ISSN:0975-9646

An Incremental Feature Clustering Algorithm for Text Classification

Johny Thomas[#1], Abishek Nair[#2], Arpit Gupta[#3]

[#1,#2,#3] *UG Students,*
Department of Computer Science and Engineering,
SRM University, Chennai, India

Abstract— **Text classification is a challenging task due to the large dimensionality of the feature vector. To alleviate this problem, feature reduction techniques are applied for reducing the amount of time and complexity for text classification. In this paper, we propose a novel fuzzy self constructing algorithm for feature clustering. Feature clustering is a feature reduction method which drastically reduces the dimensionality of feature vectors for text classification. Here, words are grouped into clusters based on degree of similarity. Each cluster is characterized by a membership function with statistical mean and deviation. When all the words are fed in,** words similar to a cluster are **grouped into the same cluster otherwise new clusters are created. The derived feature vectors describe properly the real distribution of the training data. The user need not specify the number of extracted features in advance.**

Keywords— **Feature Clustering, Clustering Algorithm, Text Classification.**

I. INTRODUCTION

The dimensionality of a feature vector is huge in text classification. This leads to inefficient text classification process. Feature reduction techniques are employed to reduce the number of features for efficient time complexity. Feature reduction can be done by either feature extraction or feature selecting. In feature selection, the most important features are selected and the rest are neglected. These important features have the ability to best represent a document. In feature extraction, new features are created which are more effective than the original feature. Both these approaches are computationally intensive.

Classical feature extraction methods convert the representation of the original high-dimensional data set into a lower-dimensional data set by a projecting process through algebraic transformations. For example, Principal Component Analysis [1], Linear Discriminant Analysis [2], Maximum Margin Criterion [3], and Orthogonal Centroid algorithm [4] perform the projection by linear transformations, while Locally Linear Embedding [5], ISOMAP [6], and Laplacian Eigenmaps [7] do feature extraction by nonlinear transformations. However, the complexity of these approaches is still high.

Feature clustering is an efficient form of feature reduction where the idea is to group similar features into clusters. Each cluster then produces a single new feature which represents all the features in that cluster. This way, the dimensionality of the features can be drastically reduced but at the same time, it maintains high document classification accuracy, which is essential for text classification. Other feature clustering algorithms use hard clustering and also, the statistical measures are not considered when computing similarity with respect to a cluster. Furthermore, these methods require the number of new features be specified in advance.

In this paper, we propose a fuzzy incremental feature clustering algorithm to reduce the number of features for text classification. In a fuzzy approach, rather than a feature belonging strictly to a single cluster, it may belong to multiple clusters and have multiple degrees of relationship with each cluster. Fuzzy clustering is a class of cluster analysis algorithms in which the allocation of points to clusters is not hard but fuzzy, i.e., data elements can belong to more than one cluster and each data element has a membership level which indicate the strength of association between the data element and the cluster.

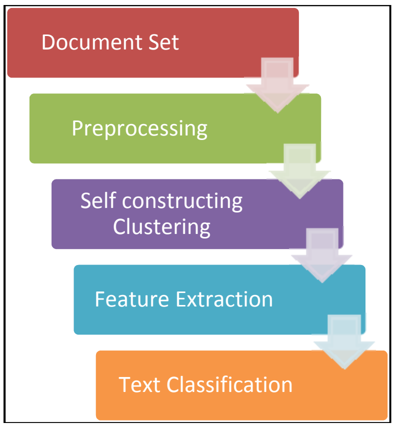
The main applications of text classification are:

- Email classification and spam filtering: Identifying whether an e-mail contains legitimate content or is a spam. Mail can also be categorized into Social, Promotion etc. based on classification techniques.
- Feedback Mining: Identifying whether the intent of a customer's feedback on a product automatically using sentiment analysis.
- News: News articles are generated frequently and hence, automatic classification are required to categorize them.

II. RELATED WORK

Some techniques previously applied to reduce features by clustering them.
Al-Mubaid and Umair [8] used distributional clustering to generate an efficient representation of documents and applied a learning logic approach for training text classifiers. The divisive information-theoretic feature clustering algorithm was proposed by Dhillon et al. [9], which is an information-theoretic feature clustering approach, and is more effective than other feature clustering methods. In these feature clustering methods, each new feature is generated by combining a subset of the original words.

