A Novel Method for Text Classification Dealing with Local Data Structures and High Data Dimensionality

Weijia Zhong[1]

[1] University of British Columbia, Vancouver BC V6T 1Z4, Canada

wegenuss@gmail.com

Abstract. Text categorization involves training models and making predictions over high-dimensional feature space. With the exponential growth on the amount of text resources on the Internet, the demand and difficulty of classifying digital text documents increases over the past decades. k nearest neighbors (kNN) has been proved to possess satisfying predictive power on text classification problems. However, kNN is sensitive to local data structures and its time cost is proportional to the product of the training set size and the feature space dimensionality. This paper proposes a novel text classification method called PCA pre-processed local correlation vector kNN (PCA-Local-CorVec-kNN) that applies PCA and kNN consecutively to the text datasets and introduces local correlation vectors to the sets of nearest neighbors produced by kNN. The proposed method has lower time cost than conventional kNN because of using PCA and is robust to local data structures since correlation vectors are introduced. Problems such as imbalanced datasets, choosing different k values for different samples, weighted voting are also tackled by the proposed method.

Keywords: K nearest neighbors, local data structures, principal component analysis, text categorization.

1. Introduction

With the development of Internet technologies, publication and access to digital text documents have been made easy. Over the past two decades, the amount of digital text documents grows exponentially [1]. Therefore, efficiently retrieving useful information from large digital text datasets has become a hotspot for research. The mainstream opinion on resolving the issue lies on mimicking real libraries. That is, classifying digital text documents to enable easy future retrieval. As a result, a new field called text categorization emerged. Text categorization is the study of classifying text documents by assigning them to one of the pre-defined classes. Many researchers have put efforts on text categorization and their results have been used in various domains such as webpage classification [2-4], sentiment identification [5,6], spam filtering [7-9], and search result disambiguation [10,11].

A typical text classification process usually involves four steps - feature extraction, feature selection, feature transformation, and classification [12]. Text extraction step is responsible for extracting useful information from the text corpus and converting them into appropriate feature representation and format that can be fed into subsequent steps to conduct text classification. Feature selection step selects relevant features and drops irrelevant features from the feature set constructed by the feature extraction steps in the text extraction step according to some feature selection metric. Feature transformation takes on the responsibility of making further processing on the features chosen by the feature selection step to make them be ready to be fed into the classification step. The processing made in the feature transformation step includes feature space transformation, dimension reduction, centering, standardizing, etc. Finally, the classification step performs text classification using the data produced at the end of the feature transformation step. It should be noted that the second and third steps are not mandatory for a text classification model since the minimal tools needed to perform text classification is a feature extractor that extracts information from the corpus and a classifier that utilizes the extracted information to perform text classification.

kNN is a well-known machine learning classifier that is a considered better for text categorization than other famous classifiers such as Naive Bayes and Term Graph [13]. It predicts the label of a given sample according to the result of the majority voting of the labels among its *k* nearest neighbors in the feature space. However, there are several issues plagued kNN, and the one that is considered of much importance is kNN's inability to detect local data structures. Conventional kNN finds a given test sample's *k* nearest neighbors by computing the Euclidean distances between the test sample and every training example, and then selecting the top k training examples with the closest distances. This approach could run into trouble when the given test sample is near class boundaries, the sample space is sparse, or some other complex and subtle situations occur. Figures 1-1, 1-2 and 1-3 demonstrate three situations in which local data structures cannot be detected by conventional kNN. In Figure 1-1, the given test sample is near the class boundary of a binary classification problem. By looking at the graph, a natural observation that the test sample belongs to circular class could be made. However, if kNN is used to make prediction, different results will be obtained. If *k* is set to 5, kNN would predict circular class for the test sample, but if *k* is set to 7, kNN would predict triangular class. In Figure 1-2, the feature space is sparse. It would be intuitive to predict the test sample belonging to circular class by looking at the graph. However, kNN would predict triangular class for both $k = 3$ and $k = 5$. In Figure 1-3, kNN is used to perform binary classification on imbalanced dataset. Although it can be observed directly from the graph that the given test sample is very likely to belong to circular class, the majority rule used by kNN will always result in the prediction of triangular class for any $k \geq 7$. Therefore, as illustrated by Figures 1-1 to 1-3, kNN is unable to detect local data structures.

