Mining with Neural Networks

Himashu Verma[1] and Prerna Mahajan[2]
[1]Research Scholar, Department of Computer Science, IITM Janakpuri, New Delhi,India
[2]Professor, Department of Computer Science, IITM Janakpuri, New Delhi, India
himanshu.iitm027@gmail.com, prerna.mahajan00@gmail.com

Abstract

In the present scenario, it is important to mine valuable data from the elephantine set of data. In order to analysis high-dimensional data that is a task where software tools can reasonably assist the data analyst, by visualizing, and thereby uncovering, the inherent structure and topology of the data collection. Here, the neural network models may be one solution that can produce results autonomously. Text mining, also referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. The purpose of this paper is to learn how the text is mined with Adaptive Neural Network technique so that a valuable output is to be generated.

Keywords: Text mining, Adaptive neural network, Neuron, Information retrieval, AHIGG, HMM's, IGG

1 Introduction

Data mining on text has been designated at various times as statistical text processing, knowledge discovery in text, intelligent text analysis, or natural language processing, depending on the application and the methodology that is used. Examples of text mining tasks include classifying documents into a set of specified topic areas (supervised learning), grouping documents such that each member of each group has similar meaning (clustering or unsupervised learning), and finding documents that satisfy some search criteria (information retrieval). Here, the Adaptive Hierarchical Incremental Grid Growing (AHIGG) is used in order to combines the features of the Growing Hierarchical Self-Organizing Map with the Incremental Grid Growing.

In the current scenario, everyone wants the output is to be measured in terms of correctness, effectiveness and easy to understand. To make this possible, various techniques is used so that a knowledgeable result is to be generated after mining the huge set of data.

2 Neural Network

It consist of a set of nodes (neurons/units) connected by links. Each links has a numeric weight. Each unit or neurons has:-

- A set of input links from other units.
- A set of output links to other units.
- A current activation level
- An activation function in order to compute the activation level in the next time step.

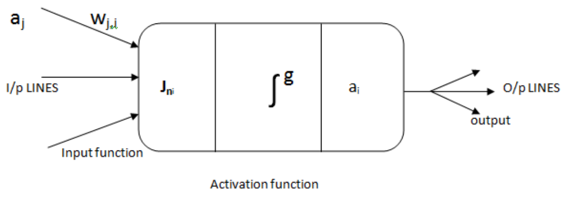

Figure 1: Node

Illustration of fig 1:-

The figure 1 depicts the typical structure of a node. There is a weight associated with every link. The following symbols depicts:-

- $\sum$ - that computes the total input that it receive from the other neuron.
- $\int^g$ - that is the function of the total input that the neuron "i" receives.

The total weighted inputs is the sum of the

Inputs activations times their respective weights:-

$$\mathbf{in_i = \sum W_{j,i}\ a_j}$$

Neural network is just a model for computation. Neural networks mimic the human brain by learning from a training dataset and applying the learning to generalize patterns for classification and prediction. These algorithms are effective when the data is shapeless and lacks any apparent pattern. The basic unit of an artificial neural network is modeled after the neurons in the brain. This unit is known as a node and is one of the two main structures of the neural network model. The other structure is the link that corresponds to the connection between neurons in the brain.

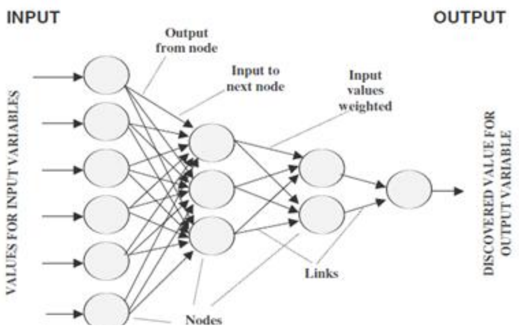

Figure 2: Neural Network

Consider a structure having 4 inputs and 3 outputs and we will be given a training data set. Training sets will be valid input output pairs. Set of cases where we are given i1, i2, i3, i4 for given values of i1, i2, i3, i4 as an input and we have to compute o1, o2, o3 values as an output. Several such cases will be given and the objective is to make this network learn that function.

