Managing Document using Deep Belief Network Algorithm

Yasmin Shafiee¹, Azliza Mohd-Ali², Sofianita Mutalib³ and Shuzlina Abdul-Rahman⁴
¹,²,³,⁴Faculty of Computer and Mathematical Sciences, UniversitiTeknologi MARA, 40450 Shah Alam, Selangor
¹yasminShafiee@gmail.com, ²{azliza, sofi, shuzlina}@tmsk.uitm.edu.my

ABSTRACT

Document management becomes one of the important issues nowadays due to the exponential growth of documents in the Internet. Without proper management, documents are difficult to track and easily loss. This paper investigates a deep belief network algorithm with the aim to classify the document based on its category. The deep belief network constructs two layers of hidden nodes; the pre-training phase and the fine tuning phase for the modeling process. Parameter such as number of hidden nodes, learning rate and epoch size were investigated in getting the optimum model. After a series of experiment, our results showed that deep belief network outperformed the k-Nearest Neighbors algorithm.

Keywords: Deep Belief Networks, Document Classification, Restricted Boltzmann Machine

I INTRODUCTION

The exponential growth of document, the emergent network technologies and the existence of the websites have encouraged many individuals and organization to make their data available online. Document classification is one of the popular researches in classification data mining that helps users in finding their information quickly and efficiently. It aims to map text documents into one or more predefined category based on its contents of keyword. Therefore, document classification becomes important where it can be utilized in many applications such as classification of news stories, e-mail classification (Al-Sallab& A-Rashwan, 2012) and others (Al-Diabat, 2012; Nidhi& Vishal, 2011). The remainder of this paper is organised as follows. Section 2 explains DBN by describing the underlying theories of the algorithms. Section 3 and 4 present the method and the experimental setup. Section 5 discusses the results and finally, in Section 6, several points are brought to a conclusion.

II DEEP BELIEF NETWORKS

Deep Belief Network (DBN) is a generative model consisting of multiple, stacked level of neural networks that each can efficiently represent non-linear ties in training data. DBN model enables the network to generate visible activations based on hidden unit’s states which represents the network belief. DBN algorithm is composed of two phases, the first phase is pre-training phase and the second phase is fine tuning phase (Ziqianget.al., 2010). In the first phase, a stack of Restricted Boltzmann Machine (RBM) is trained with the unlabelled training set examples. Meanwhile the second phase is to fine tune a deep artificial neural network using back propagation neural network or other classifier, starting from the weight obtained in pre-training.

A. Pre-training Phase: Restricted Boltzmann Machine

Restricted Boltzmann Machine (RBM) is a network that consists of a layer of visual neurons and a layer of hidden neurons. Normally, Restricted Boltzmann Machine is the structure block networks for the DBN. When composing RBM together with DBN, one would suppose to be able to represent highly non-linear patterns in the training data because RBM
is non-linear. DBN uses RBM to simplify for training time complexity, which only allows connections between a hidden neuron and a visible neuron and no connection between two visible or hidden neurons. In RBM, the energy is defined as (Chandarakala, et.al., 2011):

\[
\text{Energy} (v, h) = h^T W v + b^T v + c^T h \quad (1)
\]

where\
\( (v, h) \) is the configuration \\
\( W \) is the weight matrix \\
\( b \) is the bias vectors on the visible neuron \\
\( c \) is the bias vectors on the hidden neurons

The goal of RBM is to update the weights and biases in order to reduce the energy of the configuration of the network when training data is used as the input. During the training process, most documents are likely to have similar training vectors and therefore similar configuration will be generated on the RBM. The probability of a configuration is determined by the energy of a configuration, whereby the lower energy configurations will have higher probability in the function:

\[
P(v, h) = \frac{1}{Z} e^{-\text{energy}} \quad (2)
\]

where \( Z \) is a normalization constant

After that, Contrastive Divergence algorithm is used for sampling the hidden units and visible units. A step in the Contrastive Divergence algorithm is thus taken to calculate these parameters as follow:

\[
P(V_k = 1|h) = \text{sigmoid} \left( -b_k - \sum_j W_{kj} h_j \right) \quad (3)
\]

\[
P(h_j = 1|v) = \text{sigmoid} \left( -c_j - \sum_k W_{kj} v_j \right)
\]

Finally, the weights and biases for the RBM are updated once the sampling of the hidden units and the visible units are completed.

