IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, p-ISSN: 2278-8719, www.iosrjen.org
Volume 2, Issue 9 (September 2012), PP 36-44

A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification

A.Kavitha[1,] Y.Sowjanya Kumari[2,] Dr.P.Harini[3]
[1] *II M.Tech C.S.E St.Ann's College of Engineering &Technology,Chirala*
[2] *M.Tech (C.S.E)Department of Associate Professor St.Ann's College of Engineering &Technology,Chirala.*
[3] *Professor St.Ann's College of Engineering &Technology,Chirala. (C.S.E).*

Abstract—*Feature clustering is a powerful method to reduce the dimensionality of feature vectors for text classification. In this paper, we propose a fuzzy similarity-based self-constructing algorithm for feature clustering. The words in the feature vector of a document set are grouped into clusters, based on similarity test. Words 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. 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. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided.*

Index Terms—*Fuzzy similarity, feature clustering, feature extraction, feature reduction, text classification*

I. INTRODUCTION

In text classification, the dimensionality of the feature vector is usually huge. For example, 20 Newsgroups [1] and Reuters21578 top-10 [2], which are two real-world data sets, both have more than 15,000 features. Such high dimensionality can be a severe obstacle for classification algorithms [3], [4]. To alleviate this difficulty, feature reduction approaches are applied before document classification tasks are performed [5]. Two major approaches, feature selection [6], [7], [8], [9], [10] and feature extraction [11], [12], [13], have been proposed for feature reduction. In general, feature extraction approaches are more effective than feature selection techniques, but are more computationally expensive [11], [12], [14]. Therefore, developing scalable and efficient feature extraction algorithms is highly demanded for dealing with high-dimensional document data sets.

Classical feature extraction methods aim to 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 [15], Linear Discriminant Analysis [16], Maximum Margin Criterion [12], and Orthogonal Centroid algorithm [17] perform the projection by linear transformations, while Locally Linear Embedding [18], ISOMAP [19], and Laplacian Eigenmaps [20] do feature extraction by nonlinear transformations. In practice, linear algorithms are in wider use due to their efficiency. Several scalable online linear feature extraction algorithms [14], [21], [22], [23] have been proposed to improve the computational complexity. However, the complexity of these approaches is still high. Feature clustering [24], [25], [26], [27], [28], [29] is one of effective techniques for feature reduction in text classification. The idea of feature clustering is to group the original features into clusters with a high degree of pairwise semantic relatedness. Each cluster is treated as a single new feature, and, thus, feature dimensionality can be drastically reduced.

The first feature extraction method based on feature clustering was proposed by Baker and McCallum [24], which was derived from the "distributional clustering" idea of Pereira et al. [30]. Al-Mubaid and Umair [31] used distributional clustering to generate an efficient representation of documents and applied a learning logic approach for training text classifiers. The Agglomerative Information Bottleneck approach was proposed by Tishby et al. [25], [29]. The divisive information-theoretic feature clustering algorithm was proposed by Dhillon et al. [27], 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. However, difficulties are associated with these methods. A word is exactly assigned to a subset, i.e., hard-clustering, based on the similarity magnitudes between the word and the existing subsets, even if the differences among these magnitudes are small. Also, the mean and the variance of a cluster are not considered when similarity with respect to the cluster is computed. Furthermore, these methods require the number of new features be specified in advance by the user.

We propose a fuzzy similarity-based self-constructing feature clustering algorithm, which is an incremental feature clustering approach to reduce the number of features for the text classification task. The words in the feature vector of a document set are represented as distributions, and processed one after another. Words 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. Three ways of weighting, hard, soft, and mixed, are introduced. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided. Experiments on real world data sets show that our method can run faster and obtain better extracted features than other methods.

II. BACKGROUND AND RELATED WORK

To process documents, the bag-of-words model [32], [33] is commonly used. Let $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots\ldots, \mathbf{d}_\text{n}\}$ be a document set of *n* documents, where $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots\ldots, \boldsymbol{d}_n$ are individual documents, and each document belongs to one of the classes in the set $\{c_1, c_2, \ldots\ldots, c_p\}$. If a document belongs to two or more classes, then two or more copies of the document with different classes are included in **D**. Let the word set $\mathbf{W} = \{w_1, w_2, \ldots\ldots, w_m\}$ be the feature vector of the document set. Each document $\boldsymbol{d}_i, 1 \leq i \leq n$, is represented as$\mathbf{d}_i = < d_{i1}, d_{i2}, \ldots\ldots, d_{im} >$, where each d_{ij} denotes the number of occurrences of wj in the ith document. The feature reduction task is to find a new word set $\mathbf{W}^{'} = \{w_1^{'}, w_2^{'}, \ldots\ldots, w_k^{'}\}$,k << m, such that **W** and $\mathbf{W}^{'}$ work equally well for all the desired properties with **D**. After feature reduction, each document $\mathbf{d}_i$ is converted into a new representation $\mathbf{d}_i^{'} = < d_{i1}^{'}, d_{i2}^{'}, \ldots\ldots, d_{ik}^{'} >$ and the converted document set is $\mathbf{D}^{'} = \{\mathbf{d}_1^{'}, \mathbf{d}_2^{'}, \ldots\ldots, \mathbf{d}_\text{n}^{'}\}$. If k is much smaller than m, computation cost with subsequent operations on $\mathbf{D}^{'}$ can be drastically reduced.

