An Improvement of Centroid-Based Classification Algorithm for Text Classification

Zehra Cataltepe, Eser Aygun
Istanbul Technical Un.
Computer Engineering Dept.
Ayazaga, Sariyer, Istanbul, Turkey
cataltepe@itu.edu.tr, eser.aygun@gmail.com

Abstract

k-nearest neighbor and centroid-based classification algorithms are frequently used in text classification due to their simplicity and performance. While k-nearest neighbor algorithm usually performs well in terms of accuracy, it is slow in recognition phase. Because the distances/similarities between the new data point to be recognized and all the training data need to be computed. On the other hand, centroid-based classification algorithms are very fast, because only as many distance/similarity computations as the number of centroids (i.e. classes) needs to be done. In this paper, we evaluate the performance of centroid-based classification algorithm and compare it to nearest mean and nearest neighbor algorithms on 9 data sets. We propose and evaluate an improvement on centroid-based classification algorithm. Proposed algorithm starts from the centroids of each class and increases the weight of misclassified training data points on the centroid computation until the validation error starts increasing. The weight increase is done based on the training confusion matrix entries for misclassified points. The proposed algorithm results in smaller test error than centroid-based classification algorithm in 7 out of 9 data sets. It is also better than 10-nearest neighbor algorithm in 8 out of 9 data sets.

We also evaluate different similarity metrics together with centroid and nearest neighbor algorithms. We find out that, when Euclidean distance is turned into a similarity measure using division as opposed to exponentiation, Euclidean-based similarity can perform almost as good as cosine similarity.

1 Introduction

As the size of the internet grows, ability to automatically cluster or classify documents, accurately and fast is becoming more and more important. While in some cases there is no or small number of labeled documents, in some other cases there is a set of labeled documents. The first set of problems is solved using (unsupervised) clustering techniques [7], while the second type of problems are solved using (supervised) classification techniques [11]. In this study, we focus on labeled problems and hence classification methods.

One of the most popular and simple text classification methods is the k-nearest neighbor classification [1]. Although it has been shown to perform quite well, comparable to even SVMs [3], k-nearest neighbor algorithm gets slower as the number of documents in the training set increases. There have been attempts to make k-nearest neighbor method faster, for example by obtaining approximate solutions [6, 10] or by centroid-based methods [5, 8]. In this study, we evaluate improvements on centroid-based classification algorithms.

Traditionally, the feature vectors for the classification task are obtained using stop word removal, stemming and tf-idf (Term Frequency-Inverse Document Frequency) computation. The feature vector for each document is very high dimensional (total number of words in the corpus) and inherently sparse. The classification or clustering algorithms depend on a measure of similarity/distance between documents. It has been shown in [12] that, for document clustering, cosine, Pearson and extended Jaccard similarities perform a lot better than an Euclidean-distance based similarity. In this study, we evaluate the performances of algorithms using different similarity metrics and find out that the Euclidean-distance based similarity could perform well when a proper transformation to similarity is used.

The rest of the paper is organized as follows: In Section 2 we describe the datasets we used. Section 3 describes the similarity metrics we used and also the classification algorithms used. A new classification algorithm is also introduced in this section. In Section 4 we discuss the results of our experiments. The paper closes with the conclusions in Section 5.

2 Data

We used yahoo news and classic3 [2] data sets and data sets available from Karypis Lab (http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/-datasets.tar.gz) for performance comparison of different algorithms. For all data sets, first stop words were removed from documents. Then Porters stemming algorithm [9] was used to compute roots of words. Following [12]s suggestion, words whose frequencies in a document were outside [0.01-0.1] range were removed and the remaining words were used for tf-idf computation. Table 1 gives a summary of all data sets we used.

