Text Categorization Using Weight Adjusted *k*-Nearest Neighbor Classification *

Eui-Hong (Sam) Han George Karypis, Vipin Kumar

Department of Computer Science and Engineering
Army HPC Research Center
University of Minnesota
4-192 EECS Bldg., 200 Union St. SE
Minneapolis, MN 55455, USA

{han,karypis,kumar}@cs.umn.edu

Abstract

Text categorization is the task of deciding whether a document belongs to a set of prespecified classes of documents. Automatic classification schemes can greatly facilitate the process of categorization. Categorization of documents is challenging, as the number of discriminating words can be very large. Many existing algorithms simply would not work with these many number of features. *k*-nearest neighbor (*k*-NN) classification is an instance-based learning algorithm that has shown to be very effective for a variety of problem domains including documents. The key element of this scheme is the availability of a similarity measure that is capable of identifying neighbors of a particular document. A major drawback of the similarity measure used in *k*-NN is that it uses all features in computing distances. In many document data sets, only smaller number of the total vocabulary may be useful in categorizing documents. A possible approach to overcome this problem is to learn weights for different features (or words in document data sets). In this paper, we propose the Weight Adjusted *k*-Nearest Neighbor (WAKNN) classification algorithm that is based on the *k*-NN classification paradigm. In WAKNN, the weights of features are learned using an iterative algorithm. In the weight adjustment step, the weight of each feature is perturbed in small steps to see if the change improves the classification objective function. The feature with the most improvement in the objective function is identified and the corresponding weight is updated. The feature weights are used in the similarity measure computation such that important features contribute more in the similarity measure. Experiments on several real life document data sets show the promise of WAKNN, as it outperforms the state of the art classification algorithms such as C4.5, RIPPER, Rainbow, PEBLS, and VSM.

Keywords: Classification, text categorization, nearest neighbor classification, weight adjustment, data mining.

*This work was supported by NSF grant ASC-9634719, Army Research Office contract DA/DAAH04-95-1-0538, Cray Research Inc. Fellowship, and IBM partnership award, the content of which does not necessarily reflect the policy of the government, and no official endorsement should be inferred. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute, Cray Research Inc., and NSF grant CDA-9414015. See **http://www.cs.umn.edu/~han** for other related papers.

1 Introduction

Text categorization is the task of deciding whether a document belongs to a set of prespecified classes of documents. Automatic classification schemes can greatly facilitate the process of categorization. For instance, Yahoo! [Yah99] currently uses human experts to categorize the documents. However, given the fast growth in online document data, this would become more difficult with time. At legal publishing houses such as the West Group, human indexers read legal documents and index them manually [CT97]. This step can become a bottleneck in publishing legal documents. Web browser users keep bookmarks to remember sites they are interested in. Often times, they categorize these sites according to their interests such as business, sports, travel, books, and movies. It will be a great help to the users if the automatic text categorization could classify all the searched documents from the web based on the existing bookmarked documents of different user categories.

Text categorization is essentially a classification problem. The words occurring in the document sets become variables or features for the classification problem. A relatively moderate size of document sets could easily have a vocabulary of tens of thousands of distinct words. Many existing algorithms simply would not work with these many number of attributes. Feature selection methods based on document frequency, mutual information, or information gain could be used to reduce the number of words [YP97, Joa97, McC96]. However, if we become too aggressive in reducing the number of words, then we might lose critical information for categorization tasks. Normally, the number of words after feature selection could be still in thousands.

There are several classification schemes that can be potentially used for text categorization. However, many of these existing schemes do not work well in the text categorization task due to the problems mentioned above. For example, widely used classification decision tree induction algorithm like C4.5 [Qui93] or rule induction algorithms such as C4.5rules [Qui93] and RIPPER [Coh95] do not work well when the number of distinguishing features is large. Even though Naive-Bayes classification techniques, such as Rainbow [McC96], are commonly used in text categorization [LG94, LR94, Lew98, MN98], the independence assumption severely limits their applicability.

k-nearest neighbor (k-NN) classification is an instance-based learning algorithm that has shown to be very effective for a variety of problem domains [DH73]. This algorithm has also been used in text classification [Yan94, CH98]. The key element of this scheme is the availability of a similarity measure that is capable of identifying neighbors of a particular document. A major drawback of the similarity measure used in k-NN is that it uses all features in computing distances. In many document data sets, only smaller number of the total vocabulary may be useful in categorizing documents.

