Machine Learning Assignment – Kernelized PCA and Clustering on 20 Newsgroups

Chia-Man Hung
CDT AIMS, University of Oxford
chia-man.hung@eng.ox.ac.uk

November 18, 2017

Abstract

Kernelized Principal Components Analysis is a method for performing a nonlinear form of Principal Components Analysis. By using the kernel trick, one can compute the principal components efficiently without having to compute the feature map directly. K-means is a classical and simple algorithm for performing clustering. By combining these two methods, three categories of documents from 20 Newsgroups Data Set are classified into three classes. We experiment with different kernel functions and evaluate their performance by their classification accuracy.

1 Introduction

This is the assignment from an one-week long Machine Learning course[1]. We implement a simple unsupervised learning algorithm to classify documents from 20 Newsgroups Data Set[2] into several categories. A detailed description on the datasets can be found in the assignment sheet[3]. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. First, we transform documents into tf-idf (term frequency inverse - document frequency) vectors as their representation. Then, we perform a Kernelized Principal Components Analysis to reduce the dimensionality. Last, we cluster the vectors formed by the top principal components using the K-means algorithm.

In the following, we first explain how the algorithms work in theory – Kernelized Principal Components Analysis and K-means algorithm. Then, we show the experiments conducted and evaluate the kernel functions by computing the classification accuracy on test data set. We also explore the relationship between the number of principal components and the classification accuracy.

[1] http://www.cs.ox.ac.uk/people/varun.kanade/teaching/ML-AIMS-MT2017/

[2] http://qwone.com/~jason/20Newsgroups/

[3] http://www.cs.ox.ac.uk/people/varun.kanade/teaching/ML-AIMS-MT2017/practicals/practical2.pdf

2 Methods

2.1 Kernelized Principal Components Analysis (PCA)

2.1.1 Introduction

Principal Components Analysis (PCA) is a linear dimensionality reduction technique. To reduce the dimensionality to k, it finds the top k orthogonal directions of largest variance and projects every data point $(x_i)_{i=1}^N \in \mathbb{R}^D$ to these directions (note that the data points are centered: $\sum_{i=1}^N x_i = 0$). Denote $X \in \mathbb{R}^{N\times D}$ as the data matrix (the i-th row of X is x_i). Finding the top k orthogonal directions of largest variance is equivalent to finding the top k eigenvectors of the covariance matrix X^TX with the largest eigenvalues.

B. Schölkopf et al. [2] generalized the standard PCA to a nonlinear one by combining it with kernel functions. In the following, we describe the algorithm step by step, without giving any theoretical justification.

2.1.2 Algorithm

Input arguments

- N training data points of dimension D: $(x_i)_{i=1}^N \in \mathbb{R}^D$.
- Number of principal components $k < N$.
- A kernel function $\kappa : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Denote the corresponding feature map $\phi : \mathbb{R}^D \to \mathbb{R}^M$. $\phi(X) \in \mathbb{R}^{N\times M}$. Recall the kernel trick

$$\kappa(x_1, x_2) = \phi(x_1) \cdot \phi(x_2). \tag{1}$$

Training Data Fitting phase

- Compute the centered kernel matrix

$$\widetilde{K} = K - UK - KU + UKU, \tag{2}$$

where K is the kernel matrix

$$K = \phi(X)\phi(X)^T = \begin{bmatrix} \kappa(x_1,x_1) & \kappa(x_1,x_2) & \cdots & \kappa(x_1,x_N) \\ \kappa(x_2,x_1) & \kappa(x_2,x_2) & \cdots & \kappa(x_2,x_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(x_N,x_1) & \kappa(x_N,x_2) & \cdots & \kappa(x_N,x_N) \end{bmatrix}, \tag{3}$$

$$U = 1 \cdot 1^T/N. \tag{4}$$

- Compute the top k eigenvectors with the largest eigenvalues. Store these k normalized eigenvectors divided by the square-root of their eigenvalue in $\alpha \in \mathbb{R}^{N\times k}$. Each column is one of the computed vectors. (This corresponds to $\sigma^{-1}u$ in the slides[4].)

[4] http://www.cs.ox.ac.uk/people/varun.kanade/teaching/ML-AIMS-MT2017/slides/slides01.pdf

New Data Prediction Phase

- Compute the centered kernel matrix of the new data points $(x_{new,i})_{i=1}^{N_{new}} \in \mathbb{R}^D$. Note $X_{new} \in \mathbb{R}^{N_{new} \times D}$ the corresponding data matrix.

$$\begin{aligned}&[\widetilde{K}_{new}]_{i,j}\\ =&[\widetilde{\phi}(X_{new})\widetilde{\phi}(X)^T]_{i,j}\\ =&\kappa(x_{new,i}, x_j) - \frac{1}{N}\sum_l \kappa(x_{new,i}, x_l) - \frac{1}{N}\sum_l \kappa(x_j, x_l) + \frac{2}{N^2}\sum_l\sum_m \kappa(x_l, x_m).\end{aligned} \tag{5}$$

  $\widetilde{K}_{new} \in \mathbb{R}^{N_{new} \times N}$.

