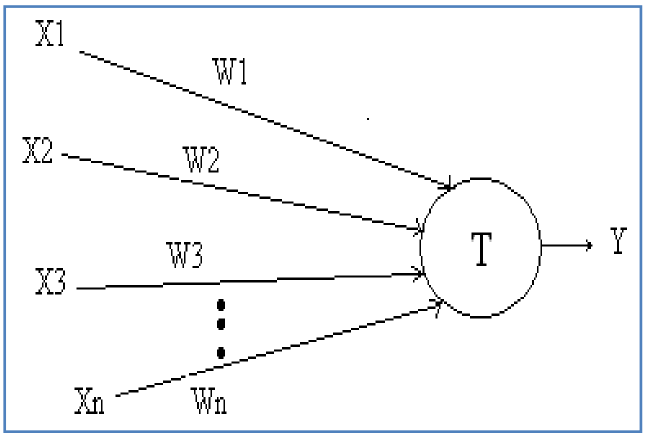

Fig. 1- The Perceptron

Here it important to note that the inputs pass through the weighted edges and are multiplied by the corresponding weights on those edges. The net input to the neuron at any time is the sum of all the weighted inputs. If this input exceeds a threshold then the neuron will trigger and produces an output of '1' or else the output will be '0'. This is illustrated in [Fig-1]. From [Fig-1] it is clear that the procedure constitutes the following two steps.

Step 1: To start with the weights of the perceptron are assigned in an arbitrary manner which are of the form (w_1, w_2,…., w_n).

Step 2: A series of inputs of the form (x_1, x_2,…., x_n) are presented to the perceptron during the training period.

Step 3: The threshold value T is set as per the net input value either less than or greater then the threshold value.

Step 4: Certainly there is desired output of either 0 or 1 for each such input.

Step 5: The final output is determined by the net input given by

$$Input = \sum_{i=1}^{n} W_i X_i \tag{1}$$

Here the output is '0' or '1' according as the net input is less or greater than the threshold value.

Exceptional Cases

During the process there could be some situations where the perceptron gives an undesirable output in which case some modifications are needed.

Case 1- The desired output is '0' and the net input is above threshold value.

In this case the actual output will be '1' and as a result the net input should decrease. This is possible only when the weights are decreased, which is determined using the perceptron learning algorithm which predicts that the decrease in the weight of an edge should be directly proportional to the input through that edge. This leads to the following expression:

New w(i) = Old w(i) + Cx_i where 'C' is a constant

Which means that the new weight of an edge i is equal to the old weight diminished by the product Cx_i

Case 2- The desired output is '1', but the net input is below the threshold value.

In this case the net input should increase which is possible only when the weights are increased. This suggests that the increase in weight of an edge should be proportional to the input through that edge. This leads to the following expression.

New w(i) = Old w(i) + Cx_i where 'C' is a constant

Which means that the new weight of an edge I is equal to the sum of old weight and the product Cx_i to summarize :

1. The perceptron learning algorithm will be called as a fixed increment rule when 'C' is very constant.
2. The perceptron may not correct its mistake immediately when 'C' is very small. The mistake free output is possible by the repeated application of the same input.
3. It is possible to choose 'C' in such a way that the most recent mistake could be avoided, which is known as the absolute correction rule.

Classification Using NN

Solving a classification problem using NNs involves several steps. Firstly, determine the number of nodes, input attributes, hidden layers. Secondly, determine the weights and functions. Thirdly, for each tuple in the training set propagate it through the network and evaluate the output prediction to the actual result and make appropriate classification.

