Information Technology Journal 6 (1): 154-159, 2007
ISSN 1812-5638

Classification of Textual Documents Using Learning Vector Quantization

Muhammad Fahad Umer and M. Sikander Hayat Khiyal
Department of Computer Science, International Islamic University, Islamabad, Pakistan

Abstract: The classification of a large collection of texts into predefined set of classes is an enduring research problem. The comparative study of classification algorithms shows that there is a tradeoff between accuracy and complexity of the classification systems. This study evaluates the Learning Vector Quantization (LVQ) network for classifying text documents. In the LVQ method, each class is described by a relatively small number of codebook vectors. These codebook vectors are placed in the feature space such that the decision boundaries are approximated by the nearest neighbor rule. The LVQ require less training examples and are much faster than other classification methods. The experimental results show that the Learning Vector Quantization approach outperforms the k-NN, Rocchio, NB and Decision Tree classifiers and is comparable to SVMs.

Key words: Learning vector quantization, text classification, artificial neural networks

INTRODUCTION

The automated classification has gained invigorated interest in the last decade. The information on the Internet continues to grow at an incredible speed with more than 4.5 billion pages available online. It has become a very challenging task to classify such large collection of information. Text Classification (TC) is one of the prime techniques to deal with the textual data. TC systems are used in a number of applications such as, filtering email messages, classifying customer reviews for large e-commerce sites, web page classification for an internet directory (e.g., Google), evaluating exams paper answers and organizing document databases in semantic categories.

The term classification has been used in a broader context in human activity. We may have a set of observations and want to infer classes or clusters within the set. Or we may have a certain number of classes and we want to classify a new sample into one of the existing classes. The former type is known as Clustering and the latter is known as Classification.

The TC system categorizes the documents into a fixed number of predefined classes. Formally, it can be defined as the task of assigning a Boolean value to each pair (d_i, c_j) where $d_j = \{d_1,d_2,d_3,\ldots d_n\}$ is the set of text documents and $c_i = \{c_1,c_2,c_3,\ldots c_n\}$ is the set of class labels. The value assigned to the pair could be true if the document d_i falls under class c_i or false if the document d_i does not belong to class c_i (Sebastiani, 2002).

The research in automated text classification started in early 1960s. A long list of successes and failures are reported in this field. Many methods had been proposed and a lot of experiments had been carried out. All this research had been done in the field of Information Retrieval and classification was a part of it. The rapid growth of Internet has revived the interest in automated text classification. Hand-built directories of web content suggest one solution to the dilemma, but unfortunately creating and maintaining such directories requires enormous amounts of human effort.

Many classification methods have been suggested in literature such as; Rocchio's classifiers (Rocchio, 1971) and (Joachims, 1997), k-NN (Yang and Liu, 1999), Naïve Bayes (Lewis, 1998), Decision Tree (Dumais, 1998) and Support Vector Machines (Joachims, 1998, 1999).

The k-NN is an example based classifier. For deciding whether a document d belongs to a class c or not, k-NN retrieve the k neighboring documents of d and they vote for the classification; if there is a majority vote for class c, a positive decision is taken and negative otherwise. The success of classification in k-NN depends upon the value of k, but there is no defined way to calculate it. Since k-NN is a lazy classifier, i.e., there is no training stage and all the computation is performed at the classification time, it cannot be used in real-time scenarios to classify large collection of texts.

The Rocchio method (1971) and (Joachims, 1997) selects an average prototype vector for every class. It calculates the similarity between a document and each of prototype vectors and the document is assigned to the class with maximum similarity. A problem with Rocchio classifier discussed by Lam and Ho (1998) is that it restricts the hypothesis space to the set of linear separable hyper plane regions, which has less expressive power than that of k-NN algorithms.

Corresponding Author: Muhammad Fahad Umer, Department of Computer Science, International Islamic University, Islamabad, Pakistan

The Naïve Bayes algorithm (Lewis, 1998) calculates the probability of each class for a document. The document is assigned to the class for which the probability is highest. There are many improvements to the Naïve Bayes classification model. A problem with Naïve Bayes discussed by Shen and Jiang (2003), is that when asked to make predictions; it always gives class posteriors very close to 0 or 1 and smoother class posteriors cannot be determined.