III. Our Method



In the existing techniques, there were many issues. First, the parameter k which indicates the number of extracted features must be manually supplied by the user which is tedious. The user can find this value only by trial-and-error. Second, when calculating similarity, the variance of the cluster is not taken into consideration. Third, all the words in a cluster have the same degree of relationship to the extracted feature.

Our feature clustering algorithm is proposed to deal with these issues.

A. Document Representation

The documents are represented in a bag-of-words model in which the entire string is tokenized and the collection of tokens represent a document.

Let **D = {d_1, d_2, ..., d_n}** be a document set of n documents, where d_1, d_2..., d_n are individual documents and each document belongs to one of the classes in the set {c_1, c_2, ..., c_p}

Let the word set **W = {w_1, w_2, ..., w_m}** be the feature vector of the document set.

Each document d_i, 1<= i <= n, is represented as d_i = <d_{i1}, d_{i2}, ..., d_{im}>, where d_{ij} represents the number of occurrences of w_j in the document d_i.

After performing feature reduction, the objective is to obtain a new word set W'={w'_1, w'_2, ..., w'_k}, where $k < m$, such that W' works equally well to represent a document as did W.

After reduction, each document d_i is converted into a new representation d'_i = <d'_1, d'_2, ..., d'_p>.

Since k is much smaller than m, computation cost is drastically reduced.

B. Word Pattern Construction

Word pattern represents the probability of a word occurring in a given class.

Given a document D of n documents d_1, d_2, …, d_n, and feature vector W of m words w_1, w_2, …, w_m and p classes c_1, c_2, …, c_p, we construct the word pattern x_i for a word w_i as,

$$x_i = < x_{i1}, x_{i2}, \dots x_{ip} >$$
$$= < P(c_1|w_i), P(c_2|w_i), \dots, P(c_p|w_i) >$$

where

$$P(c_j|w_i) = \frac{\sum_{q=1}^{n} d_{qi} \times \delta_{qj}}{\sum_{q=1}^{n} d_{qi}}$$

for $1 \le j \le p$ where d_{qi} indicates the number of occurrences of w_i in document d_q.

Also, δ_{qj} is defined as

$$\delta_{qj} = \begin{cases} 1, & if\ document\ d_q, belongs\ to\ class\ c_j \\ 0, & otherwise \end{cases}$$

Hence, we have m word patterns in total.

C. Self-Constructing Clustering

Our clustering algorithm is an incremental, self-constructing learning approach. Word patterns are considered one by one. The user does not need to have any idea about the number of clusters in advance. No clusters exist at the beginning, and clusters can be created if necessary. For each word pattern, the similarity of this word pattern to each existing cluster is calculated to decide whether it is combined into an existing cluster or a new cluster is created. Once a new cluster is created, the corresponding membership function should be initialized. On the contrary, when the word pattern is combined into an existing cluster, the membership function of that cluster should be updated accordingly.

Let k be the number of currently existing clusters. The clusters are G_1, G_2, ... , G_k, respectively. Each cluster G_j has mean m_j and deviation σ_j . Let S_j be the size of cluster G_j. Initially, we have k = 0. So, no clusters exist at the beginning. For each word pattern xi = <x_{i1}, x_{i2}, … x_{ip}>, $1 \le i \le m$, we calculate the similarity of x_i to each existing cluster, i.e.

$$\mu_{G_j}(x_i) = \prod_{q=1}^{p} exp\left[-\left(\frac{x_{iq} - m_{jq}}{\sigma_{jq}}\right)\right]$$

for $1 \le j \le k$, we say that x_i passes the similarity test on cluster G_j if

$$\mu_{G_j}(x_i) \ge \rho$$

where ρ, $0 \le \rho \le 1$, is a predefined threshold. If the user intends to have larger clusters, then he/she can give a smaller threshold. Otherwise, a bigger threshold can be given. As the threshold increases, the number of clusters also increases. A larger value will make the boundaries of the Gaussian function sharper, and more clusters will be obtained for a given threshold. On the contrary, a smaller value will make the boundaries of the Gaussian function smoother, and fewer clusters will be obtained instead.