Related Work

Outside the data stream world, there is prior work on using perceptrons or similar classifiers in decision trees. Decision trees in the form of the Perceptron Decision Trees are presented in [16]. In this paper each leaf node uses the perceptron as a classifier. Hybrid Decision Trees are presented in a hybrid learning approach [17]. In this approach the authors have combined decision trees with neural networks. Regression approach is extensively presented in [6,7]. In this paper the authors have proposed a fast incremental model tree for regression on static data streams. The work on concept drift[5], is proposed as FIRT-DD as an adaption of the FIMT algorithm to time-changing distributions. The algorithms use a perceptron learner at the leaves to perform regression. Considering classification methods for data streams. Two new ensemble learning methods are presented in [4] : one using bagging with decision trees of different size and one using ADWIN, an adaptive sliding window method that detects change and adjusts the size of the window correspondingly. In [8] the authors have proposed a clustering model for static streams. While in [9-13] an exhaustive work on performance of classifiers on Edu-data is presented. In [14,15] have worked extensively on the classification techniques using Naive Bayes and Naive Bayes Multinomial algorithms. No work is found in the literature which is mainly about the behavior of the perceptron model on data streams which uses different classification performance evaluators. The present approach concentrates on the behavior of the perceptron model on four different classification performance evaluators viz., EWMA, fading factor, basic and window classification performance evaluators respectively.

Methodology

The steps involved in simple perceptron model using MDM are presented in [Fig-2].

A. Steps in SPM

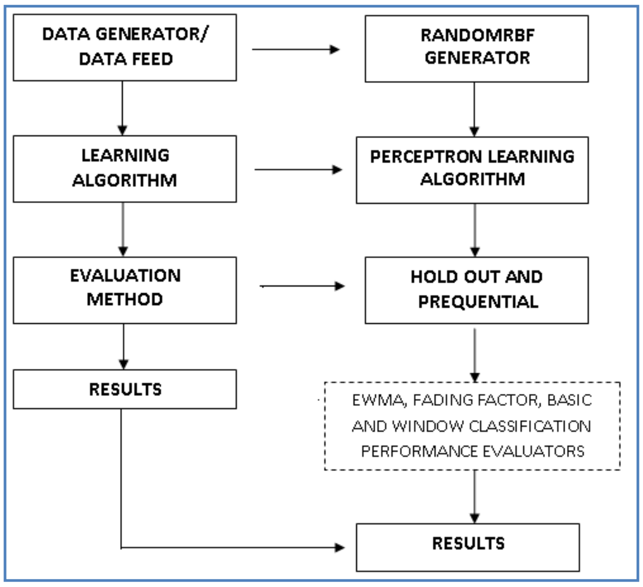
DATA GENERATOR/ DATA FEED
RANDOMRBF GENERATOR
LEARNING ALGORITHM
PERCEPTRON LEARNING ALGORITHM
EVALUATION METHOD
HOLD OUT AND PREQUENTIAL
RESULTS
EWMA, FADING FACTOR, BASIC AND WINDOW CLASSIFICATION PERFORMANCE EVALUATORS
RESULTS

Fig. 2- Steps in Simple Perceptron Model (SPM)

Data stream used is RANDOMRBFGENERATOR. Perceptron learning algorithm is used with varying learning rates. Basic evaluation methods used are Hold out and prequential methods. The hold out method is suitable when the division between train and test sets is predefined so that the results from different studies could be directly compared. Prequential method is suitable when each individual example can be used to test the model before it is used for training and accordingly the accuracy can be incrementally updated. Four different classification performance evaluators viz., EWMA, fading factor, basic and window classification performance evaluators considered.

B. Perceptron Learning Algorithm

The perceptron learning algorithm [2] is an example of supervised learning with reinforcement. The online version of the perceptron learning algorithm is used that employs the sigmoid activation function instead of the threshold activation function which results in the optimization of the squared error which has one perceptron per class value. Given a data stream (x_i, y_i), where x_i is an example and y_i is its example class, the main purpose of the classifier is to minimize the number of misclassified examples. Let $h_w(x_i)$ be the hypothesis function of the perceptron for instance x_i.