Support Vector Machines are employed in text classification by Joachims (1998, 1999). SVMs are linear classifiers that define a decision surface to separate classes of data as positive and negative. Kernel functions are used for nonlinear separation. SVMs have shown superb performance for text classification tasks and perhaps the best classifiers till now (Yang and Liu, 1999). SVMs' only potential drawback is their training time and memory requirement. For n training instances held in memory, the best-known SVM implementations take time proportional to n^a, where a is typically between 1.8 and 2 (Chakrabart *et al*., 2002).

The Decision Tree (DT) classifiers are based on tree induction algorithms. A DT classifier is a tree, which internal nodes denote the terms and the branches departing from them are labeled with predicate applied to the terms. Each leaf nodes denotes a class. The DT classifiers discussed in literature are based on ID3 (Fuhr *et al*., 1991), C4.5 (Cohen and Hirsh, 1998) and C5 (Li and Jian, 1998).

THE CLASSIFICATION SYSTEM

The automated text classification system is shown in Fig 1. The first phase is the selection of training and test material. This training set is processed through several pre-processes phases such as, the removal of common words, feature selection and word stemming.

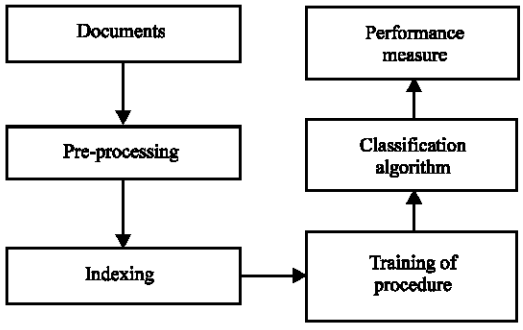

Fig. 1: The classification system

A vocabulary list is constructed containing all of the important terms and this list is used to index the training set. The training set is used for learning the classification. The test set is used to generate the results and the results are analyzed by using some standard performance measures for the evaluation of the classifier.

TEXT SELECTION AND INDEXING

The Reuters-21578 text collection (Lewis, 2006) has been used for evaluation purpose. There are many version of Reuters-21578 available and the ModeApte version was selected. From this collection, the documents having more than 2,000 characters have been selected, as most of the documents in the collection contain only a single line or just news heading. This filtering process made the collection more meaning full and helped us to quickly generate the results from the experimental setup. The final subset contains five categories and the number of document as shown in Table 1.

Normally, text categorization systems use a vector model representation of the documents. The same representation has been used for this system. Each document is represented as a vector and each cell of the vector represents the weighted frequency of the term in that document. There are many different schemes in use for the weighting of the term frequencies. One that seems effective is the Term Frequency-Inverse Document Frequency. That is, the number of times the word appears in the document multiplied by a function of the inverse of the number of documents in which the word appears. Terms that appear often in a document and do not appear in many documents therefore have an important weight.

The creation of vectors for documents consists of following steps:

- A word list containing all of the important terms is created for all the documents.
- This list can be scanned further to remove some common words and to eliminate the most and least frequently occurring terms.
- The document collection is indexed on the basis of word list using normalized TF-IDF.

The creation of a vocabulary list for a collection of documents is not a trivial task. The problems involved in constructing such systems as described by

Table 1: Total number of documents per category

Category	Interest	Ship	Trade	Crude	Wheat	Total
Document	83	81	108	151	56	479

van Rijsbergen (1979) are (1) removal of high frequency or common words, (2) suffix stripping, (3) detecting equivalent stems. Comparing the input text with a stop list of words can easily carry out the removal of high frequency words. This process reduces the size of the dictionary to a considerable limit.

The next problem of suffix stripping is more complicated. Two words should be compared for their conceptual meaning because a conventional string comparison will produce high error rate. The solution to this problem is to remove the suffixes from the different forms of a word. There are standard algorithms defined for suffix stripping called stemming algorithms such as Lovins' algorithm (Andrew, 1971).

Many words, after suffix removal map to one morphological form, but still there are some, which don't. One way to deal with this problem is to have a list of equivalent stems and two words should be considered equivalent if and only if their stems match and there is an entry in the list defining their suffixes as equivalent. This paper does not consider the problem of equivalency of stems as there are very few such words in a document and usually they don't affect the accuracy of the classifier (van Rijsbergen, 1979).

