MATH 6380: DATA ANALYSIS

Classification of NIPS Conference Papers with Respect to High Frequency Key Words

Group memeber: Ye Rougang, Tan Chunxi, Han Ruijian
March 13, 2017

1 INTRODUCTION

The dataset we choose is the NIPS Conference Papers 1987-2015 dataset. The dataset is in the form of 11463×5812 matrix of word counts, containing 11463 words and 5811 NIPS conference papers (the first column contains the list of the words). Each column contains the number of times each word appears in the corresponding article.

After applying TF-IDF (term frequency-inverse document frequency) algorithm on the original dataset, we used two methods to classify NIPS conference papers by taking "new word frequency" as their features. The first one is LSA, i.e., latent semantic analysis, with the use of SVD (singular value decomposition) and K-means clustering. The second one is using SVC, i.e., support vector machine for classification problems.

2 HIGH FREQUENCY KEY WORDS

As there are many words recorded in the dataset which many of them don't hold valuable information like "abalone", "bat" and so on, we used the TF-IDF algorithm to process the original dataset.

Here TF-IDF is the abbreviation of "term frequency-inverse document frequency". We use $n_{i,j}$ to stand for the (i, j)-th element of the term-document matrix, which is the frequency of i-th word in the j-th document. The TF (term frequency) is defined as

$$TF_{i,j} = \frac{n_{i,j}}{\sum_{i=1}^{N} n_{i,j}},$$

where N is the total number of words in the matrix, i.e., the row number of the matrix. We use $m_{i,j}$ to indicate whether the i-th word is in the j-th document. If it is, $m_{i,j} = 1$. Otherwise, $m_{i,j} = 0$. The IDF (inverse document frequency) is defined as

$$IDF_{i,j} = \log \frac{d}{\sum_{j=1}^{d} m_{i,j}},$$

where d is the total number of documents in the matrix, i.e., the column number of the matrix. Then the TF-IDF of (i, j)-th element is defined as

$$TFIDF_{i,j} = TF_{i,j} \times IDF_{i,j}.$$

From the above definition, we can get the main idea of TF-IDF is decreasing the weight of words that appeared in many articles such as "abstract", "model" etc. Actually these words should not be our focus. So after processing by TF-IDF, words appearing frequently in some articles but not in many ones would get larger TF-IDF values.

After applying TF-IDF algorithm, values of words like "abstract", "model" is smaller compared with original frequency and actually "model" ranks the first in terms of frequency in the raw dataset. Yet words like "network", "kernel", and "spike" get larger values now. These words are indeed symbols and feature of machine learning articles. The complete result of TF-IDF is in the attachment.

Now we selected "real high-frequency" words from the original dataset, we used a python package *wordcloud* to generate the vivid wordcloud of top 100 high frequency words. The visualized wordcloud is generated with respect to the weight values computed by TF-IDF algorithm.

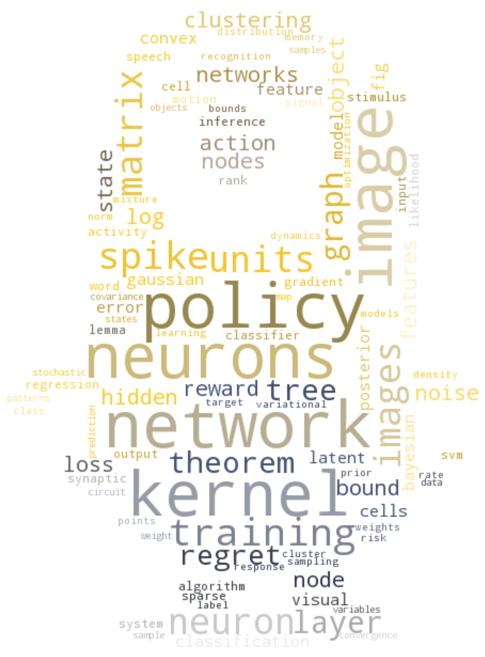

Figure 2.1: Word Cloud of Top 100 High Frequency Words

From the figure2.1, one can see that "network", "kernel", "neurons", "training" etc. are bigger words which signify their importance. Thus later we would take some of them as features to classify these NIPS papers.

3 CLASSFICAITON OF NIPS PAPERS

In this section, we use two methods to make the classification with respect to high frequency keywords we selected in the section 2. The first one is LSA, a classical method as for the term-document matrix, implemented by SVD and K-means. The second one is SVC, i.e., support vector classification.

3.1 LSA: SVD and K-means

Considering there are lots of meaningless words, we choose top 500 words from the ordered word-article dataset by TF-IDF as features of our classification here. Thus we would analyze the information from this new word-article matrix of the shape 500×5812.