The mean-square error is represented as $J(w) = \frac{1}{2}\Sigma(y_i - h_w(x_i))^2$

instead of the 0-1 loss function, since it is differentiable. The classic perceptron takes a linear combination and thresholds it. The predicted class of the perceptron is $h_w(x_i) = \mathrm{sgn}(w^T x_i)$, where a bias weight with constant input is included. The hypothesis function $h_w = \sigma(w^T x)$, instead uses the sigmoid function $\sigma(x)=1/(1+e^{-x})$ since it has the property $\sigma'(x) = \sigma(x)(1- \sigma(x))$. Thus, it is possible to compute the gradient of the error function.

$$\nabla J = -\sum_i (y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))\nabla h_{\boldsymbol{w}}(\boldsymbol{x}_i) \qquad (2)$$

where for sigmoid hypothesis

$$\nabla h_{\boldsymbol{w}}(\boldsymbol{x}_i) = h_{\boldsymbol{w}}(\boldsymbol{x}_i)(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i)) \qquad (3)$$

with the following weight update rule

$$\boldsymbol{w} = \boldsymbol{w} + \eta \sum_i (y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))h_{\boldsymbol{w}}(\boldsymbol{x}_i)(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i))\boldsymbol{x}_i \qquad (4)$$

Since the work is carried out in a data stream scenario, rather than performing batch updates, the stochastic gradient descent is used where the weight vector is updated after every example. For multi-class problems, one perceptron is trained for each class. To classify an unseen instance x, the predictions $h_{w1}(x)$.... $h_{wn}(x)$ are obtained from the perceptrons, and the predicted class is $\mathrm{argmax}_{class} h_{wclass}(x)$. The pseudocode is shown in [Fig-3].

```
PERCEPTRON LEARNING($Stream, \eta$)
1  for each class
2        do PERCEPTRON LEARNING($Stream, class, \eta$)

PERCEPTRON LEARNING($Stream, class, \eta$)
1  ▷ Let $w_0$ and $\boldsymbol{w}$ be randomly initialized
2  for each example $(\boldsymbol{x}, y)$ in Stream
3        do if $class = y$
4              then $\delta = (1 - h_{\boldsymbol{w}}(\boldsymbol{x})) \cdot h_{\boldsymbol{w}}(\boldsymbol{x}) \cdot (1 - h_{\boldsymbol{w}}(\boldsymbol{x}))$
5              else  $\delta = (0 - h_{\boldsymbol{w}}(\boldsymbol{x})) \cdot h_{\boldsymbol{w}}(\boldsymbol{x}) \cdot (1 - h_{\boldsymbol{w}}(\boldsymbol{x}))$
6           $\boldsymbol{w} = \boldsymbol{w} + \eta \cdot \delta \cdot \boldsymbol{x}$

PERCEPTRON PREDICTION($\boldsymbol{x}$)
1  return $\arg\max_{class} h_{\boldsymbol{w}_{class}}(\boldsymbol{x})$
```

Fig. 3- Pseudo-algorithm of perceptron learning algorithm

C. Data Stream used in SPM

The data stream used in the analysis is RANDOMREBFGENERATOR [11]. It generates a random radial basis function stream. In order to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model this generator was designed. The RBF (Radial Basis Function) generator works as follows: A fixed number of random centroids are generated where, each center has the following features: random position, a single standard deviation, class label and weight. By selecting a center at random and taking weights into consideration new examples are generated so that centers with higher weight are more likely to be chosen. In order to offset the attribute values from the central point a random direction is chosen and by using gaussian the length of the displacement is randomly drawn where the chosan centroid determines the class label of the example. this process effectively creates normally distributed hypersphere of examples which surrounds each central point with varying densities. It is important to note that only numeric attributes are generated.

