Principal Component Analysis for Semantic Classification AMATH 582 Final Project

Benjamin Liu, Kelsey Maass, and Riley Molloy

March 17, 2016

Abstract

Principal component analysis (PCA) and classification by supervised learning are two popular topics in data science today. In this project, we combine techniques from both areas in order to classify news articles based on their word frequency content. We find that we can accurately classify the data by projecting onto a small subset of principal components, reducing the feature space from nearly 10 000 elements to just four. We also compare results from the traditional and robust PCA formulations, and discuss what additional semantic information can be inferred from our results.

1 Introduction and Overview

Dimensionality reduction is a property which all data scientists look for in their work. It allows for ease and speed in computation and promotes less complexity in many dense problem spaces. Principle component analysis is a standby in dimensionality reduction techniques, grounded in the power and breadth of the singular value decomposition.

Our work employs the use of PCA to lower the dimensionality of a large set of word-frequency data in five different classes of articles published by the British Broadcasting Corporation (BBC): Business, Sports, Entertainment Politics, and Technology. Each article, considered a single observation, is treated as a vector in the space of words used in every article. Each component of a vector represents the frequency of a given word throughout the article. We look to reduce this large space of words and represent each article with a small number of components, then classify each article as one of the five article types—i.e., we look to learn the semantics of each article in its low-dimensional feature space. This type of analysis is often referred to as "Latent Semantic Indexing."

In order to test the validity of our dimensionality reduction, we employ four different supervised learning techniques for classification: Nearest mean, K-nearest neighbors, Logistic regression, and a Neural network. We examine the accuracy of the classification for each method and how it varies with the number of principle components used to represent the feature space.

In addition to PCA, we also employ a variant known as Robust PCA, and classify using its results as well. Robust PCA first separates the original data matrix into two matrices: a sparse matrix and a lower rank matrix. In our application, the sparse matrix should indicate words which are not found in many articles, but are prominent in a few. Thus, the low rank matrix contains fewer outliers and consists of data on which the traditional PCA procedure should yield better results.

2 Theoretical Background

2.1 Principal Component Analysis

Principal component analysis is a powerful tool for isolating low-dimensional structure embedded in high-dimensional data. The goal of PCA is to find the best linear transformation of the data which maximizes the variance in the first principal component direction and in each subsequent orthogonal directions. To perform PCA, we recall the Singular Value Decomposition, or SVD,

$$A = U\Sigma V^*, \tag{1}$$

which exists for every matrix $A \in \mathbb{R}^{m \times n}$. We refer to the number of nonzero singular values of a matrix as the *rank* of a given matrix. We refrain from listing otherwise significant information about the SVD in hopes that the reader holds some level of awareness of the relevant theorems and properties.

We arrange our matrix of data $A \in \mathbb{R}^{9635 \times 2225}$ such that each row represents a different word (or *feature*) and each column represents a different article. We arrange the columns by the classes of articles. In addition, we require this matrix to have rows of mean zero. Since this will not be true of the data initially, we redefine X by subtracting off the mean of each row of X, and letting this be the "new" data matrix. We then take the SVD of this matrix as in (1).

The matrix U now consists of columns which represent the ordered principle components, Σ represents the importance of each singular value with respect to the others, and V represents how each article is projected onto the principle component space. To represent each article i in terms of the determined principle components, we can view each article as $\sigma_i v_i$, where σ_i the corresponding singular value from Σ, and v_i is the i-th *row* of V. Since we look to reduce the dimensionality of the data, we intend to only examine a small number of principle components; therefore, we look at only the first several components of v_i.

2.2 Robust PCA

Robust Principle Component Analysis is a variant of PCA which presupposes the data has some low dimensionality suitable for PCA, but with an unknown amount of sparse noise within the data. That is, for the data matrix A, we assume that

$$A \approx L + S,$$

where L has low rank and S is sparse. To penalize these assumptions convexly, we consider the formulation

$$\min_{L,S} \|A - L - S\|_F^2 + \lambda_1 \|L\|_* + \lambda_2 \|S\|_1, \tag{2}$$

where $\|\cdot\|_*$ is the nuclear norm, defined by

$$\|\cdot\|_* = \|\sigma(\cdot)\|_1$$

where σ maps a matrix to a vector of its singular values. Therefore the nuclear norm of a matrix is the sum of its singular values. In addition, $\|\cdot\|_1$ is the vectorized ℓ_1 norm, treating $S \in \mathbb{R}^{m \times n}$ as $S \in \mathbb{R}^{mn}$. The quantity $A - L - S$ is considered an error term, added in for computational stability. The parameters λ_1 and λ_2 can be tuned to prioritize low rank in L or sparsity in S. We can now choose one of many available methods for solving convex optimization problems. Our approach is discussed in the Algorithms section of this report.