A possible approach to overcome this problem is to learn weights for different features (or words in document data sets). In this approach, each feature has a weight associated with it. A higher weight for a feature implies that this feature is more important in the classification task. Note that when the weights are either 0 or 1, this approach becomes the same as the feature selection.

PEBLS [CS93] is a k-NN classification algorithm that uses the Modified Value Difference Metric (MVDM) [CS93] to determine the importance of categorical features. The distance between data points is determined by the MVDM. PEBLS builds distance measure between every possible values of a feature. The distance between values X and Y of a feature is measured according to the class distribution of these values. According to MVDM, the distance between X and Y is small if they occur with the similar relative frequency in many different classes. The distance is large if they occur with the different relative frequency in many different classes. The distance between two data points is calculated by the squared sum of individual feature value distances determined by the MVDM. PEBLS can be used in document data sets by considering each word to be either present or absent in a document. A major problem with PEBLS is that it computes the importance of a feature independent of all the other features. Hence, like the Naive-Bayes classification techniques, it is unable to take into account the interactions among different features.

VSM [Low95] is another k-NN classification algorithm that learns the feature weight using conjugate gradient optimization [She94]. Unlike PEBLS, VSM improves the weight in each iteration according to an optimization function. This algorithm is specifically developed for data with Euclidean distance measure. A potential problem of this approach comes from the fact that the k-nearest neighbor classification problem is not linear (i.e., its optimization function is not a quadratic function). Hence, few of the convergence guarantees of linear conjugate gradient optimization hold for this approach [She94]. Furthermore, the conjugate gradient optimization in this type of problem is not guaranteed to converge to the global minimum when the optimization function has multiple local minima [She94].

In this paper, we propose the Weight Adjusted k-Nearest Neighbor (WAKNN) classification algorithm that is based on the k-NN classification paradigm. In WAKNN, the weights of features are learned using an iterative algorithm. In the weight adjustment step, the weight of each feature is perturbed in small steps to see if the change improves

the classification objective function. The feature with the most improvement in the objective function is identified and the corresponding weight is updated. The feature weights are used in the similarity measure computation such that important features contribute more in the similarity measure. Experiments on several real life document data sets show the promise of WAKNN, as it outperforms the state of the art classification algorithms such as C4.5, RIPPER, Rainbow, PEBLS, and VSM.

2 Weight Adjusted *k*-Nearest Neighbor Classification Algorithm (WAKNN)

In this section, we present the details of WAKNN. For the data representation, we have followed the vector space model commonly used in Information Retrieval systems [Sal89]. In this vector model, each document is a vector and its element corresponds to words in the whole document set. The whole training set can be viewed as a matrix where each row is a document and its columns are words. The values in the matrix can be binary, 1 for presence of the word and 0 for absence of the word. They can also be the within-document word frequency (TF), Inverse Document Frequency (IDF), or TFIDF which is the combination of TF and IDF [Sal89]. Most popularly used schemes in Information Retrieval is TFIDF. However, previous studies in text classification [YC94] and clustering [BGG+99] indicate that TFIDF is not very effective. In WAKNN, we have adopted TF as the entry in the matrix and normalized per row such that each row adds up to 1.0. This step eliminates problems due to the differences in the document size.

For the similarity between documents, several measures are available including 1-norm, 2-norm, and cosine measure [Sal89]. The cosine similarity measure is commonly used in Information Retrieval [Sal89] and hence is adopted as the basic similarity measure in WAKNN. The weighted cosine measure between document X and Y with weight vector W and set of terms (or words) T as

$$cos(X, Y, W) = \frac{\sum_{t \in T}(X_t \times W_t) \times (Y_t \times W_t)}{\sqrt{\sum_{t \in T}(X_t \times W_t)^2} \times \sqrt{\sum_{t \in T}(Y_t \times W_t)^2}}, \quad (1)$$

where X_t and Y_t are normalized TF of word t for X and Y, respectively, and W_t is the weight of word t.