2.1 Feature Reduction

In general, there are two ways of doing feature reduction, feature selection, and feature extraction. By feature selection approaches, a new feature set $\mathbf{W}^{'} = \{w_1^{'}, w_2^{'}, \ldots\ldots, w_k^{'}\}$ is obtained, which is a subset of the original feature set **W**. Then $\mathbf{W}^{'}$ is used as inputs for classification tasks. Information Gain (IG) is frequently employed in the feature selection approach [10]. It measures the reduced uncertainty by an information-theoretic measure and gives each word a weight. The weight of a word w_j is calculated as follows

$$IG(w_j) = -\sum_{l=1}^{p} P(c_l)logP(c_l) + P(w_j)\sum_{l=1}^{p} P(c_l|w_j)logP(c_l|w_j) + P(\overline{w_j})\sum_{l=1}^{p} P(c_l|\overline{w_j})logP(c_l|\overline{w_j}) \qquad (1)$$

where $P(c_l)$ denotes the prior probability for class c_l, $P(w_j)$ denotes the prior probability for feature w_j, $P(\overline{w_j})$ is identical to $1 - P(w_j)$, and $P(c_l|w_j)$ and $P(c_l|\overline{w_j})$ denote the probability for class c_l with the presence and absence, respectively, of w_j. The words of top *k* weights in **W** are selected as the features in $\mathbf{W}^{'}$.

In feature extraction approaches, extracted features are obtained by a projecting process through algebraic transformations.

An incremental orthogonal centroid (IOC) algorithm was proposed in [14]. Let a corpus of documents be represented as an $m \times n$ matrix $\mathbf{X} \in \mathbf{R}^{m\times n}$, where m is the number of features in the feature set and n is the number of documents in the document set. IOC tries to find an optimal transformation matrix $\mathbf{F}^* \in \mathbf{R}^{m\times k}$, where k is the desired number of extracted features, according to the following criterion:

$$\mathbf{F}^* = \arg\max trace\ (\mathbf{F}^T\mathbf{S}_b\mathbf{F}), \qquad (2)$$

Where $\mathbf{F} \in \mathbf{R}^{m\times k}$ and $\mathbf{F}^T\mathbf{F} = \mathbf{I}$, and

$$\mathbf{S}_b = \sum_{q=1}^{p} P(c_q)(\mathbf{M}_q - \mathbf{M}_{all})(\mathbf{M}_q - \mathbf{M}_{all})^T \qquad (3)$$

with $P(c_q)$ being the prior probability for a pattern belonging to class c_q, $\mathbf{M}_q$ being the mean vector of class c_q, and $\mathbf{M}_{all}$ being the mean vector of all patterns.

2.2 Feature Clustering

Feature clustering is an efficient approach for feature reduction [25], [29], which groups all features into some clusters, where features in a cluster are similar to each other. The feature clustering methods proposed in [24], [25], [27], [29] are "hard" clustering methods, where each word of the original features belongs to exactly one word cluster. Therefore each word contributes to the synthesis of only one new feature. Each new feature is obtained by summing up the words belonging to one cluster. Let **D** be the matrix consisting of all the original documents with m features and $\mathbf{D}'$ be the matrix consisting of the converted documents with new k features. The new feature set $\mathbf{W}' = \{w_1', w_2', \ldots\ldots, w_k'\}$ corresponds to a artition $\{\mathbf{W}_1, \mathbf{W}_2, \ldots\ldots, \mathbf{W}_k\}$ of the original feature set **W**, i.e., $\mathbf{W}_t \cap \mathbf{W}_q = \emptyset$, where $1 \leq q, t \leq k$ and $t \neq q$. Note that a cluster corresponds to an element in the partition. Then, the tth feature value of the converted document $\mathbf{d}_i'$ is calculated as follows:

$$d_{it}' = \sum_{w_j \in \mathbf{W}_t} d_{ij} \qquad (4)$$

which is a linear sum of the feature values in $\mathbf{W}_t$. The divisive information-theoretic feature clustering (DC) algorithm, proposed by Dhillon et al. [27] calculates the distributions of words over classes, $P(\mathbf{C}|w_j), 1 \leq j \leq m$, where, $C = \{c_1, c_2, \ldots, c_p\}$ and uses *Kullback-Leibler* divergence to measure the dissimilarity between two distributions.