After applying stop list and stemming algorithm, a vocabulary list of 468 words was obtained containing important terms of all documents. We indexed the document collection on the basis of vocabulary list obtaining a two dimensional sparse matrix which contained documents row wise and terms column wise.

NEURAL NETWORKS CLASSIFICATION

A Neural Network (NN) is a network of units called neurons. The neuron is the basic processing element of NN. The inputs to a neuron arrive through synaptic connections. The efficacy of inputs is modeled by the weights attached with every input. The response of the neuron is a nonlinear function of its weighted inputs.

The classification model based on NN has generally, more than one layer of connected neurons. The input layer has the input units representing terms and the output layer output units representing the classes. The intermediate or hidden layers are used for the computing the classification decision. For classifying a document d, its terms t_k with weights w_k are loaded into the input units; the output of these units is propagated through the intermediate neuron layers (if present) to the output layers and the value of the output units determine the classification decision.

A usual way of training NN is back-propagation. When a learning pattern is presented, the activation values of input neurons are propagated through the intermediate layers to the output layers and the actual output is compared with target output, if a miss-match occurs, the error is back-propagated so as to change the parameters of the network to minimize the error.

The simplest type of NNet classifier is the perceptron discussed in Dagan *et al.* (1997) and (Ng *et al.*, 1997) which is a linear classifier. A nonlinear NNet (Lam and Lee, 1999) and (Ruiz and Srinivasan, 1999) is instead a network with one or more additional layers of units, which in TC usually represent higher order interactions between terms that the network is able to learn (Sebastiani, 2002).

LVQ networks are based on the supervised competitive learning. LVQ networks attempt to define decision boundaries in the input space, given a set of exemplary decisions (the training data). Topologically; the network contains an input layer, a competitive layer and an output layer. The output layer has the neurons equal to the number of classes. In the competitive layer, each competitive unit corresponds to a cluster, the center of which is called a codebook vector. The Euclidean distance of an input vector is computed with each codebook vector and the nearest codebook vector is declared winner. Unlike perceptron, LVQ networks can classify any set of input vectors, not just linearly separable sets of input vectors. The only requirement is that the competitive layer must have enough neurons and each class must be assigned enough competitive neurons.

LVQ ALGORITHMS

There are a number of somewhat different LVQ algorithms appearing in the literature, they are all based on the following basic algorithm:

- A learning sample consisting of input vector x_i together with its correct class label c_i is presented to the network.
- A suitable number of codebook vectors are selected for every class label c_i.
- Using distance measures between codebook vectors and input vector d_i, the winner is determined. In some cases, the second best winner is also determined.

We have used LVQ1 (Kohonen, 1990b), LVQ2.1 (Kohonen, 1990a), LVQ3 (Kohonen, 1990b) and the optimized learning rate algorithms OLVQ1 (Kohonen, 1992), OLVQ3 for the classification task.

LVQ1: The LVQ1 (Kohonen, 1990b) selects a single set of best matching codebook vectors is selected and moved closer or further away from each data vector, per iteration if the classification decision is correct or wrong, respectively.

OLVQ1: The Optimized LVQ1 (Kohonen, 1992) is same as LVQ1, except that each codebook vector has its own learning rate

LVQ2.1: Two sets of best matching codebook vectors are selected and only updated if one belongs to the desired class and one does not and the distance ratio is within a defined window. The value of the window is defined as the mid-point of the two codebook vectors.

LVQ3: The same as LVQ2.1 except if both set of codebook vectors are of the correct class; they are updated but adjusted using an epsilon value. The epsilon value is used to adjust the global learning rate.

OLVQ3: The Optimized LVQ3 is same as LVQ3 except each codebook vector has its own learning rate in the same manner as OLVQ1

These training algorithms yield almost similar accuracies, although different techniques underlie each other. The experiment has been performed using all five algorithms and the results are analyzed to determine that which algorithm performs well for text classification. In the next step, the best LVQ algorithm is compared with the other classification algorithms.

THE EXPERIMENTAL RESULTS

Table 2 shows the experimental results of the five LVQ algorithms applied to the document vectors. The parameters for every algorithm have been selected empirically by slightly increasing and decreasing their value and analyzing the output.