3 Similarity Metrics and Classification Algorithms

Let x and y be the tf-idf weight vectors of two documents. We consider the following similarity metrics [12] as a measure of how close x and y are:

- Cosine similarity:

$$s_c(x, y) = \frac{x^t y}{||x||||y||} \tag{1}$$

- Pearson Correlation ($\bar{x}$ denotes the average of x over all feature dimensions):

$$s_p(x, y) = 0.5(\frac{(x - \bar{x})^t (y - \bar{y})}{||x - \bar{x}||||y - \bar{y}||} + 1) \tag{2}$$

- Euclidean-based similarity computed by an exponential transformation

$$s_{e1}(x, y) = e^{-||x-y||^2} \tag{3}$$

- Euclidean-based similarity computed by division:

$$s_{e2}(x, y) = \frac{1}{1 + ||x - y||^2} \tag{4}$$

We use the following classification algorithms:

- **[kmeans]**: Classification using the centers determined by k-means clustering algorithm: The k-means algorithm starts with initial random centers chosen from the data set. For each data point the closest center is found, and then the centers are recomputed using data points clustered around the same center. The algorithm terminates when the center memberships dont change anymore. Although k-means is actually a clustering algorithm, it can be used as a preprocessing step for classification, especially when the number of labeled data points is much less than the number of training data points. We use k-means for classification as follows: We determine the label of each center as the majority label among the closest training data points to that center. A test point is labeled with the label of center closest to it.

- **[1nn]**: 1-nearest neighbor classification: The training data and their labels are stored. When a new point is tested, the closest point is found from the training set, the label of that closest point is assigned to be the class of the test point.

- **[10nn]**: 10-nearest neighbor classification: Same as 1-nearest neighbor, except instead of 1, membership of 10 points from the training set determine the class membership of a new test data point.

- **[Centroid]**: Centroid-based classification: Compute one centroid per class, as the mean of the training data points in that class. A new test point is classified according to its similarity to the centroids.

- **[CentroidW]**: Centroid-based classification using a weighted mean of data points as the centroid point: Compute one centroid per class as the weighted mean of the training data points in that class. A new test point is classified according to its similarity to the centroids. We first start with equal weights for each data point around a centroid. Let the training inputs X be partitioned according to the class labels as $X = \bigcup_{k=1}^{K} X_k$ where $X_k = \{x_1^k, x_2^k, \ldots, x_{N_k}^k\}$ denotes the inputs that belong to class k. In CentroidW classification scheme, we compute the centroid for class k, μ_k, as:

$$\mu_k = sum_{i=1}^{N_k} w_i^k * x_i^k \tag{5}$$

Since the algorithm starts with the normal centroid, initially, $w_i^k = \frac{1}{N_k}$ for all i and k.

We classify training data in all classes according to the closest centroid to them and compute the confusion matrix on the training error. Let j be the index of a misclassified training data point from class k and let $m_j^k \in 1, \ldots, K$ be the (mis)predicted class. We increase the weight w_j^k according to:

$$w_j^k = w_j^k + 1 + \frac{M(k, m_j^k)}{sum_{k,k'=1,k \neq k'}^{K} M(k, k')} \tag{6}$$

where $M(k, k')$ denotes the confusion matrix entry between classes k and k'. We set aside a validation set

(10% of training data) and compute the validation error. We use the weights at the minimum validation error as the weights for each training data point.

4 Results

We evaluate test performance of each algorithm as follows: We randomly partition the available data set into 90% training and 10% test set. We make sure each class has representation proportional to their numbers in training data. We measure the classification error (which is 1 - accuracy) on the test set for each method. We report the average test error over 10 different train/test partitions for each algorithm below.

We have first examined the similarity metrics we described above and found out that cosine, Jaccard similarities and Pearson correlation show very similar behavior, while Euclidean-based similarities are usually worse than these three similarities. We have also found out that Euclidean-based similaritys performance depends on the transformation used to turn the distance to a similarity. Table 2 shows these results in more detail.

As seen in Table 3 centroid-based classification used with cosine similarity performs better than 10-nn in 8 out of 9 data sets that we experimented on. Considering the fact that centroid-based classification is much faster and requires much little memory compared to k-nearest neighbor, we believe that this is very good reason to use centroid-based classification as opposed to k-nearest neighbor. We note that computing the optimal value of k using cross-validation could result in smaller test errors for k-nearest-neighbor method. We should also confirm previous observation by other authors (see for example [3]) that large values of k perform usually better than k=1 when using k-nearest neighbor for text classification.