3.1.1 SVD: singular value decomposition

Before we make the classification, we need to reduce the dimension of matrix. We use SVD here. For a word-document matrix, SVD is:

$$M = U\Sigma V,$$

where M is word-document matrix, U is the matrix about the information of word-word, V is the matrix about the information of document-document and Σ is the diagonal matrix with singular value. The largest 3 singular value we get is :

$$\sigma_0 = 103.12, \sigma_1 = 3.83, \sigma_2 = 1.05$$

and the rest singular value is smaller than 1.

However, with these three singular values, the classification we get is not as good as what we expected. With the reference–Latent Semantic Analysis (LSA) Tutorial, we know that for documents, the first dimension correlates with the length of the document. For words, it correlates with the number of times that word has been used in all documents. Both of these two parameters are not useful for our classification. It suggests us threw out the first dimension.

In the PCA (principle component analysis), we always do the centralization for the matrix first. With the consideration of centralization, we would be unwilling to threw out the first dimension. Yet here we give up the centralization since the centralization will instead change the original sparse matrix to a dense matrix which usually costs more efforts to analyze. Thus we did not use the largest singular value and take the other two singular values:

$$\sigma_1 = 3.83, \sigma_2 = 1.05$$

We use U^* to stand for second and third rows of matrix U and V^* to stand for second and third columns of matrix V and use S^* to stand for $\text{diag}\{\sigma_1, \sigma_2\}$. So we get 2 dimensional word vectors $U^* S^*$ and 2 dimensional document vectors $S^* V^*$.

3.1.2 K-means clustering

Now we have already got the lower dimension vectors of words and documents. We use K-means clustering to classify documents.

Table 3.1: Part of results of K-means clustering

label	words and documents
5	keywords: image, unit documents: 1988_78, 1989_14, 1993_65, 2002_124...
13	keywords: speech, word, input, output... documents: 1987_29, 1988_46, 1991_2, 1995_19...
32	keywords: loss, models, stochastic, gaussian, random... documents: 1987_9, 1988_45, 1989_5, 1991_6...

The KMeans algorithm divides a set of N samples into K disjoint clusters, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids". The K-means algorithm aims to choose centroids that minimize the *inertia* (sum of distances of samples to their closest cluster center):

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$$

Given an initial set of k means $m_1^{(1)}, ..., m_k^{(1)}$ (see below), the algorithm (also referred to as Lloyd's algorithm) proceeds by alternating between two steps:

- **Assignment step**: Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean.

  $$S_i^{(t)} = \{x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \forall j, 1 \leq j \leq k\}$$

  where each x_p is assigned into exactly one $S^{(t)}$.

- **Update step**: Calculate the new means to be the centroids of the observations in the new clusters.

  $$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

  Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares (WCSS) objective.

Here we consider selected data as sample data and try to figure out different clusters with different keywords described as features by K-means Clustering.We expect to use words as features to categorize the NIPS papers to 40 classes. As for the limited space, we just present a part of the result in the table 3.1. The whole result can be found in the attachment.

However, not all of the results are as good as expected. For example, the word "Monte" and "Carlo" are not in the same class and there are only words included in some classes. This is caused by firstly, the SVD lost some information of the raw data. In addistion, the K-means clustering didn't perform so well when handling too much data.

3.2 SVC:SUPPORT VECTOR CLASSIFICATION

As the machine learning is the cross field of computer science and statistics, it might be interesting to classify these NIPS papers to two classes: CS and statistics. Cause some articles tend to cover more stuff about algorithm which are closer to computer science, while other articles discuss more about regression, stochastic process that indicate they are related to statistics more.

We choose two groups of three words as features to decide which class a article should belong to. After analyzing the top 150 high frequency words processed by TF-IDF, we choose (each word followed by its order in the top 150 inside the parentheses)

- Computer Science: kernel(3), SVM(62), recurrent(137)
- Statistics: clustering(17), Bayesian(39), stochastic(96)

as 3 features to describe the two classes respectively. "Kernel" and "SVM" are just two high frequency words appeared in CS algorithms. "Recurrent" is always related to recurrent neural network, i.e., RNN which gets popular as a important network structure in the deep learning. We believe these three words can act as features of CS class. In the other hand, "clustering", "Bayesian" and "stochastic" are three representative words for the statistics class in the top 150 words.

