CHAPTER 4

Text Categorization by Back Propagation Network
Chapter-4

Text Categorization by Back Propagation Network

4.1 Introduction

The Back Propagation network (BPN) algorithm is a neural network algorithm used for many image processing problems and it is found to produce good results. Considering this, the author of this research decided to use this algorithm for the text categorization problem too with slight modifications. Hence this chapter deals with the BPN algorithm. This chapter speaks about the implementation of Back propagation network algorithm for the Text Categorization problem. Also it discusses about categorizing the text by non-linear, feed-forward neural network trained by the Back propagation learning rule. That is, the neural network for classifying the text under supervised learning is applied. The BPN algorithm to the text categorization problem and the implementation details of it are also provided in this chapter.

4.2 Need for Neural Networks

Back propagation method is a technique of Artificial Neural Network System (ANS). There is a strong reason for using ANS in text categorization. For the problems which cannot be solved sequentially or by sequential algorithms ANS provides better solution. Apart from text categorization, for the other applications like pattern matching of images, non-sequential algorithms are provided by artificial neural network. These algorithms are better in performance. Among the various algorithms provided by ANS, Back propagation is a popular algorithm. Back propagation as an ANS is very useful in recognizing complex patterns and performing nontrivial
mapping functions. Recall the figure 2.4 found in chapter 2. That figure represented a simple Back propagation diagram. The rounded objects represent the neurons or processing elements of a neural network. The directed lines that are connecting the neurons are called as weights. Also every line of processing elements is a layer of a network. Thus in figure 2.4 there are three layers available. Though there can be multiple layers present in ANS generally there will be three layers present in Back propagation network (BPN).

4.3 Classification Problem

Though this problem has been discussed earlier in chapter 2, for the better understanding of this chapter it is revisited. Classification problem can be stated like this in general. If a set of objects are present, it has to be divided into different groups. This task of grouping objects into different groups based on their characteristics and properties is called as classification. While discussing about classification it is important to distinguish it from clustering. In classification the objects are assigned to predefined categories. These predefined categories are called as classes. But in clustering there are no such predefined classes existing. The clustering module has to identify such classes and group the objects accordingly.

In computer based classification, an object is usually represented by a set of attributes or features. Each feature corresponds to the property of the object. Usually the set of features are grouped to form a feature vector, in which each vector component corresponds to a feature. For example, the following is a feature vector(X) with n features (f1, f2, …fn):

\[ X = \langle f_1, f_2, \ldots, f_n \rangle \]
Figure 4.1 shows a general model of the computer based classification task. In this model, one set of n-dimensional feature vectors representing the objects to be classified is given to the classifier as input. Based on the features contained in each input feature vector, the classifier will make the decision of class assignment for the objects. This decision is represented by an m-dimensional output vector, where m is equal to the number of pre-defined classes in particular classification task.

![Figure 4.1: Classification Model (Courtesy: Lam Lai Yin, Dominic Savio, 1996)](image)

Here the classifier is a function which maps an n-dimensional feature vector into an m-dimensional output vector. The output vector from the classifier is sometimes called a classification vector.

### 4.4 Feature Identification

Set of features that describe the properties of the objects have a greater impact on the classification of the objects. Hence to improve the accuracy of the classifier, identifying a set of “good” features for object representation sometimes take lots of time and becomes a critical step in constructing the classification system. In most of the cases this involves first identifying a set of “raw” features or measurements of the various object properties and then these features have to be converted to “improved” features. These improved features will be suitable for particular classification.
This process of feature refinement is known as feature selection or feature extraction. These things have been discussed in chapter 3. But still a few more points have to be highlighted here as it would give holistic idea of the problem.

Feature selection is the process of selecting a subset of \( d \) features from an initial set of \( D \) features, where \( d < D \). In many cases, this is done manually in such a way that some human experts choose those features from the initial set that they think are useful for classification. Other features are dropped. The disadvantage of manual approach is that it is time consuming and it requires lot of experience and domain knowledge. Hence it is desirable to automate the feature selection process.