The distribution of a cluster $\mathbf{W}_t$ is calculated as follows:

$$P(\mathbf{C}|\mathbf{W}_t) = \sum_{w_j \in \mathbf{W}_t} \frac{P(w_j)}{\sum_{w_j \in \mathbf{W}_t} P(w_j)} P(\mathbf{C}|w_j) \qquad (5)$$

The goal of DC is to minimize the following objective function:

$$\sum_{t=1}^{k} \sum_{w_j \in \mathbf{W}_t} P(w_j) KL(P(\mathbf{C}|w_j), P(\mathbf{C}|\mathbf{W}_t)) \qquad (6)$$

which takes the sum over all the k clusters, where k is specified by the user in advance.

III. OUR METHOD

There are some issues pertinent to most of the existing feature clustering methods. First, the parameter k, indicating the desired number of extracted features, has to be specified in advance. This gives a burden to the user, since trial-and-error has to be done until the appropriate number of extracted features is found. Second, when calculating similarities, the variance of the underlying cluster is not considered. Intuitively, the distribution of the data in a cluster is an important factor in the calculation of similarity. Third, all words in a cluster have the same degree of contribution to the resulting extracted feature. Sometimes, it may be better if more similar words are allowed to have bigger degrees of contribution. Our feature clustering algorithm is proposed to deal with these issues.

Suppose, we are given a document set **D** of n documents $\mathbf{d}_1, \mathbf{d}_2, \ldots\ldots \mathbf{d}_n$ together with the feature vector **W** of m words $w_1, w_2, \ldots\ldots w_m$ and p classes $c_1, c_2, \ldots\ldots c_{p,}$ as specified in Section 2. We construct one *word pattern* for each word in **W**. For word w_i, its word pattern $\mathbf{x}_i$ is defined, similarly as in [27], by

$$\mathbf{x}_i =< x_{i1}, x_{i2}, \ldots\ldots, x_{ip} >$$
$$=< P(c_1|w_i), P(c_2|w_i), \ldots\ldots, P(c_p|w_i) > \qquad (7)$$

Where

$$P(c_1|w_i) = \frac{\sum_{q=1}^{n} d_{qi} \times \delta_{qj}}{\sum_{q=1}^{n} d_{qi}} \qquad (8)$$

for $1 \leq j \leq p$. Note that d_{qi} indicates the number of occurrences of w_i in document d_q, as described in Section 2.

Also, δ_{qj} is defined as

$$\delta_{qj} = \begin{cases} 1, & \text{if document } \mathbf{d}_q \text{ belongs to class } c_j \\ 0, & \text{otherwise.} \end{cases} \qquad (9)$$

Therefore, we have m word patterns in total. It is these word patterns, our clustering algorithm will work on. Our goal is to group the words in **W** into clusters, based on these word patterns. A cluster contains a certain number of word patterns, and is characterized by the product of p one-dimensional Gaussian functions. Gaussian functions are adopted because of their superiority over other functions in performance [34], [35]. Let G be a cluster containing q word patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots.\mathbf{x}_q$. Let $\mathbf{x}_j =< x_{j1}, x_{j2}, \ldots\ldots, x_{jp} >, 1 \leq j \leq q$. Then the mean $\mathbf{m} =< m_1, m_2, \ldots\ldots, m_p >$ and the deviation $\boldsymbol{\sigma} =< \sigma_1, \sigma_2, \ldots\ldots, \sigma_p >$ of G are defined as

$$m_i = \frac{\sum_{j=1}^{q} x_{ji}}{|G|} \tag{10}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{q}(x_{ji} - m_{ji})^2}{|G|}} \tag{11}$$

for $1 \leq i \leq p$, where $|G|$ denotes the size of G, i.e., the number of word patterns contained in G. The *fuzzy similarity* of a word pattern $\mathbf{x} =< x_1,\ x_2, \ldots \ldots, x_p >$ to cluster G is defined by the following membership function:

$$\mu_G(\mathbf{x}) = \prod_{i=1}^{p} \exp\left[-\left(\frac{x_i - m_i}{\sigma_i}\right)^2\right] \tag{12}$$

Notice that $0 \leq \mu_G(\mathbf{x}) \leq 1$. A word pattern close to the mean of a cluster is regarded to be very similar to this cluster, i.e.,
$\mu_G(\mathbf{x}) \approx 1$. On the contrary, a word pattern far distant from a cluster is hardly similar to this cluster, i.e., $\mu_G(\mathbf{x}) \approx 0$.