According to Table 3 and Figure 1, for 6 out of 9 data sets, CentroidW results in better test error than centroid-based classification.

We should note that, although we tried using medoid (i.e. the training data point in class i whose similarity to all other training data points in class i is maximum), resulted in worse performance than using centroid. We think that this is due to the fact that the tf-idf vectors are very sparse and the mean vector has a better chance of being close to most of the training data points than medoid. Another method we thought made sense, tried and did not work is using a robuster centroid by not considering $alpha$¿0 percent of training data points that are farthest away from the centroid ($alpha$-trimmed mean). Giving weight to a training data point proportional to its similarity to the centroid did not result in any improvements either. These results show that due to the sparsity and high dimensionality of tf-idf weights, methods that could have worked in some other pattern recognition problem may not necessarily work in text classification.

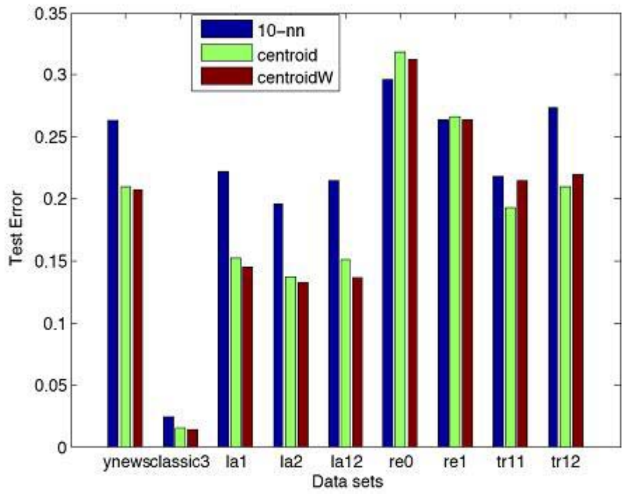

Figure 1. Test error of each method on different data sets.

We evaluated the performance of centroid-based classification and other methods. We did our evaluation based on the mean ($\bar{D}$) and standard deviation (S) of the paired error differences $\bar{D}$ of nearest-mean, 1-nn, 10-nn algorithms and the centroid based classification algorithm [4]. The t-statistic value $T_0 = \frac{\bar{D}\sqrt{9}}{S}$ for these algorithms were $8.0, 3.6$ and 3.1 respectively, which means that with 95% confidence, the centroid algorithm performs better than nearest-mean, 1-nn and 10-nn algorithms. The same statistic for the weighted centroid algorithm is 0.22 which means the weighted centroid-based algorithm is not statistically significantly better than the centroid-based classification algorithm when all 9 datasets are considered. Please note that when the tr11 and tr12 datasets are not considered, the weighted centroid algorithm is better than the centroid algorithm with 95% confidence.

5 Conclusions

We have examined centroid-based classification, k-nearest neighbor classification and an proposed an improvement centroid based classification. We have also evaluated performance of different similarity metrics.

Acknowledgements

Zehra Cataltepe was supported partially by Tubitak EEEAG project 105E162. Authors would like to thank

Deniz Isik, Utku Ozbek and Onur Dolu of Istanbul Technical University Computer Engineering Department for their help with processing of Classic3 data set and Karypis Lab for making most of the other data sets in the paper available.

References

[1] L. Baoli, L. Qin, and Y. Shiwen. An adaptive k-nearest neighbor text categorization strategy. *ACM Transactions on Asian Language Information Processing*, 3(4):215–226, December 2004.

[2] D. Boley, M. Gini, R. Gross, E. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27:329–341, 1999.

[3] A. Cardoso-Cachopo and A. Oliveira. An empirical comparison of text categorization methods. In *String Processing and Information Retrieval: 10th International Symposium, SPIRE 2003, Manaus, Brazil, October 8-10*, pages 183–196, 2003.