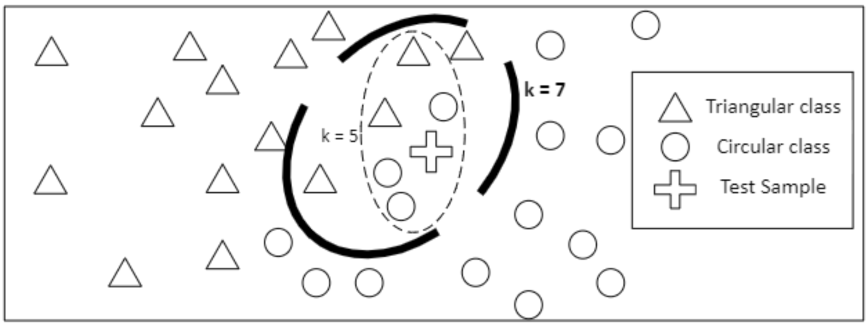

Figure 1-1. Binary classification using kNN at class boundaries

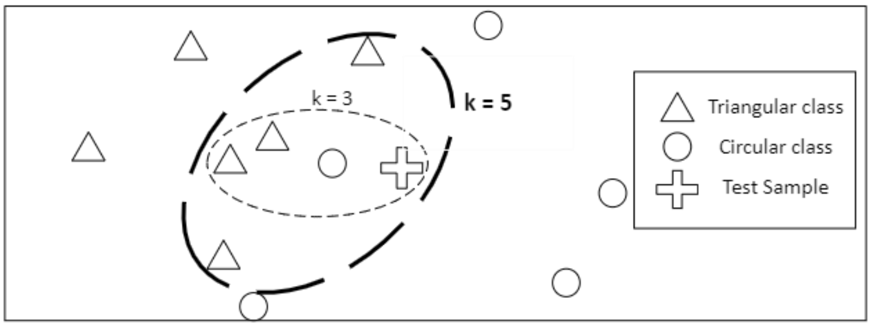

Figure 1-2. Binary classification using kNN in sparse feature space

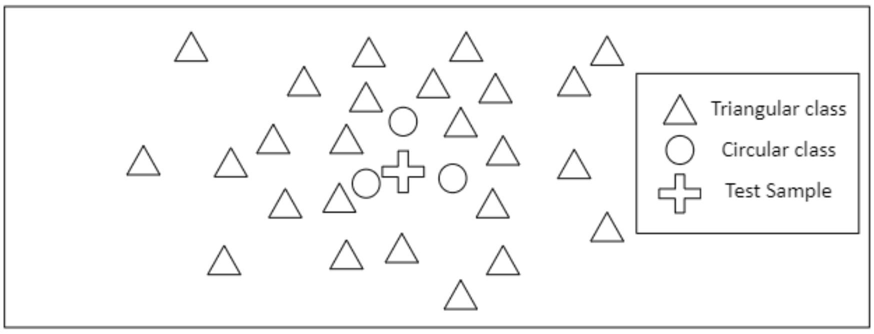

Figure 1-3. Binary classification using kNN in imbalanced dataset

Field researchers have been putting their efforts on solving this deficiency of kNN, and correlation matrix kNN (CM-kNN) was proposed [14]. CM-kNN constructs a correlation matrix between the entire training set and test set. The correlation matrix is used as a weight matrix used by the training set to approximate the test set. By investigating entry values of the correlation matrix, the weight of every training sample in approximating each test sample are known. Therefore, local data structures are detected by CM-kNN since the level of similarity between each pair of training and test samples can be obtained from the correlation matrix. However, since CM-kNN uses least squares loss function to compute the correlation matrix, and the correlation matrix is computed based on the entire training set, it suffers from global outliers that have relatively strong positive correlations with the test sample under consideration the training samples that are collinear to the test sample under consideration. Figures 2-1 and 2-2 demonstrate how those global outliers and collinear training samples affect CM-kNN and lead to misleading results. In Figure 2-1, the global outlier has extremely large feature values for those features that the test sample has positive values. As a result, the fitted weight of the global outlier in approximating the test sample is 0.8, which represents a relatively strong positive correlation. In Figure 2-2, a collinear training example whose feature values are roughly two times as large as those of the given test sample is given. Thus, as indicated by the graph, the fitted weight of the training sample is 2, which indicates a string positive correlation. However, due to the doubled feature values, the collinear training example is unlikely to be a member of the k nearest neighbors of the test sample.

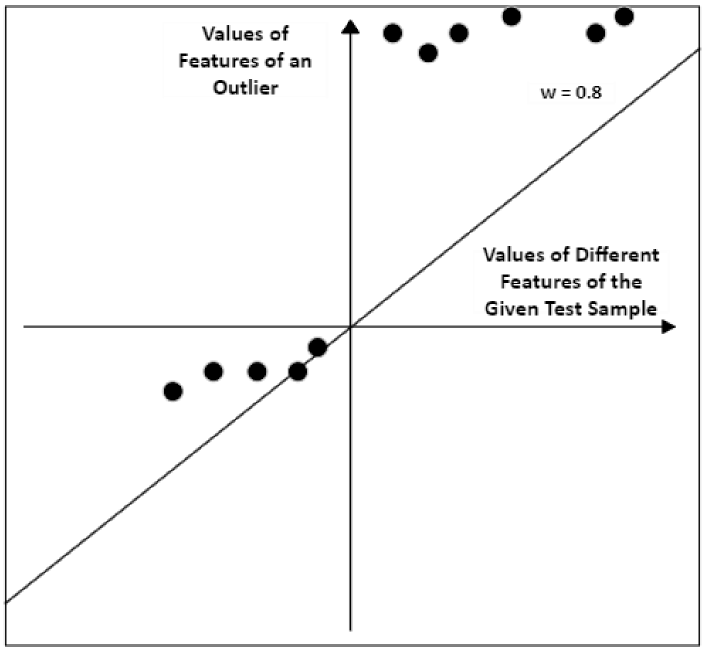
Values of Features of an Outlier
w = 0.8
Values of Different Features of the Given Test Sample

Figure 2-1. A global outlier with relatively large positive correlation with the given test sample

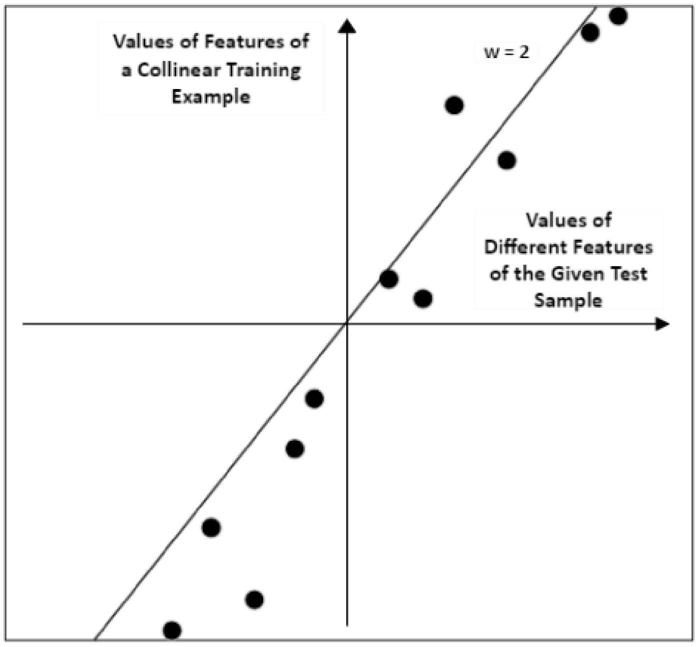
Values of Features of a Collinear Training Example
w = 2
Values of Different Features of the Given Test Sample

Figure 2-2. A training sample that is highly collinear to the given test sample

To resolve CM-kNN's sensitivity to global outliers and collinear training examples, this paper proposes local correlation vector kNN (Local-CorVec-kNN), which computes a local correlation vector between each test sample and its k nearest neighbors. Since Local-CorVec-kNN considers only the k nearest neighbors of a given test sample when computing the correlation vector, most outliers and collinear training samples have been filtered out and will not correspond to any entry in the correlation vector. As a result, Local-CorVec-kNN not only detects local data structures, but also is robust to global outliers and collinear samples. Furthermore, as individual correlation vector is computed for each test sample by Local-CorVec-kNN, the complex and time-consuming $L_{2,1}$-norm regularization and LPP regularization used in CM-kNN become unnecessary and can be dropped. Therefore, time costs are saved.