Two cases may occur. First, there are no existing fuzzy clusters on which x_i has passed the similarity test. For this case, we assume that x_i is not similar enough to any existing cluster and a new cluster G_h, h = k + 1, is created with

$m_h = x_i$, $\sigma_h = \sigma_0$

where, $\sigma_0 = < \sigma_0, \sigma_0, \dots, \sigma_0 >$ is a user-defined constant vector. The deviation of a new cluster is 0, since it contains only one member. We cannot use zero deviation in the calculation of fuzzy similarities. Therefore, we initialize the deviation of a newly created cluster by σ_0.

The number of clusters is increased by 1 and size of clusters is initialized.

k = k + 1, S_h = 1,

Second, there exists a fuzzy cluster G_t in which x_i has passed the similarity test. For this case, we add the word w_i to the cluster t.

The modification to the cluster G_t is defined as

$$m_{tj} = \frac{S_t \times m_{tj} + x_{ij}}{S_t + 1}$$

$$\sigma_{tj} = \sqrt{A - B} + \sigma_0$$

$$A = \frac{(S_t - 1)(\sigma_{tj} - \sigma_0)^2 + S_t \times m_{tj}^2 + x_{ij}^2}{S_t}$$

$$B = \frac{S_t + 1}{S_t}\left(\frac{S_t \times m_{tj} + x_{ij}}{S_t + 1}\right)^2$$

$S_t = S_t + 1$

The above process is iterated for each word pattern and we obtain k clusters.

D. Feature Extraction

Now that the features have been successfully clustered together, the next step is to extract the representative features from a cluster. This feature will be part of the reduced feature set.

D' = DT

$D = [d_1, d_2, \ldots, d_n]^T$

$D' = [d'_1, d'_2, \ldots, d'_n]^T$

$$T = \begin{bmatrix} t_{11} & \cdots & t_{1k} \\ t_{21} & \cdots & t_{2k} \\ \vdots & \ddots & \vdots \\ t_{m1} & \cdots & t_{mk} \end{bmatrix}$$

with,

$d_i = [d_{i1}, d_{i1}, \ldots, d_{im}]$

$d'_i = [d'_{i1}, d'_{i1}, \ldots, d'_{im}]$

T is a weighting matrix and the elements of **T** are binary and is defined as,

$$t_{ij} = \begin{cases} 1, & if\ w_i \in W_j \\ 0, & otherwise \end{cases}$$

where $1 \le i \le m$ and $1 \le j \le k$. i.e. if a word belongs to cluster W_j, t_{ij} is 1, otherwise t_{ij} is 0.

E. Text Classification

For a input document D, we specify the threshold ρ and apply our clustering algorithm. We then find the weighting matrix **T** on the document **D** and convert **D** to **D'**. Using **D'** as the training data, a classifier based on support vector[10] machine is built.

SVM is a kernel method, which finds the maximum margin hyperplane in feature space separating the images of the training patterns into two groups. Since we have more than 2 classes, we need to build multiple SVMs. Ideally, for **p** classes, we need to build **p** SVMs. The classifier is then the aggregation of all these SVMs.

IV. Example

This example illustrates the working of our algorithm.

Let D be the document set containing 9 documents d1, d2, …, d9 of two classes with feature vector containing 10 words.

Figure 1 Sample document D

	Office (w_1)	Building (w_2)	Line (w_3)	Floor (w_4)	Bedroom (w_5)	Kitchen (w_6)	Apartment (w_7)	Internet (w_8)	WC (w_9)	Fridge (w_{10})	class
d_1	0	1	0	0	1	1	0	0	0	1	c_1
d_2	0	0	0	0	0	2	1	1	0	0	c_1
d_3	0	0	0	0	0	0	1	0	0	0	c_1
d_4	0	0	1	0	2	1	2	1	0	1	c_1
d_5	0	0	0	1	0	1	0	0	1	0	c_2
d_6	2	1	1	0	0	1	0	0	1	0	c_2
d_7	3	2	1	3	0	1	0	1	1	0	c_2
d_8	1	0	1	1	0	1	0	0	0	0	c_2
d_9	1	1	1	1	0	0	0	0	0	0	c_2

The word pattern xi is calculated for each word in the feature vector W.