Precision and Recall measures are widely used for evaluating the classifiers. Recall is defined to be the ratio of correct assignments by the system divided by the total number of correct assignments. Precision is the ratio of correct assignments by the system divided by the total number of the system's assignments. It is hard to compare classifiers using two measures, the F1 measure, introduced by (van Rijsbergen, 1979), combines recall (r) and precision (p) with an equal weight in the following form:

$$F\beta(r,p) = \frac{(\beta^2+1)pr}{\beta^2(p+r)}, \text{where } \beta = 1$$

The F1-measure has been used for evaluating the accuracy of the classifiers. The n-fold cross validation method was used to obtain the results. This method divides the data set in equal n partitions with using one partition as test set and rest of them as training set. The results show that almost each LVQ algorithm gives similar accuracies. These algorithms also took almost similar time for building the training model. But the optimized-LVQ1 performs well than its counterparts. It gives F1-measure over 90% for wheat and trade categories. The ship category has the lowest F1-measure.i.e., 65.4%. The reason for this is not the poor performance of the classifier but the fact that there were some documents which were common between the ship and crude category and the classification system had to classify the overlapping documents in one of the target classes.

The performance of the OLVQ1 algorithm is also quite reasonable if we consider the training time of the classifier. It took only 922 ms on a 2.4 GHz machine to perform 1600 training iterations on 40 codebook vectors for building a single model for 10-folds cross validation method.

Table 3 compares the results with the other classification algorithms with the same training and test set. Other classification model, such as Naïve Bayes, k-NN and C4 give low accuracy and also their training and classification time is greater than the LVQ. Due to the memory and computation limitations, the results in Table 3 were obtained using 5-folds cross validation method. The results show that for the wheat, trade and interest category, SVMs work well than OLVQ1 and for the ship and crude, the OLVQ1 is better than SVMs. One can say that as SVMs work well for three categories and OLVQ1 is good only for two categories, it is better than OLVQ1. In fact, SVMs are, but the problem lies with the time taken by SVM to build the model. SVMs took 6.25s to build the model while OLVQ1 took only 1.06s that is six times less. This modest training and classification time encourages the use of LVQ for the classification task.

Table 2: F1-measure comparison of LVQ algorithms (10-fold cross validation)

Algorithms/Classes	LVQ1	LVQ2.1	LVQ3	OLVQ1	OLVQ3
Wheat	0.911	0.897	0.925	0.909	0.923
Trade	0.931	0.925	0.936	0.936	0.922
Ship	0.601	0.635	0.632	0.654	0.642
Interest	0.948	0.953	0.940	0.953	0.940
Crude	0.792	0.819	0.781	0.816	0.809

Table 3: F1-measure comparison of classification procedures (5-fold cross validation)

Classifier/ Class	OLVQ1	SVM	RBF Network	SOM	k-NN K = 7	C4	NB
Wheat	0.917	0.927	0.771	0.867	0.867	0.957	0.893
Trade	0.910	0.917	0.759	0.853	0.886	0.762	0.825
Ship	0.593	0.506	0.484	0.577	0.542	0.467	0.547
Interest	0.936	0.959	0.802	0.911	0.909	0.822	0.886
Crude	0.776	0.722	0.641	0.795	0.758	0.817	0.714

FUTURE WORK

This research uses Vector Space Information Model for the representation of text. The problem with Vector Space Information Model is that in the representation, all pairs are considered equally similar. Semantically relationships between the terms are not taken into account (Honkela, 1997). The LVQ classification can be applied by using some other approach such as Poisson distribution. The use of Poisson model is widely investigated in Information Retrieval but it is rarely used for the text classification.

There is no way to determine a good number of codebook vectors. Some mechanism of finding an optimal number of codebook vectors should be embedded in the learning algorithm.

The classification of binary data such as images can be explored with the LVQ.

CONCLUSIONS

This study presents an application of LVQ for text classification. The process of classifying documents with LVQ consists of three phases; pre-processing of data, training of the network and the testing of classification network. The text documents are represented as vectors using Vector Space Information Model. The results generated by the experiment are relatively exceptional. The LVQ network seems to be prospective for the classification of text documents, with the advantage of restricting documents to be a part of certain classes.

This study provides sufficient theoretical base for the development of a fully functional text classification application.