As mentioned before, high dimensionality is the major problem for text classification. There are many existing approaches to reduce feature space dimensionality, and PCA is one of the most widely applied and effective one. Therefore, the novel method proposed by this paper combines PCA and Local-CorVec-kNN to conduct text classification. Besides local data structures, collinearity, global outliers, and high dimensionality, the novel method also tackles other problems, including choosing different k values for different test samples, weighted voting, and imbalanced datasets.

The rest of the paper is organized as follows. Related works are reviewed in section II. Local-CorVec-kNN is introduced in Section III, and subsections are used to explain and analyze the detailed implementation of each part of the proposed method. Finally, we draw our conclusions and remarks in Section IV.

2. Related works

A text classification method consists of four consecutive parts. These are feature extraction, feature selection, feature transformation, and classification. Text extraction extracts information from raw text documents and converts it to features that can be used by a classifier. The extraction process is based on vector space model, that is, it views a text document as a dot in the N-dimensional feature space [15]. The mainstream approach for constructing the N-dimensional feature space is bag of words (BoW). In BoW, a term-document matrix is constructed to store useful information in different dimensions about a text document. The number of rows, *D*, of a term-document matrix represents the number of text documents in the corpus, and the number of columns, *N*, represents the number of unique words (terms) in the corpus. The entry value at the *i*-th row and *j*-th column of a term-document matrix represents the weight of term *j* in document *i*. Unlike the widely agreed opinion on using BoW to use the form of a term-document matrix to store useful information of a text document, various weight determination schemes have been proposed for computing the importance of a specific term to a specific document, and until now there is no clear winner. One of the simplest weight determination schemes uses term frequency (*tf*) to quantify the importance of a term to a particular document. In *tf*, the weight of term *j* in document *i* is the number of occurrences of the term in that document. However, due to the variable document length, terms that occur often in long documents may dominate the terms that only occur in short documents in the term-document matrix [16]. One solution to this problem is to normalize *tf* by document length to make every term weight in the matrix is on the same scale. Another solution is to use document frequency (*df*). *Df* only considers whether a term is present in a document or not. Thus, the *df* of a term is equal to the number of documents in the corpus that the term occurs. However, several facts worth noting before using *df*. Firstly, terms that occur frequently in all classes are considered less important, because it is hard to use them to discriminate documents from different classes. Secondly, terms that occur frequently in some classes are considered more important, because their weights are positively correlated with specific classes, and this information could be used to identify documents from those classes [17]. In addition, most corpora are highly skewed [18], so terms that occur frequently in larger classes usually dominate terms occur frequently in smaller classes. As a result, inverse document frequency (*idf*) was proposed to balance the weight differences cased by the differences of class sizes. Finally, as suggested by effect models, it looks reasonable to consider both the document level importance with the class level importance of a term. Therefore, term frequency-inverse document frequency (*tf-idf*) was proposed [19]. Besides *tf-idf*, many other terms weight determination schemes were also proposed by field workers. Examples of them include Bi-Normal Separation (BNS) [20] and information gain (IG) [21].

Feature selection increases classification accuracy and efficiency [22] by finding the minimal subset of features that produces classification performance that is as good as that of using all features [23]. Feature selection methods can be split into three categories - filter, wrapper, and embedded methods. Filter-based feature selection methods assign ranks to features according to some ranking criterion. Features with higher ranks will be selected over features with lower ranks. Representative filters include information gain (IG), odds ratio (OR), and Chi-square (CHI2). Wrappers, compared with filters, uses a learning algorithm to iteratively evaluate the performance of candidate feature subsets the until the (sub)optimal is found [24]. Wrappers could be further divided into three subcategories - exponential complexity methods, population-based approaches, and genetic algorithms (GAs). Embedded methods, on the other hand, are classifier specific because they do not separate feature selection and the classification, but combinedly conduct the two steps instead [25]. An embedded method iteratively feeds candidate feature subsets to the classifier to train different models. And those subsets correspond to models with better performance get favored. In general, filter is computationally less expensive than the other two categories of feature selection metrics because of its simple implementation and no incurring of other algorithms. Therefore, filter is the most popular category among the three [26].

In the field of text categorization, despite that feature selection removes many noisy and irrelevant features, feature space dimensionality is still overwhelming after feature selection, especially for large datasets. This is due to the nature of sparsity of feature space of text datasets [27]. Thus, it has become a necessity to transform the feature space of text documents into a computationally more feasible format before performing classification. Various feature transformation methods have been proposed, and most of them accomplish the goal of generating better feature format via dimension reduction. Dimension reduction is usually achieved by projecting the original feature space onto a lower dimensional subspace. Typical dimension reduction techniques for text categorization include latent semantic indexing (LSI) and PCA. Both LSI and PCA are linear in that all the direction vectors of the bases of the new subspaces produced by LSI and PCA are linearly spanned by vectors of the original feature spaces [28]. In other words, if LSI or PCA are used, there exists a covariance matrix such that the matrix multiplication of the covariance matrix and the new feature set representation is equal to the original feature set representation. Specifically, LSI takes on the assumption that terms which are close in meaning will appear in similar text documents [29]. Therefore, by treating rows of term-document matrix as terms and columns as documents, LSI performs singular value decomposition (SVD) to generate a new matrix that has fewer number of rows and keeping the relative relationships among the columns of the term-document matrix. The cosine value of the angle between two column vectors of the new matrix is to measure the corresponding text documents' similarity. Specifically, two documents are very similar if the cosine value is close to 1 and are quite different if the cosine value is close to 0. Compared with LSI, PCA does not take on any assumption and focuses on the geometrical distribution of the feature set of text documents. By setting the value of a hyper-parameter *p*, PCA compresses the original feature space to a p-dimensional subspace. The first direction of the subspace is the direction in the original space that possesses the largest variation of data. The second direction of the subspace is the direction in the original space that has the second largest variation of data, while it is orthogonal to the first direction. The remaining p - 2 directions of the subspace are determined in a similar manner.