We then use the sum of these six features as a filter to select out articles worth classifying to these two classes. In other words, we focus on 3865 papers whose the sum of numbers of these six words is greater than 1 because some papers don't contain any of these six words and might belong to other subfields. Dividing 3865 papers to 500 and 3365 papers, we take the first 500 articles as our training set and the rest 3365 papers as testing samples. As for giving labels {0,1} to the training set, the simple strategy is: when the sum of counts of "kernel", "SVM", "recurrent" is bigger than the sum of "clustering", "Bayesian" and "stochastic", then this article is labeled as a CS one. Otherwise the paper is labeled as a statistics one.

By using the python package *scikit-learn* (the code script can be found in the attachment), we carry out SVC with different kernels (linear kernel, RBF kernel, polynomial(degree 3) kernel). Taking "clustering" and "kernel" as two axises, we show the SVC result of classifying 600 test papers from testing set in the figure3.1. The red dots represent the statistics class and the blue dots represent the CS class. We assert the binary classification is good enough cause different kernels perform well to classify according to our training SVC models.

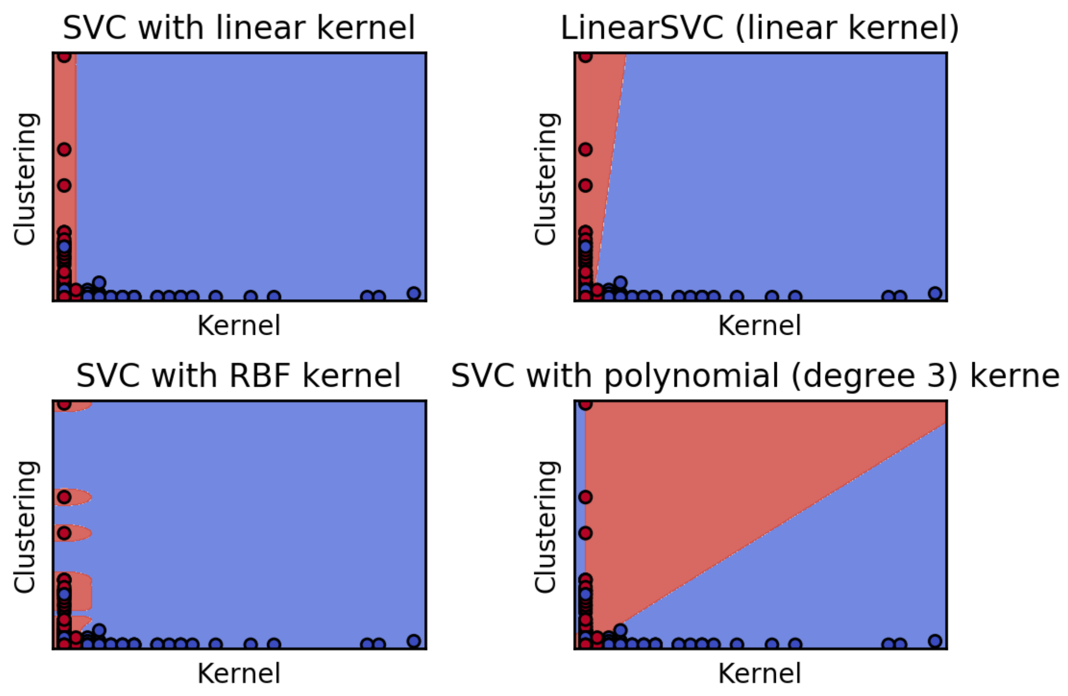

Figure 3.1: SVC with Different Kernels

4 Summary

Given this large-scale dataset containing 11463 words and 5811 NIPS conference papers in the form of a 11463 x 5812 matrix of word counts, we consider reducing dimension firstly and then try to classify those papers by their features(words).TF-IDF is the main way of lowering dimensions. According to the total frequency of word, TF-IDF selected top 500 high-frequency key words. And then we do classifications to those 500 words by 2 ways, LSA and SVC. The clustering results are showed above and in the attachment. We have considered other traditional methods to deal with high dimensional data, but they don't perform so well. For example, PCA will change sparse matrix to dense matrix and make it harder to handle. Even with LSA, the result is not as good as we expected. So it's suggested that using some machine learning algorithms suitable for processing high dimensional data.

5 Remark of contribution

Han Ruijian suggested to use TF-IDF algorithm to process the original dataset and thus to lower the dimension, then selected high frequency words and implemented

SVD of the new term-document matrix.

Tan Chunxi classified papers by using K-means clustering, summarized our work and built this LaTeX document.

Ye Rougang used word cloud to visualize high frequency keywords and classified papers using SVC.

Finally, Many thanks to our instructor Mr. Yao, who gives us the chance to deal with high dimensional dataset and the chance of this cooperation.