REFERENCES

Andrew, K., 1971. The development of a fast conflation algorithm for English. Dissertation submitted for the Diploma in Computer Science, University of Cambridge, (unpublished).

Chakrabart, S., S. Roy and M. Soundalgekar, 2002. Fast and accurate text classification via multiple linear discriminant projections. Proceedings of the 28th VLDB Conference, Hong Kong, China.

Cohen, W.W. and H. Hirsh, 1998. Joins that generalize: Text classification using WHIRL. In Proceedings of 4th International Conference on Knowledge Discovery and Data Mining, New York, pp: 169-173.

Dagan, I., Y. Karov and D. Roth, 1997. Mistaken driven learning in text categorization. In: Proceedings of 2nd Conference on Empirical Methods in Natural Language Processing, Providence, RI, pp: 55-63.

Dumais, S.T., J. Platt, D. Heckerman and M. Sahami, 1998. Inductive learning algorithms and representations for text categorization. In Proceedings of 7th ACM International Conference on Information and Knowledge Management, Bethesda, MD, pp: 148-155.

Fuhr, N., S. Hartmann, G. Knorz, G. LustiG, M. Schwantner and K. Tzeras, 1991. AIR/X-a rule-based multistage indexing system for large subject fields. In Proceedings of 3rd International Conference Recherche d'Information Assistee par Ordinateur, Barcelona, Spain, pp: 606-623.

Honkela, T., 1997. Self-Organizing-Maps in Natural Language Processing, Ph.D Thesis, Helsinki University of Technology, Finland.

Joachims, T., 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, TN, pp: 143-151.

Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of 10th European Conference on Machine Learning, Chemnitz, Germany, pp: 137-142.

Joachims, T., 1999. Transductive inference for text classification using support vector machines. In Proceedings of 16th International Conference on Machine Learning, Bled, Slovenia, pp: 200-209.

Kohonen, T., 1990a. Improved versions of Learning Vector Quantization. In: proceeding of the National Joint Conference of Neural Networks, San Diego, pp: 545-550

Kohonen, T., 1990b. The self-organizing maps. Proceedings of the IEEE, 78: 1464-1480.

Kohonen, T., 1992. New Developments of Learning Vector Quantization and the Self-Organizing Map. In: Symposium on Neural Networks; Alliances and Perspectives in Senri, Osaka, Japan.

Lam, W. and C.Y. Ho, 1998. Using a generalized instance set for automatic text categorization. In: Proceedings of 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, Australia, pp: 81-89.

Lam, S.L., and D.L. Lee, 1999. Feature reduction for neural network based text categorization. In Proceedings of 6th IEEE Int. Conference on Database Advanced Systems for Advanced Application, Taiwan, pp: 195-202.

Lewis, D.D., 1998. Naïve (Bayes) at forty: The independence assumption in information retrieval. Proceedings of 10th European Conference on Machine Learning, Chemnitz, Germany, pp: 4-15.

Lewis, D.D., 2006. http://www.daviddlewis.com/resources/testcollections/reuters21578/, Last accessed Mar 2006.

Li, Y.H. and A.K. Jian, 1998. Classification of text documents. Computer J., 8: 537-546.

Ng, H.T., W.B. Goh and K.L. Low, 1997. Feature selection, perceptron learning and a usability case study for text categorization. In: Proceedings of 20th ACM Intl. Conference on Research and Development in Information Retrieval, Philadelphia, PA, pp: 67-73.

Rocchio, Jr. J.J., 1971. Relevance Feedback in Information Retrieval. The SMART project Experiments in Automatic Document Processing, Editor: Gerard Salton, Prentice-Hall, Englewood Cliffs, New Jersey.

Ruiz, M.E. and P. Srinivasan, 1999. Hierarchical neural networks for text categorization. In Proceedings of 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, CA, pp: 281-282.

Sebastiani, F., 2002. Machine learning in classification, ACM Computing Surveys, 1: 1-47.

Shen, Y. and J. Jiang, 2003. Improving the performance of Naive Bayes for text classification, technical report (Unpublished), Stanford Natural Language Processing (NLP) Group, Stanford University, CA.

van Rijsbergen, C.J., 1979. Information Retrieval, Butterworth's, London.

Yang, Y. and X. Liu, 1999. A Re-examination of Text Categorization Methods. In: Proceedings of 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, CA, pp: 42-49.