2.3 Classification Methods

KELSEY

2

3 Algorithm Implementation and Development

3.1 Article Data

Our dataset consists of 2225 BBC articles labeled by topic, including business (510), entertainment (386), politics (417), sports (511), and technology (401). The data, collected for a paper presented at the 2006 International Conference on Machine Learning [1], consists of integer word counts for 9635 distinct words subject to the following pre-processing steps:

1. Stemming to identify like-words (e.g. fish, fisher, fishing, fishes)

2. Removal of stop words (common words such as: a, about, am, any, are, etc.)

3. Low-count (less than 3) removal

3.2 Robust PCA

As mentioned previously, many methods of optimizing (2) exist. We chosen to implement a method described in [4] known as *Alternating direction method of multipliers* (ADMM), or *Douglas Rachford splitting*. The algorithm consists of finding closed form representations of the proximal operators of the nuclear norm and the vectorized 1-norm, and evaluating them iteratively and independently, comparing their results.

3.3 Classification Schemes

Kelsey addition

To classify articles in our cross validation set, we first project both datasets onto a subset of the training datas principal components. Next we compare our classification accuracy as we increase the number of principal components used for the following classification methods:

3.3.1 Nearest Mean

For the nearest mean method, we first project the training data onto the first four principal components and calculate the average projection for each class. To classify new articles, we assign the label of the closest mean projection.

3.3.2 K-Nearest Neighbors (KNN)

The K-nearest neighbors algorithm classifies new articles by finding their K nearest neighbors and assigning the label shared by the most neighbors. For our article classification, we let $K = 5$.

3.3.3 Logistic Regression

Logistic regression classifies articles by first drawing linear decision boundaries between the classes (one-vs-all) that maximize the log-likelihood of the probabilities that the training articles belong to their respective classes. The final label is chosen from the class with the highest probability.

3.3.4 Neural Network

To classify with neural networks, we first build a network with three layers: input, hidden, output. We then train the network to minimize a cost function on the training data and assign classes to new articles by choosing the class with the highest output value.

3.3.5 Support Vector Machine

Support vector machines (SVMs) are, in their basis formulation, binary classifiers. This means that they can only classify data between two classes. Several approaches exist to extend the SVM framework to support classification between more than two classes, such as one-versus-all, where SVMs are built to classify each class against all others, and one-versus-one, where SVMs are built to classify each class against each other class.

