Classification of NIPS Conference Papers by SVD and Autoencoder

Group memeber: Ye Rougang, Tan Chunxi, Han Ruijian
October 12, 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 extracting features of papers by autoencoder which can be regarded as the non-linear extension of SVD. Finally, we compare our two methods from perspective of dependence.

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.

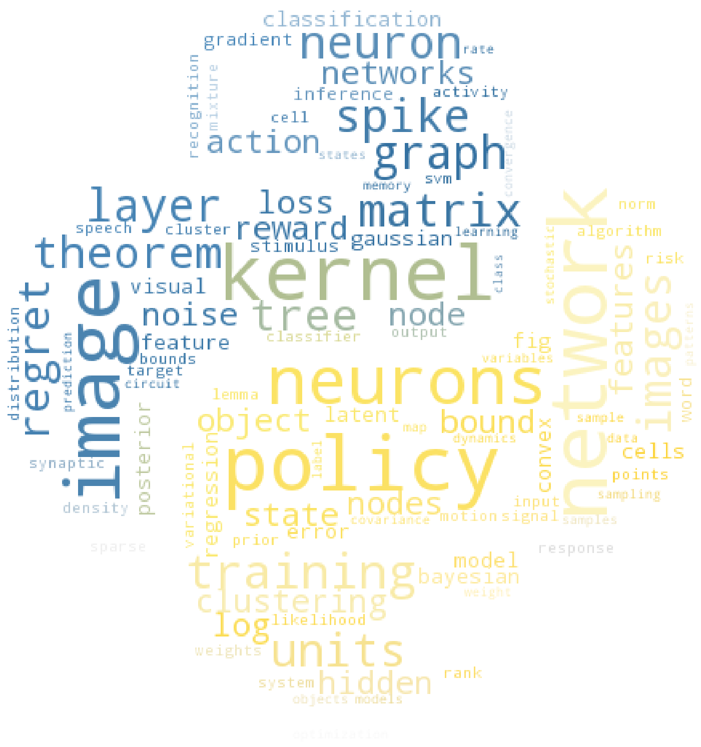

Figure 2.1: Word Cloud of Top 100 High Frequency Words

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 :

$$\sigma_0 = 3.83, \sigma_1 = 1.40, \sigma_2 = 1.04$$

and the rest singular value is smaller than 1.

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 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.

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 $m_1^{(1)}, \ldots, m_k^{(1)}$ (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.

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

Table 3.1: Part of results of K-means clustering 1

label	documents
0	1987_41: Neural Net and Traditional Classifiers William Y. Huang, Richard P. Lippmann 1988_6: Mapping Classifier Systems Into Neural Networks Lawrence Davis 1988_24: Applications of Error Back-Propagation to Phonetic Classification Hong C. Leung, et al. 1994_118: Unsupervised Classification of 3D Objects from 2D Views, Satoshi Suzuki et al. 2010_117: Efficient Optimization for Discriminative Latent Class Models, Armand Joulin et al.
7	2006_154: Fast Iterative Kernel PCA, Nicol N. Schraudolph, Simon GÅÿnter, S.v.n. Vishwanathan 2010_278: Robust PCA via Outlier Pursuit, Huan Xu, Constantine Caramanis, Sujay Sanghavi 2013_46: Online Robust PCA via Stochastic Optimization, Jiashi Feng, Huan Xu, Shuicheng Yan 2014_171: Sparse PCA with Oracle Property, Quanquan Gu, Zhaoran Wang, Han Liu
8	1998_54: Bayesian PCA, Christopher M. Bishop 1998_128: Bayesian Modeling of Facial Similarity, Baback Moghaddam, Tony Jebara, Alex Pentland 2006_72: TrueSkillÂł: A Bayesian Skill Rating System Ralf Herbrich, Tom Minka, Thore Graepel 2013_213: Machine Teaching for Bayesian Learners in the Exponential Family, Xiaojin Zhu

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.

We expect to categorize the NIPS papers to 20 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. In order to check whether our classification works well, we also find the title of almost article. Although knowing title does not means knowing the main part of article, it is no doubt that title still can give us some important information .The title of each article is also in the attachment. And we hope the title for each article can be used for the later dataset users.

However, not all of the results are as good as expected. For example, there are some other articles about 'PCA' are not in the label 7. This is caused by firstly, the SVD lost some information of the raw data. SVD works well when there are a big gap between two eigenvalues from largest to smallest order. And then we can pick the eigenvalues before that gap. However, in this document-word matrix, we do not have such big gap for eigenvalues. In addition, the K-means clustering didn't perform so well when handling too much data.