Several well-known machine learning classifiers such as support vector machines (SVMs), kNN, Naive Bayes, and decision trees has been applied to text categorization and proved to have good performance. SVM and kNN are considered the better ones among them [30]. In a binary classification setting, a SVM explores the training data to find the decision surface that maximizes the gap between two classes. Gap is defined as the sum of distances from the two closest data points of different classes to the decision surface. Thus, class labels of unseen data are determined by SVM according to the decision surface obtained in the training phase. In multi-class classification, SVM also works by adopting the one-against-all setting [31]. In an one-against-all setting, SVM treats the class of consideration as the positive class and all the remaining classes combinedly as the negative class. Then, a binary SVM could be trained to find the decision surface for the positive class. This process could be repeatedly performed to train multiple binary SVMs to find out the decision surface for every class. kNN, on the hand, does not have a training phase. kNN determines the class label of a test sample by conducting a majority voting of the class labels among the k nearest neighbors of the sample. For a given test sample, a neighbor is a training sample that is near to it in the feature space. A common measure of nearness used by kNN is Euclidean distance. To make interpretations straightforward and computations simple, Euclidean distance is adopted as the distance measure metric used in this paper.

3. PCA-Local-CorVec-kNN

As mentioned before, a text classification method can be divided into four steps - feature extraction, feature classification, feature transformation, and classification. Each part has its own tasks on classifying text documents. In this paper, the proposed method used *tf-idf* and stopword removal in the feature extraction step, CHI2 feature selection metric in the feature selection step, PCA in the feature transformation step, and Local-CorVec-kNN in the classification step. Figure 4 illustrates the overview of the implementation of PCA-Local-CorVec-kNN. Mathematical notations and detailed implementation of each part of the proposed method are explained in the following subsections.

3.1. Notations

In this study, we use boldface uppercase characters to denote matrix; boldface lowercase characters to denote vectors; plain italic lowercase characters to denote scalers. For a matrix $\mathbf{X} = [x_{ij}]$, $\mathbf{x}^i$ is used to denote its ith row and $\mathbf{x}_j$ is used to denote its jth column. $\bar{\boldsymbol{x}}$ is used to denote the mathematical mean of vector **x**. We denote the term-document matrix obtained in the feature extraction step by **X**, with the row vectors $\mathbf{x}^i$, $i = 1, 2, ..., d$, representing the documents in the training set, and column vectors $\mathbf{x}_j$, $j = 1, 2, ..., n$, representing features extracted by *tf-idf*. It follows at the end of the feature extraction step, the number of rows of the term-document matrix is *d*, and the number of columns is *n*. Let $\widetilde{\mathbf{X}}$ denote the test set, and let *t* denote the number of documents in the test set.

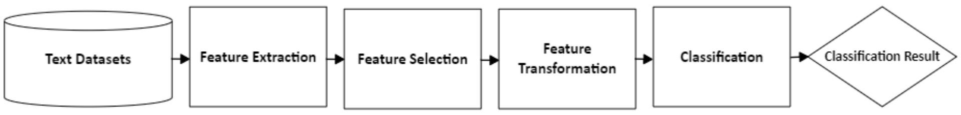

Figure 2. Overview of the implementation of PCA-Local-CorVec-kNN

3.2. Feature extraction

In the feature extraction step, *tf-idf* is used to extract features from the corpus. Common English stopword list is applied to remove terms that occur too frequently in all classes. Examples of the terms from the stopword list are *the*, *of*, and *for*, which are ubiquitous in all text documents and so they have no or little semantic contributions to text document discrimination. At the end of the feature extraction step, the training set is represented by $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & & x_{2n} \\ \vdots & & \ddots & \vdots \\ x_{d1} & x_{d2} & \cdots & x_{dn} \end{bmatrix}$, where x_{ij}, $1 \le i \le d$, $1 \le j \le n$, represents the *tf-idf* value of term *j* in document *i*.

3.3. Feature selection

CHI2 feature selection metric is used in the feature selection step to remove noisy and irrelevant features. CHI2 conducts a Chi-square test on each feature for each class [32]. A large χ^2 value in the Chi-square test imples a strong correlation between the feature and the class, and therefore, the feature is more likely to be selected as a feature for that class. Let n' denote the number of terms after feature selection and let **X**' denote the *tf-idf* representation of the training set after feature selection. Then we have $\mathbf{X}' = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n'} \\ x_{21} & x_{22} & & x_{2n'} \\ \vdots & & \ddots & \vdots \\ x_{d1} & x_{d2} & \cdots & x_{dn'} \end{bmatrix}$, where $n' \leq n$.

3.4. Feature transformation

In the feature transformation step, PCA is performed. PCA is a widely applied dimension reduction method. By setting the value of the hyper-parameter *p*, PCA reduces the dimensionality of the feature space obtained at the end of the feature selection step to *p*. Specifically, the original feature space is projected onto a *p*-dimensional feature space. The *p* direction vectors of the basis of the new feature space exactly match the p directions of the original feature space with the largest variations, and they are pairwise orthogonal to ensure that the new feature space is exactly *p* dimensional. Therefore, while PCA effectively reduces feature space dimensionality, it also keeps as much of the variation information of the dataset as possible.