One of the problems with the selection is that there is always information loss as some of the low score features are being filtered. It is true that a good scoring scheme will minimize the information loss. But this is rarely optimal especially when the features are not independent of each other. Moreover, feature selection is not effective if none of the original features are good, as there are no new features created. Feature extraction process can be described as application of set of operators on one or more original features. In this process new features may also emerge. Depending on the number of features being acted on and the number of features produced, four types of operators are identified: one-to-one operators which transform a single feature into a single new feature, one-to-many operators which transform a single feature into multiple new features, many-to-one operators which transform multiple features into a single new feature and many-to-many operators which transform multiple features into multiple features.
Coming back to the text categorization system there are three fundamental steps that have to be followed. The first step is to identify the feature set based on the content of documents. In general there will be large number of features after the first step is completed. In order to improve scalability of text classification system, dimensionality reduction should be applied. This reduction will reduce the number of features identified initially. Dimensionality reduction will be the second step. As a last step it is proposed to use three layered feed-forward network trained by Back propagation as the text classifier. As the learning paradigm for Back propagation neural networks is supervised learning, the training set should include a set of training documents, together with a specification of predefined categories.

Apart from these to select the features one has to do word stemming, stop words removal, assigning weights to feature which are covered elaborately in chapter 3.

4.5 Dimensionality Reduction

After a set of features is identified as described in the last section, the set of text documents is transformed into a set of feature vectors with each feature corresponding to the term weight of an indexing term. Thus, the number of features in these feature vectors is equal to the vocabulary size. Due to the inherent properties of textual data, there are usually thousands or even tens of thousands of unique terms in the vocabulary. As each unique term represents a new dimension in the feature space, the set of feature vectors is of very high dimensionality.

Because of the high dimensionality in the feature space, feature vectors are not suitable as input to the text classifier since the scalability will be poor. In order to improve the scalability of the text categorization system, dimensionality
reduction techniques should be employed to reduce the dimensionality of the feature vectors before they are fed as input to the text classifier. There are four dimensionality reduction techniques applicable to the text categorization system. They are (1) Document Frequency method (DF) (2) Category Frequency - Document Frequency method (CF – DF) (3) TF – IDF method (4) Principal Component Analysis. The detailed discussion on these methods has been avoided as they are beyond the scope of this work.

4.6 Neural Network Based Classifier

By applying dimensionality reduction techniques we get a reduced set of feature vectors. This set of reduced feature vectors is then fed to the text classifier as input. In the standard model, a three layer feed-forward network is used as text classifier.

4.6.1 Network Topology of the Text classifier

The following figure shows the topology of the neural network based text classifier.

![Network Topology of the Text classifier](image)

Figure 4.2: The neural network based classifier (Courtesy: Lam Lai Yin, Dominic Savio, 1996)
As shown in the figure 4.2, the neural network is a three-layer fully connected feed-forward network which consists of an input layer, a hidden layer and an output layer. All neurons in the neural network are non-linear units with sigmoid function as the activation function. In the input layer, the number of input units \( r \) is equal to the dimensionality of the reduced feature space. In the output layer, the number of output units \( m \) is equal to the number of pre-defined categories in the particular text categorization task. The number of hidden units in the neural network affects the generalization performance. The choice depends on the size of the training set and the complexity of the classification task the network is trying to learn, and can be found empirically based on the categorization performance.

For classification of the documents, reduced feature vectors representing the documents are fed to the input layer of the neural network classifier as input signal. These input signals are then propagated forward through the neural network so that the output of the neural network is computed in the output layer. As the sigmoid function is used as the activation function in the output units, the output of the neural network classifier is a real-valued classification vector with the component values in the range \([0, 1]\). This real-valued classification vector represents a graded classification decision, in which the \( i^{th} \) vector component indicates the probability that the input document belongs to the \( i^{th} \) category. If binary classification is desired, a threshold can be set such that a document is considered to be belonging to the \( i^{th} \) category only if the \( i^{th} \) component of the classification vector is greater than the threshold.
4.6.2 Training the Text Classifier

The neural network classifier must be trained before it can be used for text categorization. Training of the neural network classifier is done by the Back propagation learning rule based on supervised learning. In order to train neural network, a set of training documents and a specification of the pre-defined categories, to which the documents belong to are required. More precisely, each training example is an input-output pair:

\[ T_i = (D_i, C_i) \]

Where \( D_i \) is a reduced feature vector of the \( i^{th} \) training document, and \( C_i \) is the desired classification vector corresponding to \( D_i \). The component values of \( C_i \) are determined based on the categorization information provided in the training set. During training, the connection weights of the neural network are initialized to some random values. The training examples in the training set are then presented to the neural network classifier in random order, and the connection weights are adjusted according to the Back propagation learning rule. This process is repeated until the learning error falls below a pre-defined tolerance level.