- $\widetilde{K}_{new}\alpha \in \mathbb{R}^{N_{new} \times k}$ represents the top k principal components of the new data points.

2.1.3 Kernels

Below is a list of kernel functions we experimented with in this assignment.

- **Linear kernel:** $\kappa(x_1, x_2) = x_1 \cdot x_2$
- **Polynomial kernel of degree d:** $\kappa(x_1, x_2) = (1 + x_1 \cdot x_2)^d$
- **Gaussian kernel with parameter σ:** $\kappa(x_1, x_2) = \exp(\frac{-(x_1 - x_2)^2}{2\sigma^2})$
- **Laplacian RBF kernel with parameter σ:** $\kappa(x_1, x_2) = \exp(-\frac{|x_1 - x_2|_1}{\sigma^2})$

Note that the kernelized PCA with a linear kernel is equivalent to the standard PCA.

2.2 K-Means Clustering

K-means clustering is a method of partitioning data points in a fixed number of clusters. The term k-means is first employed by Macqueen [1] in 1967, though the idea goes back to Steinhaus [3]. There are a lot of variants, but we only describe the most common standard algorithm.

Given a set of data points $(x_i)_{i=1}^N \in \mathbb{R}^D$ and a set of k means $(m_i)_{i=1}^k \in \mathbb{R}^D$, the algorithm proceeds by alternating between the following two steps.

- **Assignment step** Assignment each data point to the closest mean (i.e. the closest cluster).
- **Update step** Compute the new means to be the centroids of the data points in the new clusters.

3 Experiments and Results

3.1 Environment and Experiments Setup

Our implementation is done in python. The code can be found on my github[5]. It is organized as follows. *kernel_pca.py* (resp. *kmeans.py*) defines a class for

[5] `https://github.com/ascane/kernel-pca`

kernel PCA (resp. the k-means algorithm). *kernels.py* contains all the kernels we experimented with. *main.py* wraps up all the steps from downloading the data, reducing the dimensionality, to clustering and computing the classification accuracy.

The total number of documents in the three categories *sci.med*, *misc.forsale*, *soc.religion.christian* is around 2600. It takes around half an hour to perform kernel PCA and k-means clustering. (The most time-consuming parts are the computation of the kernel matrix and the eigenvectors.) For this reason, the documents are shuffled randomly and we reduce the number of documents to 1000. This reduces the computation time to 5 - 10 minutes depending on the kernel. We further split the documents to 800 for training and 200 for testing.

After the clustering, a permutation may be required to find the prediction results. To solve the problem of permutation, we list all six possibilities ($3! = 6$). We apply all six permutations on the clustering results of the training data and determine the permutation by computing the classification accuracy with regards to the ground truth. We then apply the same permutation on the test data to determine the prediction results of the test data and compute its classification accuracy.

The result of k-means depends on the random initialization. Therefore, for every setting, we perform k-means multiple times (10 times in our experiments) and pick the best test accuracy. We could have also picked the clustering of the lowest k-means cost.

3.2 Results

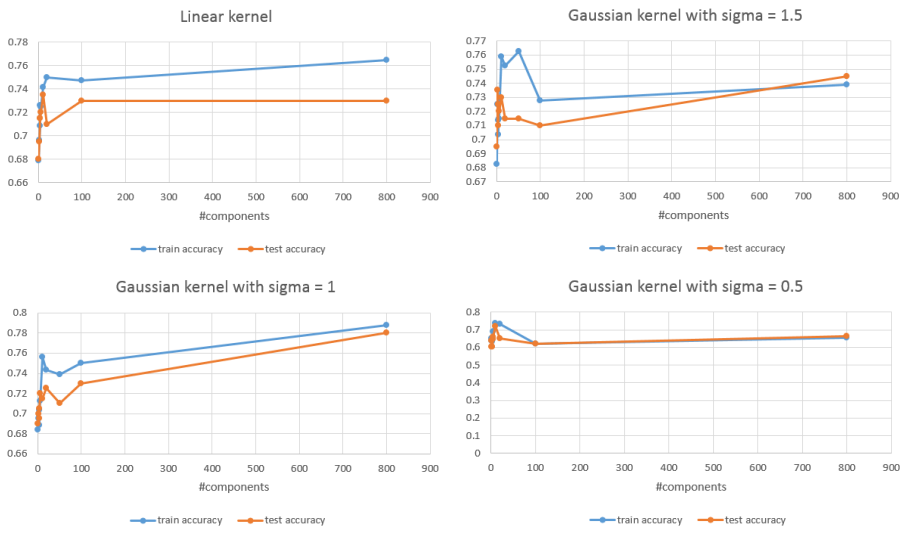

Figure 1: Classification accuracy on training and test data with different kernels and different number of components.