Note that PCA uses training set's sample variances to produce the new feature space, and sample variances are computed using sample means. Thus, we can simplify notations of subsequent steps if we center every column of **X** using the corresponding column mean. To be specific, centering is done by subtracting the corresponding column mean from each column of **X**. Let **C** denote the resultant matrix after centering. Then we have

$$\mathbf{C} = \boldsymbol{X} - \left[\overline{\boldsymbol{x}}'_1\, \overline{\boldsymbol{x}}'_2 \ldots \overline{\boldsymbol{x}}'_{n'-1}\, \overline{\boldsymbol{x}}'_n\right] = [\boldsymbol{x}'_1 - \overline{\boldsymbol{x}}'_1 \quad \boldsymbol{x}'_2 - \overline{\boldsymbol{x}}'_2 \quad \ldots \quad \boldsymbol{x}'_{n'-1} - \overline{\boldsymbol{x}}'_{n'-1} \quad \boldsymbol{x}'_n - \overline{\boldsymbol{x}}'_n]$$

Then let **Z** denote the orthonormalized basis of the new feature space obtained by PCA and **W** denote the new representation of the training data using the new basis **Z**. It follows that **Z** is of size *p* times *n*, and **W** is of size *d* times *p*. Since PCA is a linear feature transformation method, we have

$$\boldsymbol{C} = \boldsymbol{W}\boldsymbol{Z} \tag{1}$$

Since $\boldsymbol{Z}$ is an orthonormal basis, its column vectors are pairwise orthogonal. That is,

$$\boldsymbol{Z}\boldsymbol{Z}^T = \boldsymbol{I}_p \tag{2}$$

By multiplying $\boldsymbol{Z}^T$ on both sides of equation (1), we get

$$\boldsymbol{C}\boldsymbol{Z}^T = \boldsymbol{W}\boldsymbol{Z}\boldsymbol{Z}^T \tag{3}$$

Then by inserting equation (2) into equation (3), we can get expression of $\boldsymbol{W}$

$$\mathbf{CZ}^{\mathrm{T}}=\mathbf{WI}_{\mathrm{p}} \Rightarrow \mathbf{W}=\mathbf{CZ}^{\mathrm{T}} \tag{4}$$

There is one thing left to be done before conducting PCA – determination of the value of the hyper-parameter *p*. Recall that PCA tries to keep as much of the variance information of the dataset as possible. Thus, it is natural to use the percentage of original data variation being kept by the new data representation produced by PCA as a measure of goodness of different *p* values. For example, if we want PCA to produce a new data representation that preserves 90% of the variation within the original representation of a dataset and want to keep the new feature space as small as possible, we will choose the smallest *p* value such that the ratio of the summation of column variances of the two data representations is at least as large as 90%. That is,

$$p = min\{p: \frac{\sum_{j=1}^{j=p}\sum_{i=1}^{i=D} w_{ij}^2}{\sum_{j=1}^{j=N'}\sum_{i=1}^{i=D} c_{ij}^2} \geq 90\%\} = min\{p: \frac{\sum_{j=1}^{j=p}\|\boldsymbol{w}_j\|^2}{\sum_{j=1}^{j=N'}\|\boldsymbol{c}_j\|^2} \geq 90\%\} = min\{p: \frac{\|\boldsymbol{W}\|^2}{\|\boldsymbol{C}\|^2} \geq 90\%\} \text{ Subject to } p \in [1, n'-1] \tag{5}$$

After *p* is determined, the orthonormal basis $\boldsymbol{Z}$ can be formed. Then back to the problem of text classification, according to equation (4), after the original training set representation **C** is obtained in the feature selection step, we could compute the new training set representation **W**.

3.5. Classification

In the classification step, Local-CorVec-kNN is applied. Since a bunch of pre-processing techniques were applied to the training data, the same set of techniques should be applied to the test data before running kNN. Specifically, the test set should adopt the same set of features as that of the training set at the end of the feature selection step. Then, centering should be applied to the test set using the column means of the training set. Next, PCA is applied to the test set to get the test set's new representation using the orthonormal basis $\boldsymbol{Z}$, which was obtained from the training set.

As discussed before, one major problem faced by conventional kNN is its inability to detect local data structures. One solution to this is to use CM-kNN, which computes a correlation matrix to and uses it to reconstruct the entire test set using the entire training set. Each column of the correlation matrix stores the contributions of every training sample to reconstructing a given test sample. Therefore, for a given test sample, local data structures could be detected by investigating the corresponding column vector in the correlation matrix [14]. Since CM-kNN is robust to local data structures, it is robust to sparse feature space. Note that one nature of text datasets is the sparsity of its feature space [27]. Therefore, CM-kNN fits text datasets better than conventional kNN. However, CM-kNN is sensitive to the training samples that are collinear to the test sample under consideration and global outliers that have relatively strong positive correlations with the test sample under consideration. To deal with this dilemma of CM-kNN, this paper proposes Local-CorVec-KNN. Compared with CM-kNN, instead of constructing a large correlation matrix for the entire training set and test set, Local-CorVec-kNN constructs a small correlation vector of size *1* times *k* for each test sample and its k nearest neighbors. Thus, for a given test sample, the corresponding correlation vector is bound to k training samples, and this significantly reduces the number of training samples being considered. Consequently, the possibility of encountering collinear training samples and global outliers gets largely reduced.

It should be noted that Local-CorVec-kNN uses a least squares loss that is smooth and convex, a $l_2 - norm$ regularizer that generates unique solution, and a $l_1 - norm$ regularizer that generates sparsity within the correlation vector.

Let $\widetilde{\boldsymbol{x}}$ denote a test sample after pre-processing and $\boldsymbol{b}$ denote the test sample's k nearest neighbors. Then we have

$$\widetilde{\boldsymbol{x}} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}^1 \\ \boldsymbol{b}^2 \\ \vdots \\ \boldsymbol{b}^k \end{bmatrix} = [\boldsymbol{b}_1 \quad \boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{b}_{k-1} \quad \boldsymbol{b}_p] = \begin{bmatrix} b_{11} & \cdots & b_{1\mathrm{p}} \\ \vdots & \ddots & \vdots \\ b_{\mathrm{k}1} & \cdots & b_{kp} \end{bmatrix}$$

where the row vectors $\mathbf{b}^{\mathrm{i}} = (b_{i1}\ b_{i2}\ \cdots\ b_{ip})^T$, $i = 1,2,\dots,k$, represents the i-th nearest neighbor of $\widetilde{\boldsymbol{x}}$, and the column vectors $\mathbf{b}_{\mathrm{j}} = (b_{1j}\ b_{2j}\ \cdots\ b_{3j})^T$, $j = 1,2,\dots,p$, represents the j-th feature of the k nearest neighbors of $\widetilde{\boldsymbol{x}}$.