[4] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:18951924, 1998.

[5] E. Han and G. Karypis. Centroid-based document classification: Analysis & experimental results. In *4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 424–431, 2000.

[6] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *13th annual ACM symposium on Theory of computing, Dallas, Texas, United States*, pages 604–613, 1998.

[7] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[8] V. Lertnattee and T. Theeramunkong. Effect of term distributions on centroid-based text categorization. *Information SciencesInformatics and Computer Science: An International Journal, Special issue: Informatics and computer science intelligent systems applications*, 158:89–115, January 2004.

[9] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. http://www.tartarus.org/

[10] V. Ramasubramanian and K. Paliwal. An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation. *Pattern Recognition Letters*, 13:471–480, 1992.

[11] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[12] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin, TX*, 2000.

Data set	Source	No of docs	No of orig. words	No of filtered words	No of classes	Class distribution
ynews	Yahoo news	2340	21839	2903	20	142 9 24 44 74 278 70 21 14 125 65 248 158 18 187 54 494 114 141 60
classic3	Boley et. al.	3891	15530	1146	3	1460 1398 1033
la1	TREC	3204	31472	3236	6	555 341 273 943 738 354
la2	TREC	3075	31472	3329	6	905 248 301 759 487 375
la12	TREC	6279	31472	3273	6	1042 642 521 1848 1497 729
re0	Reuters	1504	2886	1007	13	16 608 319 42 60 21 80 20 37 39 11 38 15
re1	Reuters	1657	3758	1313	25	48 371 60 19 37 99 106 137 17 20 330 27 15 31 50 87 18 42 31 32 10 13 18 20 19
tr11	TREC	414	6429	3726	9	52 132 69 21 20 74 11 29 6
tr12	TREC	313	5804	3557	9	30 34 35 29 93 54 29 9

Table 1. Data sets used.

Data set	Similarity metric	k-means	1-nn	10-nn	Centroid
classic3	s_c	0.02±0.01	0.04±0.01	0.02±0.01	0.02 0.01
classic3	s_p	0.02±0.01	0.05±0.01	0.02±0.01	0.02±0.01
classic3	s_{e1}	0.64±0.02	0.65±0.02	0.65±0.03	0.67±0.02
classic3	s_{e2}	0.48±0.13	0.20±0.02	0.39±0.02	0.01±0.01
ynews	s_c	0.46±0.04	0.33±0.02	0.27±0.02	0.23±0.02
ynews	s_p	0.45±0.04	0.33±0.02	0.28±0.02	0.24±0.02
ynews	s_{e1}	0.80±0.01	0.89±0.02	0.87±0.02	0.94±0.02
ynews	s_{e2}	0.75±0.02	0.52±0.03	0.80±0.02	0.32±0.03

Table 2. Test error of different similarity metrics on Classic3 and Yahoo News data sets.

Data set	kmeans	1-nn	10-nn	Centroid	CentroidW
ynews	45.8±1.1	31.4±0.7	26.4±1.0	20.9±0.8	20.8±0.7
Classic3	4.9 ±2.5	4.5±0.2	2.4±0.1	1.5±0.2	1.4±0.2
la1	43.3±2.2	26.7±0.6	22.1±0.6	15.2±0.5	14.4±0.8
la2	38.5±1.6	24.8±0.7	19.6±0.6	13.8±0.3	13.3±0.4
la12	40.3±2.0	25.2±0.6	21.4±0.4	15.1±0.4	13.6±0.4
re0	50.4±1.2	28.8±0.8	29.6±0.7	31.8±1.2	31.2±1.2
re1	49.8±1.8	25.4±1.3	26.4±1.0	26.6±1.3	26.4±1.2
tr11	47.6±3.1	28.3±2.3	21.7±2.1	19.3±2.6	21.5±2.4
tr12	56.5±3.6	29.0±2.1	27.4±2.8	21.0±2.6	21.9±2.5

Table 3. Test error of each classification method on different data sets.