# Text Categorization by Backpropagation Network

S.Ramasundaram
Associate Professor
Dept. of Computer Science
Madurai Kamaraj University College
Madurai, India – 625 002

S.P.Victor
Reader & Head
Dept. of Computer Science
St.Xavier College,
Tirunelveli – 627 002

## ABSTRACT
Text classification gains lot of significance in the current scenario of processing and retrieval of text. Several algorithms are suggested for the text classification problem. This paper provides the solution by Back propagation network. The backpropagation network algorithm is adapted for the text classification. Before providing the algorithm the techniques used for feature identification is also discussed..

## General Terms
Neural Networks, Feature selection, Feature Extraction, Word Extraction, Stopwords, Dimensionality Reduction.

## Keywords
Stoplist, Morphological variants, Training Documents, Affix Removal Stemmers.

## 1. INTRODUCTION
Today huge chunks of information are being associated with the web technology and the internet. Also there is an effort to convert the text available in paper to the digitized form. To gather useful information from it these text has to be categorized. Hence we have to think about the ways and means for categorizing the textual information. By saying categorizing the text, we mean classifying the text or documents in the predefined categories.

This article discusses about categorizing the text by non-linear feed-forward neural network trained by Backpropagation learning rule. That is, we are applying the neural network for classifying the text under supervised learning. The algorithm for applying BPN to the text categorization problem is also provided in this paper.

## 2. Need for Neural Networks
We are trying to find the solution to the Text categorization problem by Backpropagation method, which 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 the 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, Backpropagation is a very popular algorithm. Backpropagation as an ANS is very useful in recognizing complex patterns and performing nontrivial mapping functions. Following figure represents a simple Backpropagation diagram. The rounded objects represent the neurons or processing elements of a neural network. The directed lines that are connecting the neurons are called weights. Also every line of processing elements is a layer of a network. Thus in this figure there are three layers available. Though there can be multiple layers present in ANS generally there will be three layers present in Backpropagation network (BPN)

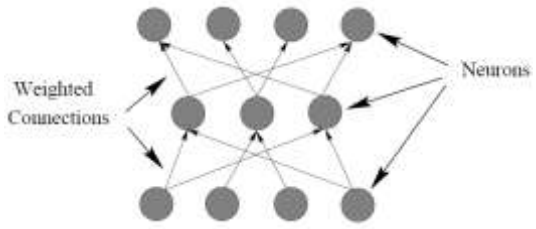


Fig 1 : Backpropagation Network

## 3. Classification Problem
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=< f1,f2,…,fn >

Figure 2 shows a general model of the computer based classification task. In this model, a 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

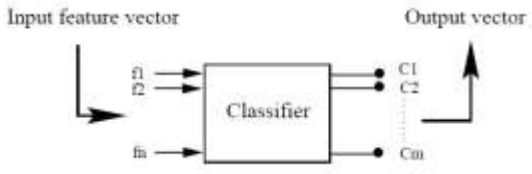


Fig.2 : General classification model

output vector, where m is equal to the number of pre-defined classes in particular classification task. Here 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. Feature Selection and Feature Extraction

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.

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 problem with 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 to 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 a multiple new features, many-to-one operators which transform a multiple features into a single new feature and many-to-many operators which transform a multiple features into multiple features.

Coming back to the text categorization system there are fundamentally three steps 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 features identified initially. Dimensionality reduction will be the second step. As a last step we propose to use three layered feed-forward network trained by Backpropagation as the text classifier. As the learning paradigm for Backpropagation neural networks is supervised learning, the training set should include a set of training documents, together with a specification of predefined categories.

## 5. Techniques used for feature identification

In this section we will be discussing about the techniques that will be useful in identifying the features.

### 5.1 Word Extraction

In a text document we can find a long stream of characters. This long stream of characters should be converted to words or token for purpose of feature identification. In information retrieval, this text tokenization process is called as word extraction, word breaking, word segmentation, or lexical analysis.

Depending on the natural language the document is written in, the word extraction process involve very different techniques. In English language it is very easy because boundaries between words are marked by special delimiting characters such as spaces and punctuations. But for other languages like Japanese, Chinese and Korean it is not so easy. Hence we have to adopt different word extraction techniques for those languages. In designing the word extractor, we must define what constitutes a word in the operational point view. That is, which kind of character sequences should be considered as an English word.