Then a correlation vector $\boldsymbol{m}$ could be constructed between the given test sample $\widetilde{\boldsymbol{x}}$ and its k nearest neighbors $\boldsymbol{B}$ to build a numeric view of local data structures. That is, we use the matrix-vector product $\mathbf{B}^{\mathrm{T}}\boldsymbol{m}$ to approximate $\widetilde{\boldsymbol{x}}$, with each of $\boldsymbol{m}$'s entries representing each of $\widetilde{\boldsymbol{x}}$'s k nearest neighbors' contribution to approximating $\boldsymbol{x}$. Thus, we can build a least squares objective in search for the optimal $\boldsymbol{m}$ that achieves the best approximation performance:

$$minimize\ f(\boldsymbol{m}) = ||\boldsymbol{B}^T\boldsymbol{m} - \widetilde{\boldsymbol{x}}||^2,\ \mathbf{m} = \in \boldsymbol{R}^k \qquad (6)$$

where $\mathbf{m} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{bmatrix}$, with m_i, $i = 1,2,\dots,k$, represents the weight of the i-th nearest neighbor of $\widetilde{\boldsymbol{x}}$ in approximating $\widetilde{\boldsymbol{x}}$ using $\boldsymbol{B}$.

Note that $f(\boldsymbol{m})$ is of quadratic form, so it is a smooth and convex function. Thus, it is minimized by letting its gradient equal to $\mathbf{0}$:

$$\nabla f(\boldsymbol{w}) = 2\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{m} - \widetilde{x}) = \mathbf{0} \;\; => \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{m} = \boldsymbol{B}^{\boldsymbol{T}}\widetilde{\boldsymbol{x}} \;\; => \; \boldsymbol{m} = (\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}})^{-\mathbf{1}}\boldsymbol{B}^{\boldsymbol{T}}\widetilde{\boldsymbol{x}} \tag{7}$$

However, the $\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}}$ in equation (7) is not guaranteed to be invertible. A common solution to this issue is adding a $l_2 - norm$ regularizer to objective function (6):

$$minimize\, f(\boldsymbol{m}) = ||\boldsymbol{B}^T\boldsymbol{m} - \widetilde{\boldsymbol{x}}||^2 + \lambda||\boldsymbol{m}||^2,\ \mathbf{m} \in \mathbf{R}^\mathrm{k},\ \lambda \in \mathbf{R}^+ \tag{8}$$

where λ representing the amount of the penalty put on the entry values of $\boldsymbol{m}$, with big λ representing large penalty and small λ representing small penalty.

The linear system is still smooth and convex after adding the $l_2 - norm$ regularizer. However, if we let its gradient equal to $\mathbf{0}$, it will always generate a unique solution with the $l_2 - norm$ regularizer added

$$\nabla f(\boldsymbol{w}) = 2\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{m} - \widetilde{\boldsymbol{x}}) + 2\lambda\boldsymbol{m} = \mathbf{0}$$

$$=> (\boldsymbol{B}\boldsymbol{B}^T + \lambda\boldsymbol{I})\boldsymbol{m} = \boldsymbol{B}^{\boldsymbol{T}}\widetilde{\boldsymbol{x}} \;\; => \boldsymbol{m} = (\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}} + \lambda\mathbf{I})^{-\mathbf{1}}\boldsymbol{B}^{\boldsymbol{T}}\widetilde{\boldsymbol{x}} \tag{9}$$

where $\boldsymbol{B}\boldsymbol{B}^{\boldsymbol{T}} + \lambda\mathbf{I}$ is guaranteed to be invertible due to the adding of the term $\lambda\mathbf{I}$.

However, the added penalty term $\lambda||\boldsymbol{m}||^2$ in objective (8) only decreases the weights of less relevant or irrelevant neighbors, but not selecting neighbors. To enable neighbor selection, a $l_1 - norm$ regularizer is added to the objective to set the weights of those neighbors to be dropped to zero

$$\mathrm{min}imize\, f(\boldsymbol{m}) = ||\boldsymbol{B}^T\boldsymbol{m} - \widetilde{\boldsymbol{x}}||^2 + \lambda_1||\boldsymbol{m}||^2 + \lambda_2||\boldsymbol{m}||,\ \mathbf{m} \in \mathbf{R}^\mathrm{k},\ \lambda_1, \lambda_2 \in \mathbf{R}^+ \tag{10}$$

By adding the $l_2 - norm$ regularizer and the $l_1 - norm$ regularizer, not only a unique correlation vector could be obtained for each test sample, but also the most relevant neighbors among the k nearest neighbors of a test sample will be selected. In the field of statistical learning, objective function (10) is called an elastic net regression.

Since least square loss function is convex, $l_1 - norm$ regularizer and $l_2 - norm$ regularizer are convex, and the summation of convex functions are convex, objective function (10) is convex. However, it is not smooth at $\boldsymbol{m} = \mathbf{0}$ because of the $l_1 - norm$ regularizer. Therefore, we can solve it by using partial derivatives rather than gradient. So, by expanding objective function (10), we have

$$minimize\, f(\boldsymbol{m}) = ||\boldsymbol{B}^T\boldsymbol{m} - \widetilde{\boldsymbol{x}}||^2 + \lambda_1||\boldsymbol{m}||_2^2 + \lambda_2||\boldsymbol{m}||_1$$

$$= \sum_{i=1}^{d}\left[\sum_{j=1}^{k}\left(B_{ij}m_i - \widetilde{\boldsymbol{x}}_i\right)\right]^2 + \lambda_1\sum_{i=1}^{k} m_i^2 + \lambda_2\sum_{i=1}^{k}|m_i|\,, \lambda_1, \lambda_2 \in \mathbf{R}^+ \tag{11}$$

or, equivalently,

$$minimize\, f(m_i) = \left[\sum_{j=1}^{k}\left(B_{ij}m_i - \tilde{x}_j\right)\right]^2 + \lambda_1 m_i^2 \pm \lambda_2|m_i|, \lambda_1, \lambda_2 \in \boldsymbol{R}^+, i = 1,2,\dots,k \tag{12}$$

where $+\lambda_2$ is used when the value of m_i is positive, and $-\lambda_2$ is used when the value of m_i is negative.