D. Massive Online Analysis (MOA) Framework

Massive data mining is performed using Massive online analysis (MOA) framework[1]. It is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. The design of MOA is done in such a way that it can handle the challenging problems of scalability of data streams including the implementation of the state of the art of algorithms to real world data sets. It consists of offline and online algorithms for classification and clustering. It also consists of tools for evaluation. At this juncture, it is emphasized that the only open source frame work that can handle evolving data streams is MOA. Another special feature of MOA is that it performs the evaluation of data stream learning algorithms on large streams under explicit memory limits. The method MDM mainly consists of the following steps.

Step 1: Select the task.

Step 2: Select the learner with different learning ratios (LR).

Step 3: Select the stream generator.

Step 4: Select the evaluators.

The model is configured with the above said steps and results are presented and discussed.

Experiments and Results

The experiments on Massive data mining are performed under the Massive online analysis frame work. The data stream selected for the analysis is RANDOMRBFGENERATOR with 10,000,000 instances.

The perceptron model constitutes the two basic classification evaluation parameters viz., evaluate prequential and hold out methods, along with four classification performance evaluators viz., EWMA, Fading Factor, Basic and Window classification performance evaluators (CPEs).

The learning ratios (LR) used in the analysis are 1, 1.01, 1.011 respectively. The results (accuracy predictions) are presented in [Table-1] and [Table-2] respectively. From [Table-1] it is found that fading factor classification performance evaluator performs in an excellent manner for all values of the learning ratio in prequential method. The graphical representation of the accuracy prediction for the case of prequential method is given in [Fig-4] which is self explanatory.

Table 1- Accuracy values for Evaluate Prequential Method

Classification Performance Evaluator	LR = 1	LR=1.01	LR= 1.011
Window	74.7	79.6.0	77.1
Basic	73.14	74.86	64.77
Fading Factor	99	99.99	100
EWMA	72.1	73.1	63.8

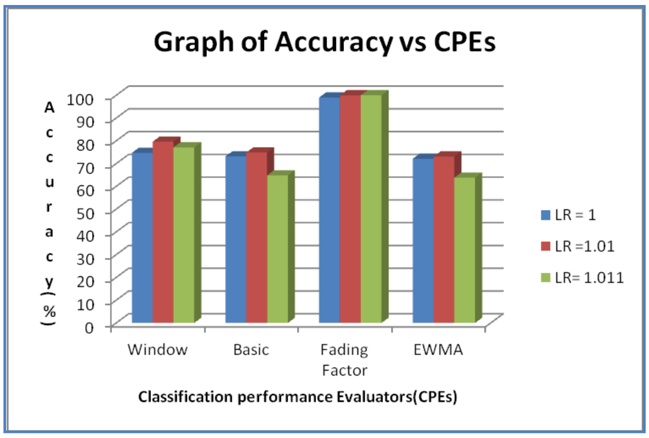

Fig. 4- Graph of a Accuracy vs. CPEs for prequential method

From [Table-2] it is found that fading factor classification performance evaluator performs in an excellent manner for all values of the learning ratio in hold out method. The graphical representation of the accuracy prediction for the case of hold out method is given in [Fig-5] which is self explanatory. It is interesting to know that both in the prequential and hold out methods the perceptron learning algorithm performs excellent in fading factor classification performance evaluation method.

Table 2- Accuracy values for hold-out Method

Classification Performance Evaluator	LR = 1	LR=1.01	LR= 1.011
Window	68.23	69.53	68.56
Basic	74.11	73.84	70.83
Fading Factor	99.99	99.99	99.99
EWMA	75.32	76.82	74.82