3.1 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 $G_1, G_2, \ldots. G_k$, respectively. Each cluster G_j has mean $\mathbf{m}_j =< m_{j1},\ m_{j2}, \ldots \ldots, m_{jp} >$ and deviation $\boldsymbol{\sigma}_j =< \sigma_{j1},\ \sigma_{j2}, \ldots \ldots, \sigma_{jp} >$. 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 $\mathbf{x}_i =< x_{i1},\ x_{i2}, \ldots \ldots, x_{ip} >, 1 \leq i \leq m$, we calculate, according to (13), the similarity of $\mathbf{x}_i$ to each existing clusters, i.e.

$$\mu_{Gj}(\mathbf{x}_i) = \prod_{i=1}^{p} \exp\left[-\left(\frac{x_{iq} - m_{jq}}{\sigma_{jq}}\right)^2\right] \tag{13}$$

for $1 \leq j \leq k$. We say that $\mathbf{x}_i$ passes the similarity test on cluster G_j if

$$\mu_{Gj}(\mathbf{x}_i) \geq \rho \tag{14}$$

where $\rho, 0 \leq \rho \leq 1$, 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. Note that, as usual, the power in (13) is 2 [34], [35]. Its value has an effect on the number of clusters obtained. 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 xi has passed the similarity test. For this case, we assume that xi is not similar enough to any existing cluster and a new cluster G_h h = k + 1, is created with

$$\mathbf{m}_h = \mathbf{x}_i, \quad \boldsymbol{\sigma}_h = \boldsymbol{\sigma}_0 \tag{15}$$

where $\boldsymbol{\sigma}_0 =< \sigma_0, \ldots \ldots . \sigma_0 >$ is a user-defined constant vector. Note that the new cluster G_h contains only one member, the word pattern $\mathbf{x}_i$, at this point. Estimating the deviation of a cluster by (11) is impossible, or inaccurate, if the cluster contains few members. In particular, 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 $\boldsymbol{\sigma}_0$, as indicated in (15). Of course, the number of clusters is increased by 1 and the size of cluster G_h, S_h,, should be initialized, i.e.,

$$k = k + 1, \quad S_h = 1 \tag{16}$$

Second, if there are existing clusters on which $\mathbf{x}_i$ has passed the similarity test, let cluster G_h be the cluster with the largest membership degree, i.e.,

$$t = arg \max_{1 \leq j \leq k} \mu_{Gj}(\mathbf{x}_i) \tag{17}$$

In this case, we regard $\mathbf{x}_i$ to be most similar to cluster G_t, and $\mathbf{m}_t$ and $\boldsymbol{\sigma}_t$ of cluster G_t should be modified to include $\mathbf{x}_i$ as its member. The modification to cluster G_t is described as follows:

$$m_{tj} = \frac{S_t \times m_{tj} + x_{ij}}{S_t + 1} \qquad (18)$$

$$\sigma_{tj} = \sqrt{A - B} + \sigma_0 \qquad (19)$$

$$A = \frac{(S_t - 1)\left(\sigma_{tj} - \sigma_0\right)^2 + S_t \times m_{tj}^2 + x_{ij}^2}{S_t} \qquad (20)$$

$$B = \frac{S_t + 1}{S_t}\left(\frac{S_t \times m_{tj} + x_{ij}}{S_t + 1}\right)^2 \qquad (21)$$

for $1 \leq j \leq p$, and

$$S_t = S_t + 1 \qquad (22)$$

Equations (18) and (19) can be derived easily from (10) and (11). Note that k is not changed in this case. The whole clustering algorithm can be summarized below.

Initialization:

# of original word patterns: m
# of classes: p
Threshold: ρ
Initial deviation: σ_0
Initial # of clusters: k = 0
Input:

```
    $\mathbf{x}_i =< x_{i1}, x_{i2}, \dots\dots, x_{ip} >, 1 \leq i \leq m$
  Output:
    Clusters $G_1, G_2, \dots. G_k$
  procedure Self-Constructing-Clustering-Algorithm
    for each word pattern $\mathbf{x}_i, 1 \leq i \leq m$
      $temp\_\mathbf{W} = \{G_j \,|\mu_{Gj}(\mathbf{x}_i) \geq \rho, 1 \leq j \leq k\}$;
      if ($temp\_\mathbf{W}$ == $\emptyset$)
        A new cluster$G_h, h = k + 1$, is created by (15)-(16);
      else let $G_t \in temp\_\mathbf{W}$ be the cluster to which $\mathbf{x}_i$ is
        closest by (17);
        Incorporate $\mathbf{x}_i$ into $G_t$ by (18)-(22);
      endif;
    endfor;
    return with the created k clusters;
  endprocedure
```

Note that the word patterns in a cluster have a high degree of similarity to each other. Besides, when new training patterns are considered, the existing clusters can be adjusted or new clusters can be created, without the necessity of generating the whole set of clusters from the scratch.