Then we can take partial derivatives on objective function (12) to get

$$\frac{\partial f}{\partial m_i} = 2m_i\sum_{j=1}^{k} B_{ij}(B_{ij} - \tilde{x}_j) + 2\lambda_1 m_i \pm \lambda_2,\ \ \lambda_1, \lambda_2 \in \mathbf{R}^+, \mathrm{i} = 1,2,\dots,\ \mathrm{k} \tag{13}$$

where $+\lambda_2$ is used when the value of m_i is positive, and $-\lambda_2$ is used when the value of m_i is negative.

By letting partial derivative (13) equal to 0, we get

$$\frac{\partial f}{\partial m_i} = 0 \;=>\; 2m_i \sum_{j=1}^{k} B_{ij}(B_{ij} - \tilde{x}_j) + 2\lambda_1 m_i \pm \lambda_2 = 0$$
$$=>\; m_i\left[2\sum_{j=1}^{k}\left(B_{ij}^2 - B_{ij}\tilde{x}_j\right) + 2\lambda_1\right] = \mp\lambda_2$$
$$=>\; m_i = \mp\frac{\lambda_2}{2\sum_{j=1}^{k}\left(B_{ij}^2 - B_{ij}\tilde{x}_j\right)+2\lambda_1}, \lambda_1, \lambda_2 \in \mathbf{R}^+ \tag{14}$$

where a positive sign on the right hand side of equation (14) is resulted from a negative m_i, and a negative sign is resulted from a positive m_i, and if neither of $m_i = \frac{\lambda_2}{2\sum_{j=1}^{k}\left(B_{ij}^2 - B_{ij}\tilde{x}_j\right)+2\lambda_1}$ and $m_i = -\frac{\lambda_2}{2\sum_{j=1}^{k}\left(B_{ij}^2 - B_{ij}\tilde{x}_j\right)+2\lambda_1}$ makes partial derivative (13) equal to 0, then it means that objective (12) is dominated by the $l_1 - norm$ regularizer and is minimized at the indifferentiable angular turning point, which is at $m_i = 0$. Therefore, the solution of equation (14) is $m_i = 0$ if neither of the expression on the right-hand side of equation (14) make objective (13) equal to 0.

3.6. Quantitative analysis of time cost complexity

If PCA is not performed and conventional kNN is used to classify text documents, the matrix for the training set would be of size *d*n'*. Then the cost of computing the kNN nearest neighbors of a single test sample would be O(*n'd*) since kNN computes the sum of the Euclidean distances of every feature between a training sample and the given test sample, and this process should be done for every test sample. Therefore, the overall time cost of running kNN for a test set of size *t* would be O(*n'dt*).

If PCA is used, we first need to determine the value of the hyper-parameter *p*. Suppose the percentage of variation of the original data representation that we want to preserve in the new feature space is known, then we need to run PCA for O(*$log_2(h)$*) times to ensure the optimal *p* value is found, where t is the number of digits after the decimal point that we want the proportion to be. This is done by performing a binary search on the interval *(0,1)*. For each value of *p*, we tried, O (*1*) time is needed to form the orthonormal *p*p* basis $\boldsymbol{Z}$. Then the new representation of the training set, $\boldsymbol{W}$, could be computed in O(*n'dp*) time according to equation (4). Accordingly, the time cost of performing PCA for the test set would be O(*tdp*). Then the time cost of running kNN on the new data representation would be O(*n'pt*).

If Local-CorVec-kNN is used, the cost of computing an entry of a correlation vector is O(*p*) according to equation (14). Thus, the cost of constructing correlation vectors for the entire test set would be O(*tkp*). Note that since the correlation vector is local, the value combination of λ_1 and λ_2 is local as well. Therefore, we empirically set $\lambda_1 = 1$, $\lambda_2 = 1$.

Therefore, the total time cost complexity of the proposed method would be O(*(n'+t)dp + n'pt + tkp*). This is of lower magnitude than O(*n'dt*) when the value of *t* is very large. Note that in real applications, we are usually given limited amount of data to train a model that makes predictions on very large datasets. This is especially true for text categorization due to the exponential growth of the amount of online digital text document resources. Hence, the proposed method not only deals with the issue of high feature space dimensionality and local data structures but can also be scaled to conduct text categorization for big data.

3.7. Discussion: Other Issues Tackled by PCA-Local-CorVec-kNN

Besides detecting local data structures and decreasing feature space dimensionality, other problems are also tackled by the proposed method. It has become an agreed opinion that different *k* values should be used for different test samples [33]. PCA-Local-CorVec-KNN can choose different *k* values for different test samples by dropping those neighbors corresponding to zero entry values in the correlation vector. In addition, correlation vector entries could be used to conduct weighted voting, since their values describe the extent of similarity of different neighbors to a given test sample. Finally, since the proposed method assigns different weights to different neighbors, neighbors from different classes are more likely to have small weights and thus, are more likely to be dropped. Therefore, even when there are many neighbors from the negative class lying around a given test sample, most of them will be detected and dropped in the end by PCA-Local-CorVec-kNN. Therefore, the proposed method is more robust to imbalanced dataset than conventional kNN.

4. Conclusion

The main contribution of this paper is the novel method for text classification – PCA-Local-CorVec-kNN. This method utilizes PCA to deal with the issue of high feature space dimensionality of text documents. The method also utilizes Local-CorVec-kNN to cope with local data structures. By using PCA in the feature transformation step, the feature space dimensionality of text documents gets effectively reduced, which in turn reduces the computing resources and running time complexity in the classification step. Local-CorVec-kNN could detect local data structures and is more robust to collinear training data samples and global outliers than CM-KNN. Furthermore, Local-CorVec-kNN also deals with other machine learning problems, including choosing different k for different samples, weighted voting, and imbalanced dataset.