Simply generating multiple SVMs does not immediately fix the problem. Figure 1 shows results from applying the one-versus-all support vector machine classification scheme on data from four classes. Though the four classes are quite well-separated in space, attempting to separate each class against the remaining three with a linear separator is not very successful; data points in the central region are negatively classified by all of the support vector machines, whereas data points in the top, bottom, left, and right regions are doubly-classified.
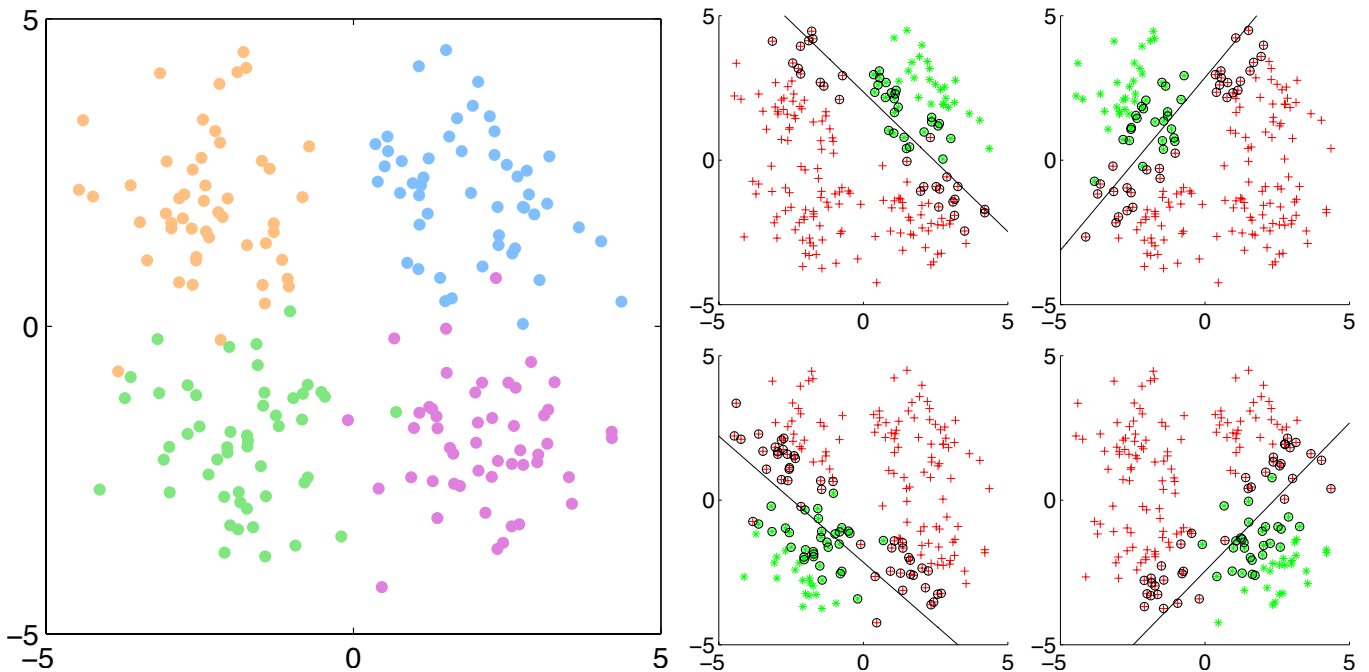

Figure 1: One-versus-all scheme for classification by support vector machine into four classes. **(Left)** Training data from four classes with some overlap. **(Right)** Support vector machines generated by considering data from each class versus all others.

To truly extend the one-versus-all SVM framework to classify multiple classes, we modify the SVM formalism so that rather than giving a binary classification, it provides a measure of goodness of fit. This can be thought of as returning the (signed) distance of the data point to the decision boundary, rather than simply which side of the decision boundary it falls on. We then compare the results of classifying the data point with each of the SVMs, and choose the SVM which classifies the data point as 'deepest' within the positive decision region. Figure 2 shows the result of applying this classification scheme on the data from Figure 1 with new test data points.
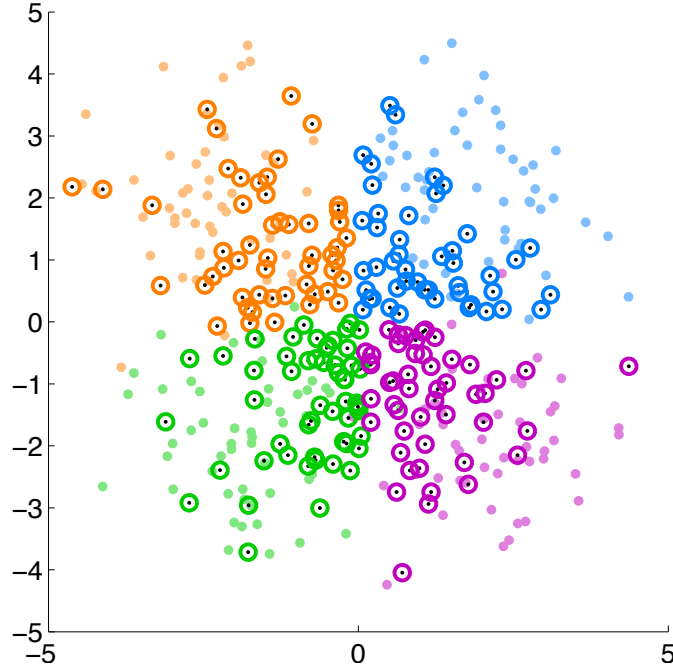
Figure 2: Results from applying the one-versus-all SVM classification scheme to test data. Classifications are shown as circles around each point.