At the end of the training if any input corresponding to any of the sample datasets is given then the "correct output" should be displayed. We can never be 100% sure whether it is giving the correct output for the others, but the objective is to make the neural network to learn the function so that it is able to extrapolate also the correct values for the input scenarios, which were not given in the training set.

Err = T – O

The main motive is that the Err(error) should be minimized.

Initially, the weights are randomly assigned. From the training set that contains the correct output "T", we are taking the input values and pass it to the network. Then the network produces the output. Based on this output, the error computation is done.

3 Applications Showcase – Neural Network

- CoEvolution of Neural Networks for Control of Pursuit & Evasion
- Learning the Distribution of Object Trajectories for Event Recognition
- Radiosity for Virtual Reality Systems
- Autonomous Walker & Swimming Eel
- Robocup: Robot World Cup Using
- HMM's for Audio-to-Visual Conversion
- Artificial Life: Galapagos
- Speechreading (Lipreading)
- Detection and Tracking of Moving Targets
- Real-time Target Identification for Security Applications
- Facial Animation
- Behavioral Animation and Evolution of Behavior
- A Three Layer Feed forward Neural Network
- Artificial Life for Graphics, Animation, Multimedia, and Virtual Reality: Siggraph '95 Showcase

4 Adaptive Hierarchical Incremental Grid Growing

The Adaptive Hierarchical Incremental Grid Growing (AHIGG), as proposed in [He01] and [MHDR03], was designed to combine the various features of the models, and to overcome the limitations for each of them. Simplied, the AHIGG combines the features of the Growing Hierarchical Self-Organizing Map with the Incremental Grid Growing as the model for the individual maps (instead of the Growing Grid as it is the case in the Growing Hierarchical Self-Organizing Map). Some more additions and improvements have been applied to the algorithm of the Incremental Grid Growing, e.g. to reduce the learning time by an improved initialisation rule, and a mechanism to determine a stagnation in the training process. In the rest of this section, the architecture and the algorithm of the AHIGG will be presented, before the model will be tested on real-world data.

4.1 The Architecture

The AHIGG is basically composed of a number of independent Incremental Grid Growing networks, which are arranged in hierarchical layers. Each of these layers depicts the input data at a specific level of granularity. On the first layer, a rough idea of the structure of the input data is given; each node of this layer may then be expanded to another IGG map in the next layer, thus giving a more detailed picture of this node's subset of the input data. The architecture of the IGG-maps, i.e. the size and layout, as well as the depth and balance of the hierarchy is determined automatically corresponding to the input space. The beginning of the AHIGG is a single node in a "virtual" layer 0, representing a statistical mean of all the input data. All nodes store values for the mean quantisation error *mqe*; each map has a vector *mqe*, i.e. the average quantisation error per vector in the map, denoted as *vMqe*.

In Figure 3, one possible architecture of a trained AHIGG is depicted.

Note that the hierarchy is not necessarily balanced.

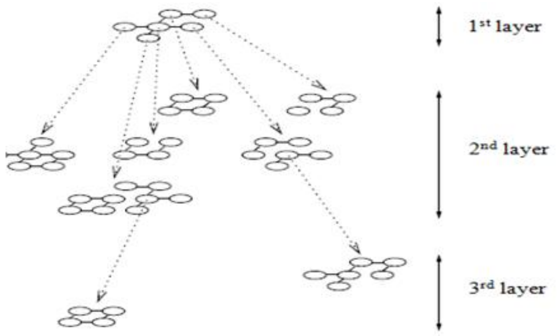

Figure 3: Architecture

4.2 The Training Algorithm

Three different phases can be identified in the training processes which are as follows:-