### 5.2 Stop Words Removal

It is well recognized among the information retrieval experts that a set of functional English words (eg. "the", "a", "and", "that") is useless as indexing terms. These words have very low discrimination value, since they occur in every English document. Hence they do not help in distinguishing between documents with contents that are about different topics.

The process of removing the set of non-content-bearing functional words from the set of words produced by word extraction is known as stop words removal. In order to remove the stop words, semi-automatic procedure is followed. This involves first creating a list of stop words to be removed, which is also called the stoplist or a negative dictionary. After this, the set of words produced by word extraction is then scanned so that every word appearing in the stoplist is removed.

The major difficulty in stop words removal is in deciding which words should be put into the stop list. This is difficult as the list of stop words is sometimes dependent on the topics of the given documents. Words that are not normally considered as stop words in general English may be considered as such in specialized document collections. For example, words like "home", "page", "world", "web" which are not considered as stop words in English may be considered in World Wide Web related documents.

### 5.3 Word stemming

In a text document a word may exist in a different morphological variants. For example the word "see" may also exist in other

morphological variants such as "seeing", "seen". While these morphological variants are different word forms, that represent same concept. In text categorization and other similar tasks it is desirable to combine these morphological variants of the same word into one canonical form. In information retrieval, the process of combining or fusing together different morphological variants of the same word into a single canonical form is called stemming. The module performing stemming is called a stemmer.

It is desirable to build stemmers which produce stems based on a set of pre-defined word translation rules. This approach is commonly used in a class of stemmers called affix removal stemmers, in which stem is formed by removing suffixes and/or prefixes from the original word forms.

The simplest kind of affix removal stemmers is one that removes plurals by removing the suffix "s". The problem with this simple stemmer is obvious as not all words ending with an "s" are plurals (e.g. success, various, etc) and some plurals do not end with an "s" (e.g. criteria,formula,etc). To avoid this problem, stemmers use rules that are conditional statements which will be applied only if some conditions are met. For example, a set of conditional rules for plural removal may look like this:

If a word ends in "ies" but not "eies" or "aies"

Then replace the suffix "ies" by "y"

If a word ends in "es" but not "aes", "ees", "oes"

Then replace the suffix "es" with "e"

If a word ends in "s" but not "us" or "ss"

Then remove the suffix "s"

Typical stemmer consist of different sets of rules for different suffix and prefix translations(e.g. "ed" -→ NULL, "ational" -→ "ate",etc). These rules may be divided into a number of steps, where a rule in each step is applied successively for multiple translations.

Stemming may be considered as a mny-to-one operator used for feature extraction. In stemmers discussed above, human knowledge is needed in creating the set of translation used in stemming.

## 5.4 Giving Weights to terms

After word extraction each text document is transformed into a set of stems corresponding to the set of words appearing in each document. Next step is to find the union of all these sets, such that the union set contains the set of stems corresponding to all of the words appearing in the given set of documents. Duplicates are removed so that each stem is unique within the union set. In information retrieval, this set of stems constitutes the set of indexing terms for the set of documents. This set of indexing terms is called indexing vocabulary.

According to the importance of the terms in the documents, measurements must be made for each term in the vocabulary for producing the set of initial features from indexing vocabulary. This involves assigning to each term a weight indicating the relative importance of the term in a document. This process of assigning a weight to each indexing term is commonly known as term weighting in information retrieval. With the set of indexing terms in the indexing vocabulary and their corresponding term weights, a document can be represented as the following feature vector:

$$DJ=< wj1,wj2,\ldots wjk >$$

Where K is the number of indexing terms in the vocabulary, or the vocabulary size, and wji is the term weight of the ith indexing term in document j.

One of the methods used for giving weight to the terms is the Inverse document frequency (IDF). IDF is a factor used for reflecting the discrimination value of each term. A commonly used definition of the IDF is given below:

$$IDFi = \log(N/n)$$

Where IDFi is the inverse document frequency of the ith indexing term, N is the number of documents in the document set, and n is the number of documents in which the ith indexing term appears. By this definition, a term appears in fewer documents will have a higher IDF. The assumption behind this definition is that the terms concentrated in a few documents are more helpful in distinguishing between documents with different topics. For term weighting, the term weight is equal to the product of the term frequency (TF) and the inverse document frequency (IDF):

$$Wji = TFji \times IDFi$$

Here the terms that appear frequently in a few documents are given higher term weights.