4.7 Revised BPN Algorithm

Back propagation Network algorithm denoted as BPN for the text categorization is discussed here. There are certain basic assumptions incorporated into this algorithm. First, the output function on all the hidden and output layer units is assumed to be the sigmoid function. Moreover, the momentum term in the weight update calculations is included in this algorithm.
The BPN algorithm is produced in two phases here. In the first phase forward signal propagation occurs in the network. In the second phase the error terms are fed back to all other input units. In this case they are the feature vectors. The BPN algorithm is provided below:

**Phase 1:**
1. Locate the first processing feature vector in the layer immediately above the current layer.
2. Set the current input total to zero.
3. Compute the product of the first input connection weight and the output from the transmitting feature vector.
4. Add that product to the cumulative total.
5. Repeat steps 3 and 4 for each input connection.
6. Compute the output value for this unit by applying the output function
   \[ f(x) = \frac{1}{1 + e^{-x}} \text{, where } x = \text{input total.} \]
7. Repeat steps 2 through 6 for each feature vector in this layer.
8. Repeat steps 1 through 7 for each layer in the network.

Once an output value has been calculated for every unit in the network, the values computed for the categories in the output layer are compared to the desired output decision, element by element. At each category in the output, an error value is calculated. These error terms are fed back to all other units in the network.
Phase 2:

1. Locate the first processing unit in the layer immediately below the output layer.
2. Set the current error total to zero.
3. Compute the product of the first output connection weight and the error provided by the unit in the upper layer.
4. Add that product to the cumulative error.
5. Repeat steps 3 and 4 for each output connection.
6. Multiply the cumulative error by \( o(1-o) \), where \( o \) is the output value of the hidden layer unit produced during the feed forward operation.
7. Repeat steps 2 through 6 for each unit of this layer.
8. Repeat steps 1 through 7 for each layer.
9. Locate the first processing unit above the input layer.
10. Compute the weight change value for the first input connection to this unit by adding a fraction of the cumulative error at this unit to the input value of this unit.
11. Modify the weight change term by adding a momentum term equal to a fraction of the weight change value from the previous iteration.
12. Save the new weight change value as the old weight change value for this connection.
13. Change the connection by adding the new connection weight change value for this connection.
14. Repeat steps 10 through 13 for each input connection to this unit.
15. Repeat steps 10 through 14 for each unit in this layer.
16. Repeat steps 10 through 15 for each layer in the network.
There are certain aspects worth mentioning in BPN. The first thing is that BPN is good at generalization. Here generalization means BPN will learn to eliminate significant similarities in the input vectors if the different input feature vectors belonging to same class are given. Note that in the step 11 of the Phase 2 of the algorithm the momentum term is added to the fraction of the weight change. This momentum term is included to reduce the number of iterations and faster convergence in the error rate. Irrelevant data will be ignored. The second thing is that if the output function is sigmoidal, then the output values should be scaled. Because of the sigmoid function, the network outputs can never reach 0 or 1. Therefore use values such as 0.1 and 0.9 to represent the smallest and largest output values.

4.8 Results

Initially in the text categorization the input data is fed into the input layer for training the network. Error rate for training the network is fixed as 0.0001 and training continues till it satisfies the error rate.

After the training is over, the test data from the test set is fed in the network. In the training set three text categories are used. They are Sports, Politics and International news. The label values for each category are given as a range. For Sports category the range is from 0.3 to 0.699. The range for Political category is from 0.7 to 0.899 and the International News has the range beyond 0.9. The network takes 5-9 minutes for learning and the loop exists for 500 to 1000 iterations.

The steps associated with the implementation of BPN algorithm are discussed here as it would throw more light on understanding the implementation details. First to find the number of neurons the reduced feature list provided by LVF algorithm is considered.
In this data set the reduced number of feature words is 78. Hence the neurons count is fixed as 78. Because of this the number of hidden layers is taken as 78.

Once the number of neurons for the network is established, the biases and weighted connections between these neurons should be added. To add a weight each neuron is randomly initialized with a weight in between -0.5 and 0.5.