Note that the order in which the word patterns are fed in influences the clusters obtained. We apply a heuristic to determine the order. We sort all the patterns, in decreasing order, by their largest components. Then the word patterns are fed in this order. In this way, more significant patterns will be fed in first and likely become the core of the underlying cluster. This heuristic seems to work well.

We discuss briefly here the computational cost of our method and compare it with DC [27], IOC [14], and IG [10]. For an input pattern, we have to calculate the similarity between the input pattern and every existing cluster. Each pattern consists of p components where p is the number of classes in the document set. Therefore, in worst case, the time complexity of our method is *O(mkp)*, where m is the number of original features and k is the number of clusters finally obtained. For DC, the complexity is *O(mkpt)*, where t is the number of iterations to be done. The complexity of IG is *O(mp+m*log*m)*, and the complexity of IOC is *O(mkpn)*, where n is the number of documents involved. Apparently, IG is the quickest one. Our method is better than DC and IOC.

3.2 Feature Extraction

Formally, feature extraction can be expressed in the following form:

$$\mathbf{D}' = \mathbf{DT} \qquad (23)$$

$$\mathbf{D} = [\mathbf{d_1}\ \mathbf{d_2}\ \dots.\ \ \mathbf{d_n}]^T, \qquad (24)$$

$$\mathbf{D}' = [\mathbf{d}_1'\ \mathbf{d}_2'\ \dots.\ \ \mathbf{d}_n']^T, \qquad (25)$$

$$\mathbf{T} = \begin{bmatrix} t_{11} & \ldots. & t_{1k} \\ t_{21} & \ldots. & t_{2k} \\ & & \\ t_{m1} & \ldots. & t_{mk} \end{bmatrix}, \qquad (26)$$

with

$\mathbf{d}_i = [d_{i1}\ d_{i2}\ \ldots..d_{im}],$

$\mathbf{d}_i' = [d_{i1}'\ d_{i2}'\ \ldots\ldots\ d_{ik}'],$

for $1 \le i \le n$. Clearly, **T** is a weighting matrix. The goal of feature reduction is achieved by finding an appropriate **T** such that k is smaller than m. In the divisive information theoretic feature clustering algorithm [27] described in Section 2.2, the elements of T in (25) are binary and can be defined as follows:

$$t_{ij} = \begin{cases} 1, & if\ w_i \in \mathbf{W}_j, \\ 0, & \text{otherwise}, \end{cases} \qquad (27)$$

where $1 \le i \le m$ and $1 \le j \le k$. That is, if a word wi belongs to cluster $\mathbf{W}_j$, t_{ij} is 1; otherwise t_{ij} is 0.

By applying our clustering algorithm, word patterns have been grouped into clusters, and words in the feature vector **W** are also clustered accordingly. For one cluster, we have one extracted feature. Since we have k clusters, we have k extracted features. The elements of **T** are derived based on the obtained clusters, and feature extraction will be done. We propose three weighting approaches: hard, soft, and mixed. In the hard-weighting approach, each word is only allowed to belong to a cluster, and so it only contributes to a new extracted feature. In this case, the elements of **T** in (23) are defined as follows:

$$t_{ij} = \begin{cases} 1, & if\ j = arg \max_{1 \le j \le k} \mu_{Gj}(\mathbf{x}_i), \\ 0, & \text{otherwise}, \end{cases} \qquad (28)$$

Note that if j is not unique in (28), one of them is chosen randomly. In the soft-weighting approach, each word is allowed to contribute to all new extracted features, with the degrees depending on the values of the membership functions. The elements of **T** in (23) are defined as follows:

$$t_{ij} = \mu_{Gj}(\mathbf{x}_i) \qquad (29)$$

The mixed-weighting approach is a combination of the hard-weighting approach and the soft-weighting approach. For this case, the elements of **T** in (23) are defined as follows:

$$t_{ij} = (\gamma) \times t_{ij}^{H} + (1 - \gamma) \times t_{ij}^{S} \qquad (30)$$

where t_{ij}^{H} is obtained by (28) and t_{ij}^{S} is obtained by (29), and γ is a user-defined constant lying between 0 and 1. Note that is not related to the clustering. It concerns the merge of component features in a cluster into a resulting feature. The merge can be “hard” or “soft” by setting γ to 1 or 0. By selecting the value of γ, we provide flexibility to the user. When the similarity threshold is small, the number of clusters is small, and each cluster covers more training patterns. In this case, a smaller γ will favor soft-weighting and get a higher accuracy. On the contrary, when the similarity threshold is large, the number of clusters is large, and each cluster covers fewer training patterns. In this case, a larger γ will favor hard-weighting and get a higher accuracy.