# 6. 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 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 have been avoided as they are beyond the scope of this paper.

## 7. 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 proposed model, a three layer feed-forward network is used as text classifier.

### 7.1 Network Topology of the Text classifier

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

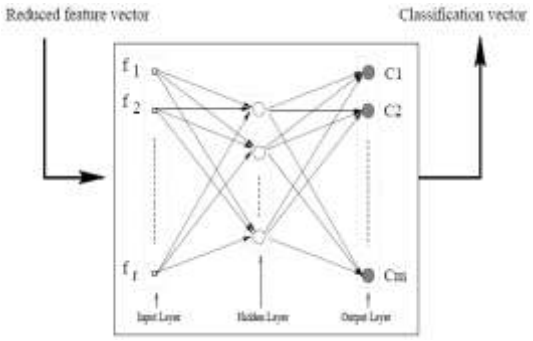


Figure.3 : The neural network based classifier

As shown in the figure, 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.

### 7.2 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 Backpropagation 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 the documents belong to are required. More precisely, each training example is an input-output pair:

$$Ti = (Di, Ci)$$

Where Di is a reduced feature vector of the ith training document, and Ci is the desired classification vector corresponding to Di. The component values of Ci 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 Backpropagation learning rule. This process is repeated until the learning error falls below a pre-defined tolerance level.

## 8. BPN Algorithm

Finally let us discuss about the Backpropagation Network algorithm denoted as BPN. 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, we have included the momentum term in the weight update calculations.

The BPN algorithm is produced in two phases here. In the thtfirst 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. Now the algorithm 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 $f(x)=1/(1+e^{-x})$, where x = 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 to 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 a same class are given. Irrelevant data will be ignored.The second thing is that if the output function is sigmoidal, then we have to scale the output values. 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.

## 9. Conclusion and future work

BPN is a very popular algorithm in the applications pattern matching, character recognition etc., Here this algorithm is discussed with reference to the Text categorization problem. This algorithm is yet to be implemented for this problem. Implementation of this algorithm is considered as part of the future work. Apart from BPN there are other algorithms found in Neural network systems. They are simulated annealing, sequential minimal optimization (SMO) and so on. The suitability of these algorithms to the Text categorization problem should also be analyzed.

## 10. ACKNOWLEDGEMENTS

Our thanks to the experts who have contributed towards development of this article.

## 11. REFERENCES

[1] Lam Lai Yin, Dominic Savio, "Learned Text Categorization by Backpropagation Neural Network", Hong Kong University Thesis, pp.41-59,1996.

[2] James A.Freeman, David M. Skapura, "Neural Networks Algorithms, Applications and programming Techniques, Pearson Education, pp.89-125, 2003

[3] Machine Learning in Automated Text categorization, FABRIZIO SEBASTIANI *Consiglio Nazionale delle Ricerche, Italy,* ACM Computing Surveys, Vol. 34, No. 1, March ,2002.

[4] Automatic Text Categorization: Case Study**,** Renato Fernandes Corrêa, Teresa Bernarda Ludermir , Centro Informática da UFPE, Proceedings of the VII Brazilian Symposium on Neural Networks (SBRN'02),© 2002 IEEE.

[5] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, Tom Mitchel, "Text Classification from Labeled and Unlabeled Documents using EM", Journal on Machine Learning, Kluwer Academic Publishers, 1999.

[6] A.Salappa, M.Doumpos, C.Zounidis, "Feature selection algorithms in classification problems: an experimental evaluation", Journal for Optimization Methods and Software, Vol.22.No.1,February 2007, 199 – 214.

[7] Fuchun Peng, Xiangji Huang, "Machine Learning for Asian language text classification", Journal of Documentation, April,2006

[8] Agustin Casamayor, Daniela Godoy, Marcelo Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach, Elsevier – Information and Software Technology", November 2009

[9] Fabio Aiolli, Ricardo Cardin, Fabrizio Sebastiani, Alessandro Sperduti,"Preferential Text Classification: Learning Algorithms and evaluation measures", Springer – Inf Retrieval 2009.

[10] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, Yond Winter, "Distributional Word Clusters vs. Words for Text Categorization", Journal of Machine Learning Research, 2002