The second step is to establish a way to transfer the input data from the input neurons to the hidden layer neurons. Each hidden layer is connected to every input neuron and that each individual hidden layer will receive the sum of the weighted input neurons. As a result of applying the bias of this hidden layer neuron it then scales this summation. Thus for each individual neuron the process should proceed according to the formula,

\[ h_{in_j} = h_{bias_j} + \sum_{i=0}^{n-1} X_i \cdot w_{ij} \]

Where \( X_i \) represents the input neuron values and \( w_{ij} \) represents the weight connecting the input neuron to the hidden layer neuron. In other words, the input value to a node is the bias for that node added to the sum of each input interconnection is the input neuron’s value multiplied by the weight of the connection.

After the hidden layer neuron input values have been determined, each hidden layer values are passed to the transfer function. Hidden layer transfer function is sigmoid function that controls the excitation state, or value of the neuron. The sigmoid function is given below:

\[ f(x) = \frac{1}{1+e^{-x}} \]

The output of the hidden layer now need to be passed over to the next layer of weighted connection to the output neuron, although these connections have their own unique set of weights and the output neuron has its unique bias.
After the output layer neuron’s input values have been determined, each output layer’s input values are passed to the transfer function. Output layer transfer functions are sigmoid function that controls the excitation state, or value of the neuron. In the third step, a method is developed for adjusting the weights and biases so that the neural network will be able to accurately predict the category based on the input variables that are presented.

The primary thing in this process is to take an output by the neural network for the given inputs and compare it to a corresponding target value. These target values are the outcomes for the given inputs; they are used as part of the training process so that the network can learn the specific patterns that can be associated with the specific values. An error information term, delta ($\delta$), is then calculated by multiplying the difference between these two terms by the derivative of the activation function. This expression is given below:

$$\delta = \frac{1}{2}(1+\text{trans}(\text{oin})) \times (1-\text{trans}(\text{oin})) \times (\text{targval} - \text{oout})$$

Here trans(oin) refers to the value of the input variable during the training, targval represent the target value and oout refers to the value of the output variable. The error term is then used to compute a weight adjustment term as well as a bias adjustment term. The computation of these terms, however, requires two additional terms to be taken into account. One term is learning alpha ($\alpha$) (learning rate) and the second term is mew ($\mu$), the momentum term. The alpha value is set as 0.3 and the mew value is set as 0.6. The weight and bias update equations including the momentum terms are as follows:

$$w_{jk}(t+1) = w_{jk}(t) + \alpha \times \delta \times \text{hout} + \mu \times (w_{jk}(t) - w_{jk}(t-1)) \quad \ldots(\text{A})$$

$$\text{bias}_k(t+1) = \text{bias}_k(t) + \alpha \times \delta + \mu \times (\text{bias}_k(t) - \text{bias}_k(t-1)) \quad \ldots(\text{B})$$
Here the equation (A) refers to the weight update equation and equation (B) is the bias update equation. Also t refers to the current set of weights/biases, t-1 refers to the previous set and t+1 denotes the new set being calculated.

Now all the weighted interconnections between the output neuron and the hidden layer neurons are updated. Before this update occurs the past weights are stored in an array and this helps to effectively utilize the momentum. Similar updation procedure is carried out for the bias value. After this the error should be propagated to the interconnections between the hidden layer and the input neurons. Finally the error rate is calculated. Till the error rate satisfies the predefined error rate, error training should be continued. It can be recalled that the error rate was earlier set as 0.00001. For training the error and calculating the error rate following equations are used:

\[
\text{Train Error} = \text{Train Error} + (\text{Target Value} - \text{Output Value})^2 \quad \text{...(C)}
\]

\[
\text{Error Rate} = \sqrt{\frac{\text{Train Error}}{\text{Number of Training Documents}}} \quad \text{...(D)}
\]

Equation (C) refers to the equation for the training error and the equation denotes the computation of the error rate.

**4.9 Conclusion**

In this chapter this revised algorithm is discussed with reference to the Text categorization problem hoping this will help in producing a better categorization algorithm. This algorithm along with the other three Text categorization algorithms is compared in the next chapter. When performance of this algorithm is considered individually, it is found to be satisfactory with reference to the dataset used. But the performance of this algorithm depends on so many other factors like the dimensionality reduction. The reduced feature list inputted should be an appropriate list. Also the activation function and the momentum terms play an important role in the performance of the algorithm.