In the weight adjustment step, we are trying to find the optimal weight vector for the classification task at hand. Starting with the initial weight vector, we try to make a small change to the weight vector to see if we improve the objective function related to the classification.

There are two important issues in this step. The first issue is how to evaluate the weight change. In the paradigm of k nearest neighbor classification, the objective function is closely related to class labels of neighbors of each training document. The best case would be when all the neighbors of each training document have the same class labels as the training document. A simple objective function can add up the number of training documents that are correctly classified using their k-nearest neighbors and can be formally defined as:

$$Obj_{simple}(D, W) = |\{d | d \in D \text{ and } \text{Correct}_{simple}(d, D, W)\}| \quad (2)$$

where D is the training document matrix, W is the weight vector, and predicate $\text{Correct}_{simple}(d, D, W)$ is true if the sum of similarities of d's *true* neighbors is greater than the sum of similarities of neighbors of any other single class. The *true* neighbors of d are the neighbors with the same class as d. This objective function tends to regard a training document d to be *correctly* classified even if it has only one neighbor with the same class label. For example, when $k = 5$, the number of classes is 5, and d has one neighbor from each of these classes, d is considered to be correctly classified if the similarity to d's *true* neighbor is the highest among all the similarities.

In WAKNN, we use an objective function that considers a document d to be correctly classified only if the sum of similarities of d's *true* neighbors is at least p percentage of the total similarity sum. This objective function is formally defined as:

$$Obj_{maj}(D, W, p) = |\{d | d \in D \text{ and } \text{Correct}_{maj}(d, D, W, p)\}| \quad (3)$$

where the predicate $\text{Correct}_{maj}(d, D, W)$ is true if $\text{Correct}_{simple}(d, D, W)$ is true and the sum of similarities of d's *true* neighbors is at least p percentage of the total similarity sum. We call p as the majority percentage. In the same example discussed earlier, when the objective function of Equation 2 with $p = 50$ is used, the training sample will be considered to be correctly classified only if the similarity of d's *true* neighbor is at lest 50% of the total similarity sum.

The second issue is how to propose possible changes to the weight. For the proposal of the possible changes, we adopted to change one word weight at a time. For each word weight, we multiply the current word weight with

different multiplication factors, {0.2, 0.8, 1.5, 2.0, 4.0}. For each of these changes, we evaluate if this change improves the objective function. We remember the best change for each word. We pick the best word and its weight that gives the best value according to the objective function. We update the weight vector with this value and continue to the next round.

The major steps of WAKNN can be summarized as follows:

1. Construct training matrix, D, where each row correspond to a training document, each column represent a word, and value in the matrix $D(i, j)$ corresponds to the number of occurrences of word j in document i.
2. Normalize word frequencies in each document such that they add up to 1.0.
3. Initialize weight vector W.
4. Determine k nearest neighbors for each training document using the weighted cosine similarity measure of Equation 1 with this initial weight vector and calculate the goodness of this initial weight vector using an objective function, $Obj_{maj}(D, W, p)$ of Equation 3.
5. While there is an improvement in $Obj_{maj}(D, W, p)$, repeat the following steps:
   (a) For each word i, determine the value of W_i that gives the best $Obj_{maj}(D, W, p)$. New possible values for W_i are proposed by multiplying the original W_i with different multiplication factors.
   (b) Select a word j that gives the best overall $Obj_{maj}(D, W, p)$ from the previous step, and update W_j with this new value.

In classifying a test document, we first construct a test vector according to the steps 1 and 2 of WAKNN. We find k nearest neighbors of the test document from the training documents using the weighted cosine similarity measure with the weight learned from WAKNN. We then sum up the similarities to the k neighbors according to their class labels. We classify the test document according to the class with the most similarity sum.

3 Experimental Results

In this section, we compare WAKNN to k-NN, C4.5 [Qui93], RIPPER [Coh95], PEBLS [CS93], Rainbow [McC96], VSM [Low95] on several synthetic and real data sets. Note that we used publicly available codes of C4.5, RIPPER, PEBLS, Rainbow, and VSM for these experiments.