B. Fine tune Phase: Back propagation Algorithm

Fine tuning phase is simply the ordinary back propagation algorithm that is applied after the pre-training phase takes place. For classification tasks, a layer of width is equals to the number of targets or classes, is added on top of the network. Each neuron of this layer is activated for each class label while the others are deactivated. The back propagation starts from the weights obtained in pre-training phase. The top layer activations are obtained for each training set example, or batch of examples, is obtained in the forward path, and then the error signal between the obtained activations and required targets is back propagated in the network for weights adjustment.

III METHOD

The methodology of this work was carried out in four stages: data collection, document pre-processing, text classifier construction and experimental setup. The next section explains the detail of these stages.

A. Data Collection

This study investigates a new dataset obtained from the 2nd International Conference on User Science and Engineering 2011 (i-USEr 2011). This conference data consist of 5 categories of articles such as Affective Design, Methodologies, Human Computer Interaction and Education Training, Culture Centered Design, and User Experience Usability. For each category, 10 articles were selected and this contributed to 50 articles. This dataset is divided into training and testing set with a ratio of 80:20 in which 80% of the articles are used for the training and 20% of the articles are used for the testing purposes. We focus on keyword selection found from the abstract section. The number of words for each text document is between 90 to 400 words.

B. Document Pre-processing

Data pre-processing is the stage to identify the important keywords that understandable by the prototype. In order to perform Document Classification, this prototype receives the keywords as the neural network input. The pre-processing stage is a function to identify important keywords from document. There are four important steps as follows:

Tokenization Analysis: Tokenizing the documents is the first step done onto the raw data set, in which it starts to process the documents by changing the condition of the data set. Tokenizing is the process that turns the documents into list of words, whereby all the words is break down at some “non-letters” point, such as dash (-), space, tab space, and so on. For instance, the sentence “Affective Design” is then break down into two words which are “Affective” and “Design”, with spaces between them act as the break down point. This tokenizing process is very helpful for the system to “digest” all the data set thoroughly and entirely, without having to sacrifice the raw data set by changing its structure, thus keeping the original and pure data collected.

Removal of Stop Words: Stop words are words which are filtered out earlier to, or after, processing of natural language data (text). It is classified as words that are not important for many text-related
processes. The stop words usually contain non-semantic words such as prepositions, conjunctions and pronouns (Suresh et. al, 2011). “And”, “are”, “of”, “the” and “with”—are just some few of the common stop words. The stop words are removed from all the documents before the classification process start in order to yield a much better classification results. Other than that, by removing stop words from the documents, it will minimize the data needed to be processed for the later classification process, which will cause the data to be processed faster, and in more effective manner. Due to the documents is in English, therefore the stop words list used is the English stop words.

**Stemming**: Stemming is a process of decreasing modified words into their root forms. This is required also for the similarity measurement process for the classification. The process of stemming the words can be very challenging, as the word modified version can be in various form. Therefore, it is not needed for the stem word to be exactly the same as the original root word, but just enough to form the relation of the word with the original form. Examples of stemming process include removing all the suffixes on the text, such as if the word ends with ‘ing’, ‘ed’, ‘ies’, and so on, those ends will be removed. These stemmed documents are then used for the classification process.

**Keywords Selection**: After all the three processes are performed, the keywords or term are defined based on the higher frequency of words count. The selection of keywords or terms (or the word count?) will be used to find the similarity with the new article that has been uploaded by the user.

C. **Classifier Construction**

Classifier construction is the key component of automatic document classification. The role of this element is to build a classifier by learning from predefined documents, which will be used to classify unidentified documents. The DBN was implemented using JAVA language and the performance was measured based on classification accuracy using hold-out method.

D. **Experimental Setup**

Three series of experiments were performed as follows:

**Experiment 1**: learning rate is varied between 0.1 until 0.5 with a constant value of epoch, 600 epochs.

**Experiment 2**: number of hidden neuron is varied from 100, 200, 300, 400, and 500 neurons for both layer, Layer 1 and Layer 2. The best learning rate in the previous experiment (Experiment 1) and a constant value of epoch is used.

**Experiment 3**: different number of epoch is applied; the best value of learning rate and hidden neuron obtained from Experiment 1 and 2 are used.

V **RESULTS AND DISCUSSION**

Table 1 shows the training result for the first experiment that try to find the best value of learning rate. The highest accuracy is achieved at 0.5 learning rate, that is 69% accurate. Table 2 shows the good performance is achieved at hidden neurons 500:500. The result of training shows, the decreasing of the number of hidden neuron make the testing accuracy for all sample decreases. This result indicates the importance of selecting a suitable size of hidden neuron at each layer. However, too many neurons make the learning slow, thus a suitable size of hidden neuron needs to be identified. Table 3 shows the best performance is achieved at epoch 600. The increase number of epoch does not help in improving the accuracy. This can be seen at epoch 900 when the accuracy is achieved at 61%. This experiment further strengthens our understanding that an optimal size of epoch is important in getting a good performance and the high number of epoch does not guarantee a good result.