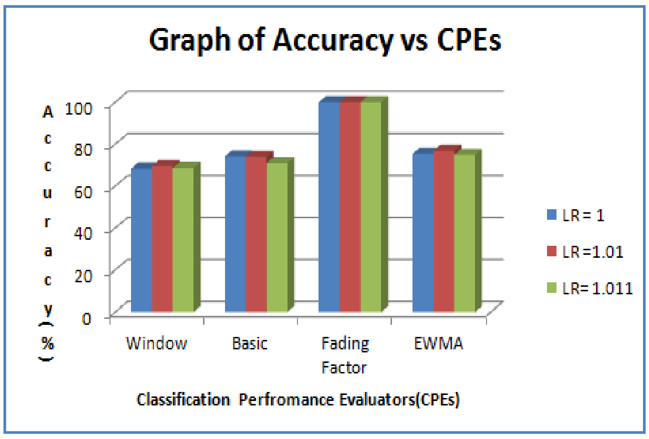

Fig. 5- Graph of a Accuracy vs. CPEs for Hold Out method

Conclusion

In this paper a simple perceptron learning model is developed for the data stream RANDOMRBFGENERATOR with 10,000,000 instances using massive online analysis framework. The characteristics features of the present investigation are as follows.

1. For the learning model four different classification performance evaluators viz., Basic, Window, fading factor and EWMA are considered.
2. Two basic classification evaluation methods namely evaluate prequential and hold out are used.
3. Three values of learning ratios are considered namely, 1, 1.01 and 1.011. When the perceptron model is configured under MOA framework, the evaluation process is based on the learning ratios.
4. The results of the present investigation predict that fading factor classification performance evaluator performs extremely well in prequential (LR,ACC:1.0,99; 1.01,99.99; 1.011,100) as well as in hold out methods (LR,ACC:1.0,99.99; 1.01,99.99;1.011,99.99).
5. No doubt the results of the present investigation provide an excellent platform for future research work but throws light on the qualitative as well as quantitative aspects of the problem.

Acknowledgment

One of the authors Mrs. Malini M. Patil acknowledges J.S.S Academy of Technical Education, Bangalore, Karnataka and Bharthiyaar University, Coimbatore, Tamilnadu, India., for providing the facilities for carrying out the research work.

References

[1] Albert B. and Richerd K. (2009) *Massive Online Analysis*.

[2] Albert B., Geoff H., Bernard P. and Eibe F. (2010) *Fast Perceptron Decision Tree Learning from Evolving Data Streams*.

[3] Bifet A., Holmes G., Pfahringer B. and Kirkby R. (2009) *New ensemble methods for evolving data streams*, 139-148.

[4] Bifet A. and Gavalda R. (2007) *Learning from time-changing data with adaptive windowing*.

[5] Domingos P. and Hulten G. (2000) *Mining High-speed Data Streams,* 71-80.

[6] Ikonomovska E. and Gama J. (2008) *Discovery Science*, 52-63.

[7] Ikonomovska E., Gama J., Sebastiao R. and Gjorgjevik D. (2009) *Discovery Science*, 121-135.

[8] Srimani P.K. and Patil M.M. (2010) *Data Stream Mining using Landmark Stream Model for Offline Data Streams*, 357-360.

[9] Srimani P.K. and Patil M.M. *AIP Conf. Proc. 1414*, 61-66.

[10] Srimani P.K. and Patil M.M. (2012) *Proceedings of ICICS by PSRC-Planetary Scientific Research Centre*, 35-30.

[11] Srimani P.K. and Patil M.M. (2012) *IJCR*, 4(1), 249-254.

[12] Srimani P.K. and Patil M.M.(2012) *IJCR*, 4(2), 183-190.

[13] Srimani P.K. and Patil M.M.(2012) *IJCR*, 4(2), 249-254.

[14] Sriman P.K. and Patil M.M. (2012) *Massive Data mining (MDM) on Data Streams Using Classification Algorithms,* 4(6), 2839-2848.

[15] Srimani P.K. and Patil M.M. (2012) *Proceedings of ICETM*.

[16] Srimani P.K. and Patil M.M. (2012) *Proceedings of ICCIT*.

[17] Utgo P.E. (1988) *AAAI*, 601-606.

[18] Zhou Z. and Chen Z. (2002) *Knowledge-based Systems*, 15(8), 515-528.