3.3 Text Classification

Given a set **D** of training documents, text classification can be done as follows: We specify the similarity threshold ρ for (15), and apply our clustering algorithm. Assume that k clusters are obtained for the words in the feature vector W. Then we find the weighting matrix **T** and convert **D** to $\mathbf{D}'$ by (23). Using $\mathbf{D}'$ as training data, a classifier based on support vector machines (SVM) is built. Note that any classifying technique other than SVM can be applied. Joachims [36] showed that SVM is better than other methods for text categorization. SVM is a kernel method, which finds the maximum margin hyper plane in feature space separating the images of the training patterns into two groups [37], [38], [39]. To make the method more flexible and robust, some patterns need not be correctly classified by the hyperplane, but the misclassified patterns should be penalized. Therefore, slack variables ξ_i are introduced to account for misclassifications. The objective function and constraints of the classification problem can be formulated as:

$$\min_{w,b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \xi_i \qquad (31)$$

$$s.t.\, y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b}) \ge 1 - \xi_i, \xi_i \ge 0, i = 1, 2, \ldots\ldots., l,$$

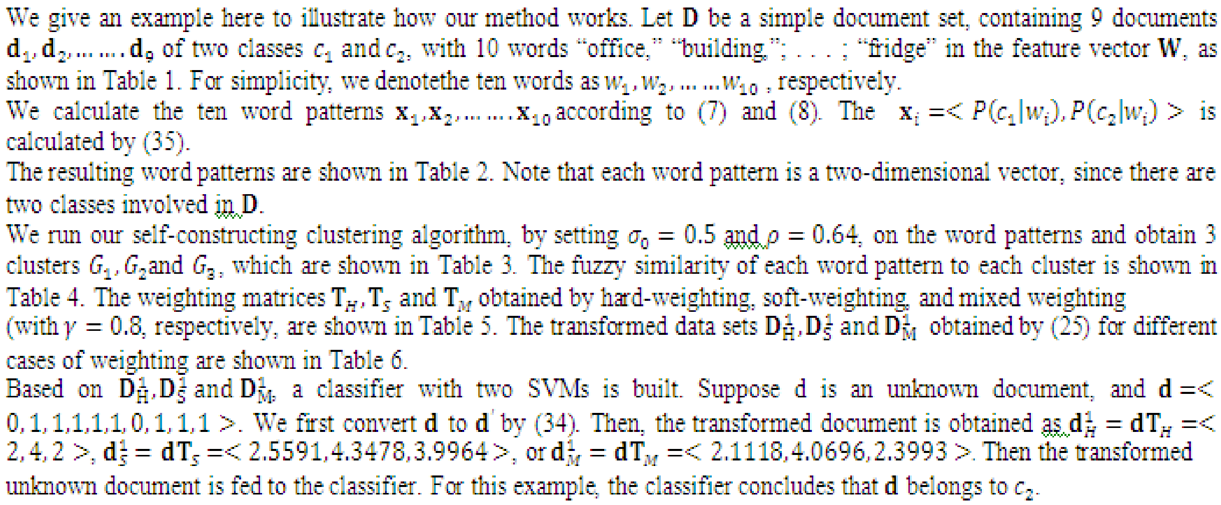
where l is the number of training patterns, C is a parameter, which gives a tradeoff between maximum margin and classification error, and y_i, being +1 or -1, is the target label of pattern $\mathbf{x}_i$. Note that $\phi: X \to F$ is a mapping from the input space to the feature space F, where patterns are more easily separated, and $\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b} = 0$ is the hyper plane to be derived with w, and b being weight vector and offset, respectively.

An SVM described above can only separate apart two classes, $y_i = +1$ and $y_i = -1$. We follow the idea in [36] to construct an SVM-based classifier. For p classes, we create p SVMs, one SVM for each class. For the SVM of class $c_v, 1 \leq v \leq p$, the training patterns of class c_v are treated as having $y_i = +1$, and the training patterns of the other classes are treated as having $y_i = -1$. The classifier is then the aggregation of these SVMs. Now we are ready for classifying unknown documents. Suppose, $\mathbf{d}$ is an unknown document. We first convert $\mathbf{d}$ to $\mathbf{d}^{'}$ by

$$\mathbf{d}^{'} = \mathbf{dT} \qquad (32)$$

Then we feed $\mathbf{d}^{'}$ to the classifier. We get p values, one from each SVM. Then $\mathbf{d}$ belongs to those classes with 1, appearing at the outputs of their corresponding SVMs.

IV. AN EXAMPLE FOR PROPOSED METHOD



We give an example here to illustrate how our method works. Let $\mathbf{D}$ be a simple document set, containing 9 documents $\mathbf{d}_1, \mathbf{d}_2, \dots \dots \mathbf{d}_9$ of two classes c_1 and c_2, with 10 words "office," "building,"; . . . ; "fridge" in the feature vector $\mathbf{W}$, as shown in Table 1. For simplicity, we denotethe ten words as $w_1, w_2, \dots \dots w_{10}$, respectively.