There are two parameters of importance in WAKNN. The first is the choice of number of neighbors (k) and the second is the majority percentage (p) in the Equation 3. We have performed experiments of changing k from 1 to 50 and p from 0% to 90%. The results (not reported here due to the space constraint) show that the classification accuracy do not vary significantly when k is between 5 and 30, and when the majority percentage is from 30% to 70%. In the subsequent experiments, we have chosen k to be 10 and the majority percentage to be 50%.

For C4.5 and RIPPER, we have a choice of either regarding each word as a discrete variable or continuous variable. The results reported here are obtained by regarding each word as a discrete variable. The results from experiments where each word is regarded as a continuous variable did not differ from the reported results. For PEBLS, there is an option to weight instances differently. We tried this option, but the result was substantially worse than results reported here, which are obtained without this option. We also varied the number of neighbors from 1 to 30, and found that 1-nearest neighbor gave the best result. Hence the result shown here is based on 1-nearest neighbor. For VSM, we have tried number of iteration from 10 to 100 for the conjugate gradient optimization. The results were not stable, as the results obtained from more number of iterations are sometimes worse than the results obtained from the fewer iterations. We report the results of VSM when the total number of iterations is 50. The number of neighbors used is 10, which is the same as the one used in WAKNN.

The summary of document data sets used in the experiments is shown in Table 1. In all of the data sets, we have used stop words and stemming using Porter's suffix-stripping algorithm [Por80]. First 7 data sets are from the statutory collections of the legal document publishing division of West Group described in [CT97]. Out of 149,655 collections of documents, we selected 7 subsets of documents that have single label. We then randomly selected training sets and test sets. Some of the examples of class labels of these documents include "counties", "sales", "worker's compensation", and "insurance".

Data set $fbis$ is from the Foreign Broadcast Information Service data of TREC-5 [TRE99a]. The class labels were generated from the relevance judgment provided by TREC-5 routing query relevance "qrels.1-243" [TRE99b]. We

	Source	# train	# test	# class	# words used
west-1	West Group	500	1500	10	977
west-2	West Group	300	900	10	1078
west-3	West Group	488	245	10	1035
west-4	West Group	559	280	10	887
west-5	West Group	621	311	10	1156
west-6	West Group	732	367	10	789
west-7	West Group	885	433	10	779
fbis	TREC-5	2463	1232	17	2000
trec6	TREC-5	1173	587	14	2000
reuters	Reuters-21578	6552	2581	59	2000

Table 1: Summary of data sets used.

	C4.5	RIPPER	PEBLS	VSM	Rainbow	*k*-NN	WAKNN
west-1	85.50	84.47	78.50	85.20	84.40	76.73	**89.60**
west-2	71.30	68.33	67.80	77.44	72.11	68.33	**80.44**
west-3	79.60	75.92	72.70	86.53	80.00	70.61	**88.16**
west-4	81.80	77.14	78.60	87.86	**88.57**	73.93	85.00
west-5	84.60	89.71	86.80	89.71	85.21	84.57	**95.18**
west-6	83.70	83.38	79.80	87.19	85.29	73.57	**88.92**
west-7	80.10	80.14	71.80	83.52	81.26	74.94	**84.42**
fbis	57.10	73.94	69.80	76.14	76.38	78.49	**81.09**
trec-6	67.50	80.58	84.30	87.56	92.16	91.99	**92.67**
reuters	84.50	85.59	84.60	87.68	**91.04**	90.62	90.04

Table 2: Classification accuracies of different classifiers. Note that the highest accuracy for each data set is highlighted with bold font.

collected documents that have relevance judgment and selected documents that have single relevance judgment. Data set *trec*6 is from the Foreign Broadcast Information Service and LA Times data of TREC-5. The class labels are generated similarly to fbis data set using TREC-6 ad hoc query relevance "qrels.trec6.adhoc" [TRE99b]. For these two data sets, we further filtered the words using mutual information [CT91, YP97]. We selected top 2000 words according to the mutual information of the training set. We then randomly selected training sets and test sets for these 2 data sets.