3.2 AUTOENCODER

Autoencoder is an artifical neural network for unsupervised learning. We can use autoencoder to extract the feature from our data and reduce the dimension of feature. The input and target for autoencoder are same. The autoencoder can be separated to two parts: encoder and decoder. In the most case, the number of hidden units of one hidden layers is less than the number of unit in output/input layers. The illustration of autoencoder is in the Figure 3.1

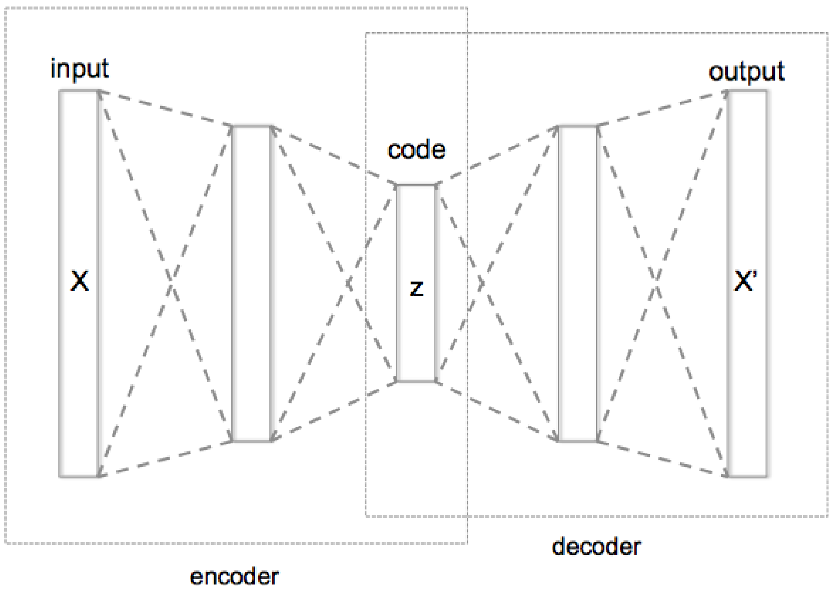

Figure 3.1: Autoencoder

We use X to stand the input, X' to stand output and Z to stand code. Assume the dimension of X, Z are N, M, the loss function we used in our autoencoder is:

$$L(X, X') = \frac{1}{N}\sum_{i=1}^{N}(X_i - X'_i)^2,$$

which is MSE. Unlike the supervised learning, our purpose is not to obtain output but to obtain Z which can be regard as the feature of X. More importantly, when using affine encoder and decoder without any nonlinearity and a squared error loss, the autoencoder essentially performs principal component analysis (PCA) So when we use non-linear activation function, we can regard autoencoder as non-linear extension of PCA or SVD.

Table 3.2: Part of results of K-means clustering 2

label	documents
0	1987_7: Centric Models of the Orientation Map in Primary Visual Cortex 2008_22: Goal-directed decision making in prefrontal cortex: a computational framework 2008_155: Modeling Short-term Noise Dependence of Spike Counts in Macaque Prefrontal Cortex
6	1987_2: Stochastic Learning Networks and their Electronic Implementation 1990_9: Stochastic Neurodynamics, J.D. Cowan 1994_34: Stochastic Dynamics of Three-State Neural Networks Toru Ohira, Jack D. Cowan 2013_87: Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n) 2013_125: Stochastic Convex Optimization with Multiple Objectives, Mehrdad Mahdavi et al.
8	1989_48: Bayesian Inference of Regular Grammar and Markov Source Models ,Kurt R. Smith et al. 1994_55: Bayesian Query Construction for Neural Network Models, Gerhard Paass et al. 1998_54: Bayesian PCA, Christopher M. Bishop 2014_182: Sparse Bayesian structure learning with ÃŠdependent relevance determinationÃŞ priors

Table 4.1: Contingency Table of Autoencoder&SVD

Class	1	2	3	4	5	Total
Autoencoder	1474	988	1599	1121	628	5810
SVD	1793	382	3241	393	1	5810
Ttoal	3267	1370	4840	1514	629	11620

In our model, we use sigmoid function as our activation function and use dropout to avoid overfitting. The code for autoencoder can be found in the attachment. After extract the feature, in order to compare the difference between SVD and autoencoder, we also use K-means clustering to classify the document. As for the limited space, we just present a part of the result in the table 3.2.

4 Difference between autoencoder and SVD

As classification results got from autoencoder and SVD show different distributions, we thus want to explore statistically whether autoencoder and SVD have diverse effects for the classification. Here we resort to the contingency table which treats two methods as two criteria variables and then use the Pearson's chi-squared test to see the dependence between them.

By classifying all of papers into 5 classes by these two methods, we can get the contingency table as shown by the table 4.1. Cells under each column of various classes show the number of papers belong to the class by two methods. Here we take 5 classes instead of 20 because we try to label same groups of papers together by two different methods and thus check variations.

The chi-squared test statistic is 1831.324 and the p-value is very close to 0 which one can reject the null hypothesis that there is no difference between the distributions of two methods. This justifies the method exerts large influence on the classification. Thus we can get the conclusion statistically that Autoencoder, a nonlinear dimension reduction, and SVD, a linear one, are total different ways for the classification of NIPS papers.

5 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 still consider reducing dimension firstly and then try to classify those papers by together 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. In the last work, we use LSA and SVC to do classifications. However, we find there are some mistakes in our code. So we correct the mistakes and then use another approach – autoencoder to try get better result. However, there is no such evaluation to determine whether our result is better or not. We find the title for each article and to judge the result of our classification. The results from SVD and autoencoder are reasonable. And we find the classification from two methods are independent through hypothesis test. In the future, we may try to combine two methods together to get better results.

6 Remark of contribution

Han Ruijian correct mistakes in the previous code and suggest to use autoencoder to check whether it can outperform SVD.

Tan Chunxi find the title for each article and make the classification for two methods.

Ye Rougang make the independence hypothesis test for results of SVD and autoencoder.

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.