We calculate the ten word patterns $\mathbf{x}_1, \mathbf{x}_2, \dots \dots \mathbf{x}_{10}$ according to (7) and (8). The $\mathbf{x}_i =< P(c_1|w_i), P(c_2|w_i) >$ is calculated by (35).

The resulting word patterns are shown in Table 2. Note that each word pattern is a two-dimensional vector, since there are two classes involved in $\mathbf{D}$.

We run our self-constructing clustering algorithm, by setting $\sigma_0 = 0.5$ and $\rho = 0.64$, on the word patterns and obtain 3 clusters $G_1, G_2$and G_3, which are shown in Table 3. The fuzzy similarity of each word pattern to each cluster is shown in Table 4. The weighting matrices $\mathbf{T}_H, \mathbf{T}_S$ and $\mathbf{T}_M$ obtained by hard-weighting, soft-weighting, and mixed weighting (with $\gamma = 0.8$, respectively, are shown in Table 5. The transformed data sets $\mathbf{D}_H^{\frac{1}{}}, \mathbf{D}_S^{\frac{1}{}}$ and $\mathbf{D}_M^{\frac{1}{}}$ obtained by (25) for different cases of weighting are shown in Table 6.

Based on $\mathbf{D}_H^{\frac{1}{}}, \mathbf{D}_S^{\frac{1}{}}$ and $\mathbf{D}_M^{\frac{1}{}}$, a classifier with two SVMs is built. Suppose d is an unknown document, and $\mathbf{d} =< 0, 1, 1, 1, 1, 1, 0, 1, 1, 1 >$. We first convert $\mathbf{d}$ to $\mathbf{d}^{'}$ by (34). Then, the transformed document is obtained as $\mathbf{d}_H^{\frac{1}{}} = \mathbf{dT}_H =< 2, 4, 2 >$, $\mathbf{d}_S^{\frac{1}{}} = \mathbf{dT}_S =< 2.5591, 4.3478, 3.9964 >$, or $\mathbf{d}_M^{\frac{1}{}} = \mathbf{dT}_M =< 2.1118, 4.0696, 2.3993 >$. Then the transformed unknown document is fed to the classifier. For this example, the classifier concludes that $\mathbf{d}$ belongs to c_2.

TABLE 1 A Simple Document Set 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

TABLE 2 Word Patterns of **W**

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
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

TABLE 3 Three Clusters Obtained

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	$< 0.5833, 0.4167 >$	$< 0.6179, 0.6179 >$

TABLE 4 Fuzzy Similarities of Word Patterns to Three Clusters

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

TABLE 5 Weighting Matrices: Hard $\mathbf{T}_H$, Soft $\mathbf{T}_S$, and Mixed $\mathbf{T}_M$

$$\mathbf{T}_H = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_S = \begin{bmatrix} 0.0003 & 0.9661 & 0.1682 \\ 0.0060 & 0.9254 & 0.4631 \\ 0.0060 & 0.9254 & 0.4631 \\ 0.0003 & 0.9661 & 0.1682 \\ 1.0000 & 0.0105 & 0.4027 \\ 0.1353 & 0.3869 & 0.9643 \\ 1.0000 & 0.0105 & 0.4027 \\ 0.4111 & 0.1568 & 0.9643 \\ 0.0003 & 0.9661 & 0.1682 \\ 1.0000 & 0.0105 & 0.4027 \end{bmatrix}, \quad \mathbf{T}_M = \begin{bmatrix} 0.0001 & 0.9932 & 0.0336 \\ 0.0012 & 0.9851 & 0.0926 \\ 0.0012 & 0.9851 & 0.0926 \\ 0.0001 & 0.9932 & 0.0336 \\ 1.0000 & 0.0021 & 0.0805 \\ 0.0271 & 0.0774 & 0.9929 \\ 1.0000 & 0.0021 & 0.0805 \\ 0.0822 & 0.0314 & 0.9929 \\ 0.0001 & 0.9932 & 0.0336 \\ 1.0000 & 0.0021 & 0.0805 \end{bmatrix}$$

TABLE 6 Transformed Data Sets: Hard $\mathbf{D}_H^1$, Soft $\mathbf{D}_S^1$ and Mixed $\mathbf{D}_M^1$

	(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

$\mathbf{D}_H^1$

	(w_1')	(w_2')	(w_3')
d_1'	2.1413	1.3333	2.2327
d_2'	1.6818	0.9411	3.2955
d_3'	1.0000	0.0105	0.4027
d_4'	5.5524	1.5217	4.4051
d_5'	0.13360	2.3192	1.3006
d_6'	0.1483	5.1362	2.3949
d_7'	0.5667	10.0829	4.4950
d_8'	0.1420	3.2446	1.7637
d_9'	0.0126	3.7831	1.2625

$\mathbf{D}_S^1$

	(w_1')	(w_2')	(w_3')
d_1'	2.0283	1.0667	1.2465
d_2'	1.1364	0.1882	3.0591
d_3'	1.0000	0.0021	0.0805
d_4'	5.1105	1.1043	2.4810
d_5'	0.0272	2.0638	1.0601
d_6'	0.0297	5.0272	1.2790
d_7'	0.1133	10.0166	2.4990
d_8'	0.0284	3.0489	1.1527
d_9'	0.0025	3.9566	0.2525

$\mathbf{D}_M^1$

V. CONCLUSION

We have presented a fuzzy self-constructing feature clustering (FFC) 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. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided.