Data set *reuters* is from Reuters-21578 text categorization test collection Distribution 1.0 [Lew99]. We split the documents into training and test set according to the modified Lewis split and selected documents with single label. We also selected top 2000 words according to the mutual information of the training set.

Table 2 shows the comparison of different classifiers on 10 data sets. C4.5 and RIPPER have significantly lower classification accuracies compared to VSM, Rainbow, and WAKNN in most of the data sets. This is due to the large number of features (or words) in these document data sets. Rainbow performed much better than C4.5 and RIPPER, which conforms to the earlier reports [LG94, LR94, Lew98, MN98] showing that Naive-Bayes classification techniques are effective in text categorization. Out of *k*-NN classification techniques, PEBLS and simple *k*-NN did not work very well for these data sets. VSM has much better accuracies compared to PEBLS and *k*-NN, and has a comparable result to that of Rainbow. This result shows that weight adjustment improves classification accuracies in these document data sets. WAKNN has the best result in 8 out of the 10 data sets. Compared to Rainbow, WAKNN is better in 8 data sets and is significantly better in 5 data sets. Compared to VSM, WAKNN is better in 9 data sets and significantly better in 5 data sets. This result shows that the weight adjustment in WAKNN performs better than VSM's weight adjustment based on the conjugate gradient optimization in these data sets.

In traditional Information Retrieval systems, given multiple classes in the document set, binary classification for individual class is considered. The effectiveness of retrieval is measured in terms of recall, precision, and F-measure for each class label separately [RL94]. In these experiments (not reported here), we have confirmed that WAKNN outperforms other classifiers in terms of the micro-averaged F-measure [RL94].

4 Conclusions and Directions of Future Research

In this paper, we presented a k-nearest neighbor classification algorithm that learns importance of attributes and utilizes them in the similarity measure. As our experimental results have shown, our algorithm is very effective in the text categorization task.

However, a number of key issues remain to be addressed. One issue is how to avoid the local minima in the search for the best weight vector. In addition to the main objective function used in WAKNN, we might need secondary objective functions to move out of the local minima. Another possible solution to the local minima problem might be changing weights of multiple words at a time. A big challenge for this solution is how to identify the set of words for the weight change.

Another issue is whether the enhanced weight adjustment leads to overfitting. Even though the relatively large number of neighbors and high majority percentage in the objective function tends to reduce the risk of overfitting, this problem can be significant in many data sets.

Finally, the computational cost of weight adjustment step of WAKNN is $O(cn^2)$ where c is the number of iterations in the weight adjustment step and n is the number of data points. Optimization schemes to improve this computational complexity is needed.

Acknowledgment

We would like to thank Paul Thompson and Isabell Moulinier of West Group for allowing us to have the data for experiments. We would also like to thank David Lowe for the VSM code.

References

[BGG+99] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems (accepted for publication)*, 1999.

[CH98] W.W. Cohen and H. Hirsh. Joins that generalize: Text classification using WHIRL. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, 1998.

[Coh95] W.W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, 1995.

[CS93] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[CT97] T. Curran and P. Thompson. Automatic categorization of statute documents. In *Proc. of the 8th ASIS SIG/CR Classification Research Workshop*, Tucson, Arizona, 1997.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[Joa97] T. Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.

[Lew98] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Tenth European Conference on Machine Learning*, 1998.

[Lew99] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/ lewis*, 1999.

[LG94] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR-94*, 1994.

[Low95] D.G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, pages 72–85, January 1995.

[LR94] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[McC96] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[RL94] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, 12(3), 1994.

[Sal89] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[She94] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/∼quake-papers/painless-conjugate-gradient-figs.ps, 1994.

[TRE99a] TREC. Text REtrieval conference. *http://trec.nist.gov*, 1999.

[TRE99b] TREC. Text REtrieval conference relevance judgements. *ftp://ftp-nlpir.nist.gov/pub/trec/qrels*, 1999.

[Yah99] Yahoo! Yahoo! http://www.yahoo.com, 1999.

[Yan94] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *SIGIR-94*, 1994.

[YC94] Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3), 1994.

[YP97] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.