In our use case, the standard PCA (PCA with linear kernel) already gives reasonably good performance. This is mostly due to the nature of our data. PCA with Gaussian kernel with $\sigma = 1$ and with $\sigma = 1.5$ both yield slightly

Linear kernel		
#components	train accuracy	test accuracy
1	0.67875	0.68000
2	0.69625	0.69500
3	0.70875	0.71500
4	0.72625	0.71500
5	0.72500	0.72000
10	0.74125	0.73500
20	0.75000	0.71000
100	0.74750	0.73000
all	0.76500	0.73000
Gaussian kernel with $\sigma = 1.5$		
#components	train accuracy	test accuracy
1	0.68250	0.69500
2	0.72500	0.73500
3	0.70375	0.71000
4	0.71375	0.72500
5	0.71500	0.72000
10	0.75875	0.73000
20	0.75250	0.71500
50	0.76250	0.71500
100	0.71750	0.71000
all	0.73875	0.74500
Gaussian kernel with $\sigma = 1$		
#components	train accuracy	test accuracy
1	0.68375	0.69000
2	0.69500	0.70000
3	0.68875	0.69500
4	0.70375	0.70500
5	0.71250	0.72000
10	0.75625	0.71500
20	0.74375	0.72500
50	0.73875	0.71000
100	0.75000	0.73000
all	0.78750	0.78000
Gaussian kernel with $\sigma = 0.75$		
#components	train accuracy	test accuracy
all	0.68875	0.68500
Gaussian kernel with $\sigma = 0.5$		
#components	train accuracy	test accuracy
1	0.64000	0.64000
2	0.65125	0.60500
3	0.63375	0.60500
4	0.64500	0.64500
5	0.68875	0.66000
10	0.73500	0.72000
20	0.73250	0.65000
100	0.62125	0.62000
all	0.65500	0.66500
Laplacian RBF kernel with $\sigma = 1$		
#components	train accuracy	test accuracy
all	0.33875	0.35500

Table 1: Classification accuracy on training and test data with different number of components. 800 documents for training and 200 for testing.

Polynomial kernel		
#degree	train accuracy	test accuracy
2	0.67250	0.61000
3	0.69675	0.69000
4	0.55625	0.51000
5	0.55250	0.48000

Table 2: Classification accuracy on training and test data with polynomial kernel of different degree. 800 documents for training and 200 for testing. #components = all.

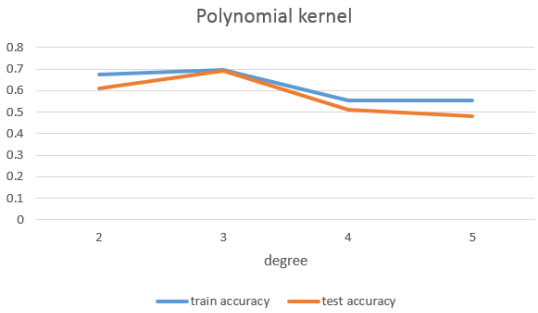

Figure 2: Classification accuracy on training and test data with polynomial kernel of different degree.

better results. PCA with Laplacian RBF kernel does not work very well. To our disappointment, PCA with polynomial kernel does not even work better than the standard PCA.

In general, the accuracy increases as the number of components increases. It is not always the case due to the random initialization of k-means. The accuracy does not increase a lot when the number of components is already large, i.e. only using the top few components (about 10 components) is enough to represent the data.

4 Conclusion

In this assignment, we described the kernelized PCA and the k-means algorithm. We applied them to a multi-class documents classification problem. Documents are represented by tf-idf vectors. We experimented with different kernels – linear kernel, polynomial kernel, Gaussian kernel, Laplacian RBF kernel. Among the experiments conducted, PCA with Gaussian kernel with $\sigma = 1$ and with $\sigma = 1.5$ work the best. The best classification accuracy achieved on test data is 78.0%, using Gaussian kernel with all components and $\sigma = 1$.

References

[1] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical*

statistics and probability, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[2] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[3] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.