REFERENCES

[1] Http://people.csail.mit.edu/jrennie/20Newsgroups/, 2010.

[2] Http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. 2010.

[3] H. Kim, P. Howland, and H. Park, “Dimension Reduction in Text Classification with Support Vector Machines,” J. Machine Learning Research, vol. 6, pp. 37-53, 2005.

[4] F. Sebastiani, “Machine Learning in Automated Text Categorization,” ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, 2002.

[5] B.Y. Ricardo and R.N. Berthier, Modern Information Retrieval. Addison Wesley Longman, 1999.

[6] A.L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," Aritficial Intelligence, vol. 97, nos. 1/2, pp. 245-271, 1997.

[7] E.F. Combarro, E. Montan˜ e´s, I. Dı´az, J. Ranilla, and R. Mones, "Introducing a Family of Linear Measures for Feature Selection in Text Categorization," IEEE Trans. Knowledge and Data Eng., vol. 17, no. 9, pp. 1223-1232, Sept. 2005.

[8] K. Daphne and M. Sahami, "Toward Optimal Feature Selection," Proc. 13th Int'l Conf. Machine Learning, pp. 284-292, 1996.

[9] R. Kohavi and G. John, "Wrappers for Feature Subset Selection," Aritficial Intelligence, vol. 97, no. 1-2, pp. 273-324, 1997.

[10] Y. Yang and J.O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," Proc. 14th Int'l Conf. Machine Learning, pp. 412-420, 1997.

[11] D.D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," Proc. Workshop Speech and Natural Language, pp. 212-217, 1992.

[12] 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.

[13] E. Oja, Subspace Methods of Pattern Recognition. Research Studies Press, 1983.

[14] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, and Z. Chen, "Effective and Efficient Dimensionality Reduction for Large-Scale and Streaming Data Preprocessing," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 3, pp. 320-333, Mar. 2006.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] K. Hiraoka, K. Hidai, M. Hamahira, H. Mizoguchi, T. Mishima, and S. Yoshizawa, "Successive Learning of Linear Discriminant Analysis: Sanger-Type Algorithm," Proc. IEEE CS Int'l Conf. Pattern Recognition, pp. 2664-2667, 2000.

[22] J. Weng, Y. Zhang, and W.S. Hwang, "Candid Covariance-Free Incremental Principal Component Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 8, pp. 1034-1040, Aug. 2003.

[23] J. Yan, B.Y. Zhang, S.C. Yan, Z. Chen, W.G. Fan, Q. Yang, W.Y. Ma, and Q.S. Cheng, "IMMC: Incremental Maximum Margin Criterion," Proc. 10th ACM SIGKDD, pp. 725-730, 2004.

[24] L.D. Baker and A. McCallum, "Distributional Clustering of Words for Text Classification," Proc. ACM SIGIR, pp. 96-103, 1998.

[25] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional Word Clusters versus Words for Text Categorization," J. Machine Learning Research, vol. 3, pp. 1183-1208, 2003.

[26] M.C. Dalmau and O.W.M. Flo´ rez, "Experimental Results of the Signal Processing Approach to Distributional Clustering of Terms on Reuters-21578 Collection," Proc. 29th European Conf. IR Research, pp. 678-681, 2007.

[27] I.S. Dhillon, S. Mallela, and R. Kumar, "A Divisive Infomation- Theoretic Feature Clustering Algorithm for Text Classification," J. Machine Learning Research, vol. 3, pp. 1265-1287, 2003.

[28] D. Ienco and R. Meo, "Exploration and Reduction of the Feature Space by Hierarchical Clustering," Proc. SIAM Conf. Data Mining, pp. 577-587, 2008.

[29] N. Slonim and N. Tishby, "The Power of Word Clusters for Text Classification," Proc. 23rd European Colloquium on Information Retrieval Research (ECIR), 2001.

[30] F. Pereira, N. Tishby, and L. Lee, "Distributional Clustering of English Words," Proc. 31st Ann. Meeting of ACL, pp. 183-190, 1993.

[31] H. Al-Mubaid and S.A. Umair, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic," IEEE Trans.