Since we empirically choose the values of λ_1 and λ_2 for each test sample, it is necessary to find better methods for optimizing the two parameters in the future. In addition, we empirically choose CHI2 feature selection metric in the feature selection step. However, there might be some other feature selection metric that fits better with PCA-Local-CorVec-kNN instead of the CHI2 in this paper. Thus, theoretical analysis as well as comparative experiments need to be conducted to figure this out.

References

[1] H. Balinsky, A. Balinsky, S. J. Simske, “DocEng ’11: Proceedings of the 11th ACM symposium on Document engineering,” 175-184 (2005).
[2] H. Haleem, C. Niyas, S. Verma, A. Kumar, and F. Ahmad, "Effective probabilistic model for webpage classification," in *Emerging Trends in Computing and Communication*, 277-290 (2014).
[3] F. Shen, X. Luo, and Y. Chen, "Text classification dimension reduction algorithm for Chinese web Page based on deep learning," in *International Conference on Cyberspace Technology*, 451-456 (2013).
[4] X. Jin, Y. Li, T. Mah, and J. Tong, "Sensitive webpage classification for content advertising," in *ADKDD'07: Proceeding of the 1st international workshop on Data mining and audience intelligence for advertising*, 28-33 (2007).
[5] V. S. Shirsat, R. S. Jagdale, and S. N. Deshmukh, "Sentence level sentiment identification and calculation from news articles using machine learning techniques," in *Computing, Communication and Signal Processing*, 371-376 (2018).
[6] A. K. Dash, J. K. Rout, and S. K. Jena, "Harnessing twitter for automatic sentiment identification using machine learning techniques," in Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics, 507-514 (2015).
[7] W. Liu and T. Wang, "Indexed-based online text classification for SMS spam filtering," in *Journal of Computers*, vol, 5, No. 6, 844-851 (2010).
[8] O. Amayri and N. Bouguila, "A study of spam filtering using support vector machines," in Artificial Intelligence Review, vol. 34, 73-108, (2010).
[9] JM. G. Hidalgo, G. C. Bringas, E. P. Sanz, and F. C. Garcia, "Content based SMS spam filtering," in *DocEng'06: Proceedings of the 2006 ACM symposium on Document engineering*, 107-114, (2006).
[10] Y. Netzer, D. Gabay, M. Adler, Y. Goldberg, and M. Elhadad, "Ontology evaluation through text classification," in APWeb2009, WAIM 2009: Advances in Web and Network Technologies, and Information Management, 210-221 (2009).
[11] S. Schmidt, S. Schnitzer, and C. Rensing, "Text classification based filters for a domain-specific search engine," in Computers in Industry, vol. 78, 70-79 (2016).
[12] A. K. Uysal, "On two-stage feature selection methods for text classification," in *IEEE Access*, vol. 6, 43233-43251 (2018).

[13] V. Bijalwan, V. Kumar, P. Kumari, and J. Pascual, "KNN based machine learning approach for text and document mining," in International Journal of Database Theory and Application, vol.7, no.1, 61-70 (2014).

[14] S. C. Zhang, X. L. Li, M. Zong, X. F. Zhu, and D. B. Cheng, "Learning k for kNN classification," in ACM Transactions on Intelligent Systems and Technologies, vol. 8, no. 3, 1-19 (2017).

[15] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review", in *Journal on Wireless Communications and Networking* (2017(1)): 211.

[16] H. Moisl, "Data normalization for variation in document length in exploratory multivariate analysis of text corpora" in *The 6th International Conference on Informatics and Systems,* (2008).

[17] A. K. Uysal and S. Gunal, "A novel probabilistic feature selection method for text classification," in Knowledge Based Systems, vol. 36, 226-235 (2012).

[18] E. Frank and R. R. Bouckaert, "Naive Bayes for text classification with unbalanced classes," in *Knowledge Discovery in Databases: PKDD 2006*, 503-510 (2006).

[19] P. Tang and T. W. Chow, "Recognition of word collection habits using frequency rank ratio and inter-term intimacy," in *Expert Systems with Applications*, vol. 40, iss. 11, 4301-4314 (2013).

[20] G. Forman, "Choose your words carefully: an empirical study of feature selection metrics for text classification," in PKDD2002: *Principles of Data Mining and Knowledge Discovery*, 150-162 (2002).

[21] D. D. Lewis and M. Ringuette, "Comparison of two learning algorithms for text categorization," in SDAIR'94: *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, (1994)

[22] Y. Li, and C. Chen, "Research on the feature selection techniques used in text classification," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 725-729 (2012)

[23] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," in *Journal of Machine Learning Research*, vol.5, 1205-1224 (2004)

[24] N. E. Aboudi and L. Benhlima, "Review on wrapper feature selection methods," in 2016 International Conference no Engineering & MIS (ICEMIS), 1-5 (2016).

[25] T. N. Lal, O. Chapelle, J. Weston, A. Elisseeff, "Embedded methods," in Feature Extraction, 137-165 (2006).

[26] A. K. Uysal, "An improved global feature selection scheme for text classification," in Expert Systems with Applications, vol. 42, 82-92 (2016).

[27] A. Rehman, K. Javed, H. A. Babri, and M. Saeed, "Relative discrimination criterion - a novel feature ranking method for text data," in Expert Systems with Applications, vol. 42, iss. 7, 3670-3681 (2015).

[28] I. K. Fodor, "A survey of dimension reduction techniques," *No. UCRL-ID-148494*, Lawrence Livermore National Lab., CA (US) (2002).

[29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," in *Journal of the American society for information science*, vol. 41, iss. 6 (1990).

[30] Y. Yang and X. Liu, "A re-examination of text categorization methods," in SIGIR'99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 42-49 (1999).

[31] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," in IEEE Transactions on Neural Networks, vol. 13, iss. 2, 415-425 (2002).

[32] H. Liu and R. Setiono, "Chi2: feature selection and discretization of numeric attributes," in *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, 388-391 (1995).

[33] S. C. Zhang, X. L. Li, M. Zong, X. F. Zhu, R. L. Wang, "Efficient kNN classification with different numbers of nearest neighbors," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5 (2018).