<table>
<thead>
<tr>
<th>Learning Rate</th>
<th>#Epoch</th>
<th>MSE</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>600</td>
<td>0.02</td>
<td>64%</td>
</tr>
<tr>
<td>0.2</td>
<td>600</td>
<td>0.01</td>
<td>60%</td>
</tr>
<tr>
<td>0.3</td>
<td>600</td>
<td>0.01</td>
<td>66%</td>
</tr>
<tr>
<td>0.4</td>
<td>600</td>
<td>0.01</td>
<td>63%</td>
</tr>
<tr>
<td>0.5</td>
<td>600</td>
<td>0.01</td>
<td>69%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hidden Neuron</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>#Epoch</th>
<th>MSE</th>
<th>Testing Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500</td>
<td>600</td>
<td>0.01</td>
<td>69%</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>400</td>
<td>600</td>
<td>0.01</td>
<td>68%</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td>600</td>
<td>0.01</td>
<td>60%</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>600</td>
<td>0.01</td>
<td>53%</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>600</td>
<td>0.01</td>
<td>55%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Rate</th>
<th>#Epoch</th>
<th>MSE</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>900</td>
<td>0.01</td>
<td>61%</td>
</tr>
<tr>
<td>0.5</td>
<td>800</td>
<td>0.01</td>
<td>63%</td>
</tr>
<tr>
<td>0.5</td>
<td>700</td>
<td>0.01</td>
<td>60%</td>
</tr>
<tr>
<td>0.5</td>
<td>600</td>
<td>0.01</td>
<td>69%</td>
</tr>
<tr>
<td>0.5</td>
<td>500</td>
<td>0.01</td>
<td>61%</td>
</tr>
</tbody>
</table>

All these results revealed the importance of experimenting different values of parameter, in particular the number of learning rate, size of
neurons at both layers layer 1 and layer 2 as well as the size of epochs. Based on these results, we obtained an optimum model with an accuracy of 69% at learning rate 0.5, hidden neuron 500:500 and epoch size of 600. Figure 1 illustrates the final architecture of Deep Belief Networks.

Figure 1. Deep Belief Networks Architecture

Further experiment was performed by comparing with k-NN and the results are presented in Table 4. As can be seen, DBN outperformed k-NN algorithm with more than 30% difference in accuracy. The good performance of DBN shows its capability in classifying large amount of data. Compared to k-NN, the means squared error for DBN is also closed to zero. However, the DBN has a significantly longer training time compared to k-NN.

Table 4. Accuracy For Different Epoch Size

<table>
<thead>
<tr>
<th>Epoch Size</th>
<th>DBN</th>
<th>k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Accuracy</td>
<td>69%</td>
<td>30%</td>
</tr>
<tr>
<td>Mean Squared Error</td>
<td>0.0099</td>
<td>0.5009</td>
</tr>
</tbody>
</table>

Document classification with DBN algorithm is an example of intelligent systems that able to perform document management according specific categorization. An intelligent system is expected to have the ability to learn and apply it’s knowledge to handle complex situation. The system could also react quickly and correctly to a new situation, which is basically simulate the same action by human. This piece of knowledge are gained by learning from the set of data given to the system. In this research context, the document classification is able to distinguish the important keyword given in the text and then calculate the frequency of the word counts, and finally determine the document to be classified into different categories. The process of reviewing hundreds of document may be limited by human time and capabilities. Though, the system did not give hundred percent accuracy, the output would be useful for the first screening and later to be validated by the reviewers’ of the documents.

V CONCLUSION

This study investigated deep belief network algorithm by experimenting a series of experiment on parameters such as learning rate, size of hidden neuron and the epoch size. The study exercised this algorithm using a database extracted from i-USeR conference dataset. It is shown that the higher number of epoch size does not guarantee a good result and DBN requires much longer training time compared to k-NN. Our future work is to compare the DBN with another robust algorithm such as support vector machines. Moreover, the study could be later be enhanced by retrieving the context of the words and not only depending on the word counts.

ACKNOWLEDGMENT

The authors would like to thank to Research Management Institute, UniversitiTeknologi MARA, Malaysia for the RIF 624/2012 research grant.

REFERENCES