- the initialization
- the IGG-based training
- the hierarchical expansion phase

Initialisation

A single IGG map is initialised in the layer 1, usually having a 2£2 grid size, with all nodes connected in the grid. As a parent node to this map in the layer 1, the "virtual" layer 0 is initialised with a model vector *m*0, being the average of all the input vectors *x*. This node's *mqe* is calculated according to:

$$mqe_0 = \frac{1}{|C|} \sum_{x \in C} ||m_0 - x|$$

where *C* is the set of input vectors. The *mqe*0 will play a crucial role during the training process (*mqe*0 is a measure for the diversity in the input set).
All nodes in 1st layer will be initialized with a random model vector; contrary to the IGG, however, this initialisation is not completely random, but takes available qualitative information in the form of the model vector of the parent node into account:-

$$m_i = m_{parent} + mqe_{parent} \cdot \mathrm{v}_{rand}$$

where the subscript *parent* denotes this map's parent node, and *vrand* is a random vector of length 1., i.e. we limit the range to the n-dimensional subspace with *mparent* as centre, and *mqeparent* as radius.

4.3 IGG based learning

The map is trained in training cycles according to the standard SOM algorithm:
A randomly chosen input vector is presented to the map, and the winning node is chosen according to Equation:-

$$c(x,t) = arg \min_i \{ ||x(t) - m_i(t)|| \}$$

Then, the model vectors of the winning node and nodes within the neighbourhood-range are adapted, according to Equation:-

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{ci}(t)[x(t) - m_i(t)]$$

For the learning rate *α*, a time decreasing function is taken, allowing a global organisation in the beginning of the training process, and a more local ordering towards its end. After each training cycle, the function is reset to its initial value.

4.4 Hierarchical Expansion

When a map in the AHIGG is completely trained, it is examined whether any of the map's nodes requires a higher resolution of its input patterns, and therefore should be expanded hierarchically, i.e. a new IGG map in the next layer will be added for this node. As a measure for nodes representing their input space inadequately, a node *i*'s mean quantisation error, calculated according to:

$$mqe_i = \frac{1}{|C_i|} \sum_{x \in C_i} ||m_i - x||,$$

where *Ci* is the set of input patterns mapped on this node, and *mi* its model vector, is utilised. The, a simple threshold logic is used: with a parameter *T*2, 0 < *T*2 < 1, all nodes for which

$$mqe_i > \tau_2 \times mqe_0$$

holds true are expanded. Here, the parameter *T*2 is thus responsible for guiding the hierarchical growth process: the lower the value for *T*2, the deeper the hierarchy will develop. The newly established map in the next layer is initialized in a similar way as the map in 1st layer.

5 Conclusion

Mining text with neural network technique is responsible for generating effective output from a large set of data. Neural network-algorithms are effective when the data is shapeless and lacks any apparent pattern. The future of Neural Networks is wide open, and may lead to many answers and/or questions.

The Adaptive Hierarchical Incremental Grid Growing (AHIGG) is used in order to combines the features of the Growing Hierarchical Self-Organizing Map with the Incremental Grid Growing.

References

[1] https://en.wikipedia.org/wiki/Artificial_neura l_network, visited on 10/12/2017.

[2] Mayor R., "Text mining with adaptive neural networks", http://www.ifs.tuwien.ac.at/~mayer/publicati ons/pdf/may_thesis04.pdf, visited on 12/12/2017.

[3] https://en.wikipedia.org/wiki/Textmining, visited on 15/12/2017.

[4] Adaptive Neural Network Filters, http://in.mathworks.com/help/nnet/ug/adapti ve-neural-network-filters.html visited on 25/12/2017.

[5] Hui S. H., "Analyzing the topology of high-dimensional data using the adaptive hierarchical incremental grid growing", Diplomarbeit, Technische Universität Wien, Austria (2001).

[6] Dieter M., Shao H.H., Michael D. and Andreas R., "Adaptive hierarchical incremental grid growing: An architecture for high-dimensional data visualization", *Proceedings of the 4th Workshop on Self-Organizing Maps, Advances in Self-Organizing Maps*, Kitakyushu, Japan, pp 293-298 (2003).