Figure 2 Word pattern X

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
0.00	0.20	0.20	0.00	1.00	0.50	1.00	0.67	0.00	1.00
1.00	0.80	0.80	1.00	0.00	0.50	0.00	0.33	1.00	0.00

Figure 3 Cluster formation

Cluster	Size S	mean **m**	Deviation $\boldsymbol{\sigma}$
G_1	3	$< 1,0 >$	$< 0.5, 0.5 >$
G_2	5	$< 0.08, 0.92 >$	$< 0.6095, 0.6095 >$
G_3	2	$< 0.5833, 0.4167 >$	$< 0.6179, 0.6179 >$

Figure 4 Fuzzy similarity of word pattern to cluster

similarity	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
$\mu_{G_1}(\mathbf{x})$	0.0003	0.0060	0.0060	0.0003	1.0000	0.1353	1.0000	0.411	0.0003	1.0000
$\mu_{G_2}(\mathbf{x})$	0.9661	0.9254	0.9254	0.9661	0.0105	0.3869	0.0105	0.1568	0.9661	0.0105
$\mu_{G_3}(\mathbf{x})$	0.1682	0.4631	0.4631	0.1682	0.4027	0.9643	0.4027	0.9643	0.1682	0.4027

Figure 5 Weighting matrix T

$$\mathbf{T}_H = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

FIGURE 6 TRANSFORMED DATA SET

	(w'_1)	(w'_2)	(w'_3)
d'_1	2	1	1
d'_2	1	0	3
d'_3	1	0	0
d'_4	5	1	2
d'_5	0	2	1
d'_6	0	5	1
d'_7	0	10	2
d'_8	0	3	1
d'_9	0	4	0

$\mathbf{D}^I_H$

V. Conclusion

We have presented a fuzzy self-constructing feature clustering algorithm, which is an incremental clustering approach to reduce the dimensionality of the features in text classification. Features that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. If a word is not similar to any existing cluster, a new cluster is created for this word. Similarity between a word and a cluster is defined by considering both the mean and the variance of the cluster. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature corresponding to a cluster is a weighted combination of the words contained in the cluster. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data.

References

[1] I.T. Jolliffe, Principal Component Analysis. Springer-Verlag, 1986

[2] A.M. Martinez and A.C. Kak, “PCA versus LDA,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 2 pp. 228-233, Feb. 2001.

[3] H. Li, T. Jiang, and K. Zang, “Efficient and Robust Feature Extraction by Maximum Margin Criterion,” T. Sebastian, S. Lawrence, and S. Bernhard eds. Advances in Neural Information Processing System, pp. 97-104, Springer, 2004.

[4] H. Park, M. Jeon, and J. Rosen, “Lower Dimensional Representation of Text Data Based on Centroids and Least Squares,” BIT Numerical Math, vol. 43, pp. 427-448, 2003.

[5] S.T. Roweis and L.K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” Science, vol. 290, pp. 2323-2326, 2000.

[6] J.B. Tenenbaum, V. de Silva, and J.C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” Science, vol. 290, pp. 2319-2323, 2000.

[7] M. Belkin and P. Niyogi, “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering,” Advances in Neural Information Processing Systems, vol. 14, pp. 585-591, The MIT Press 2002.

[8] H. Al-Mubaid and S.A. Umair, “A New Text Categorization Technique Using Distributional Clustering and Learning Logic,” IEEE Trans. Knowledge and Data Eng., vol. 18, no. 9, pp. 1156-1165, Sept. 2006.

[9] I.S. Dhillon, S. Mallela, and R. Kumar, “A Divisive InfomationTheoretic Feature Clustering Algorithm for Text Classification,” J. Machine Learning Research, vol. 3, pp. 1265-1287, 2003.

[10] C.C. Chang and C.J. Lin, “Libsvm: A Library for Support Vector Machines,” http://www.csie.ntu.edu.tw/~cjlin/libsvm. 2001.