In our work with the BBC article data, we found that the built-in optimization routines used in building the SVMs failed to converge for the large data set. We found that even in our two-dimensional test cases, that by increasing the number of data points in the training set we could induce this problem. It therefore seems that isolating the support vector training data points may have been the most significant factor in algorithm convergence, and we speculate that performing some kind of pre-processing to remove easily-classified (interior) points in each class may improve the performance of the algorithm.

4 Computational Results

4.1 PCA and Robust PCA

While data methods are powerful tools for discovering trends and structure of large in large datasets, inferring semantic meaning from these models can be difficult. For example, principal component analysis determines the best basis to represent our article data in terms of word frequency values and variance, with no regard to semantic meaning. Nevertheless, our article classification analysis uncovers some trends with meaningful interpretation in terms of the thematic content of the articles.

Below, we list the 25 words with the highest weights in each of the first four principal components. Since each principal component is a linear combination of words, it is difficult to interpret their meaning. However, we note that the fourth principal component has many technology-related words, which could explain why the inclusion of the fourth principal component is necessary to correctly classify technology articles.

- **PC 1:** game, play, win, player, firm, England, company, market, govern, year, against, team, growth, share, side, best, sale, price, club, bank, economy, go, cup, people, coach

- **PC 2:** film, award, year, best, music, party, labour, star, elect, govern, Blair, include, sale, ministry, Tori, top, show, actor, nomination, told, number, album, 2004, people, band

5

- **PC 3:** film, game, party, sale, labour, year, best, award, elect, market, firm, Blair, people, company, growth, price, share, Tori, profit, bank, music, ministry, England, star, player

- **PC 4:** people, game, phone, user, mobile, year, tech, music, service, software, Microsoft, computer, firm, best, site, program, govern, search, labour, economy, elect, digital, net, online, Blair

In Figure 3, we compare the energy percentages of each of the first ten singular values for both PCA and robust PCA, where we note that in the robust PCA results the first four principal components increase in significance.
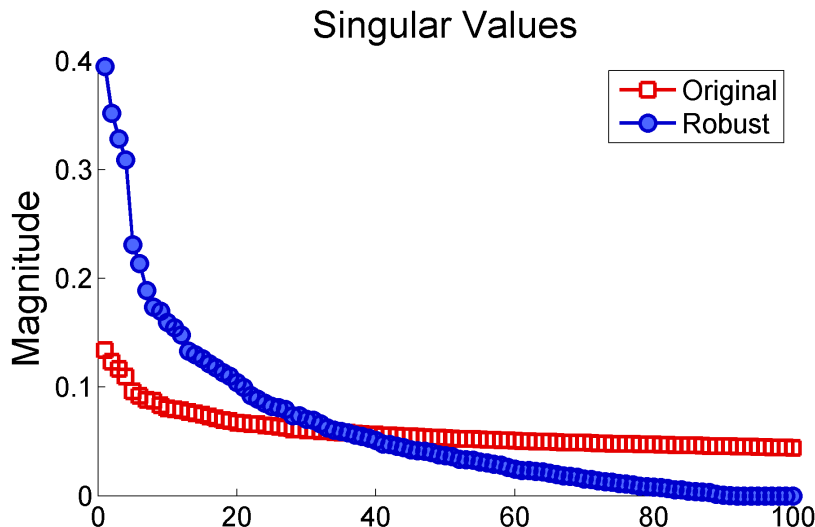

Figure 3: Principal component analysis of the article data. **Left.** Comparison of first 100 normalized singular values derived from the data with and without robust PCA. **Right.** Contribution of principal components to word frequencies for articles in each class.

4.1.1 Sparse Matrix Analysis

Robust PCA yields a sparse matrix containing elements which represent words important to a small number of articles. Our sparse matrix contained 252 non-zero elements, compared with 227,814, the total number of non-zero elements in the training dataset. We notice that these words correspond to subtopics within given category (e.g.: last names of athletes, music terminology, Yahoo). Some of the most frequent of these words are listed below.

Business	Sports	Entertainment	Politics	Tech
call	roddick	song	wage	game
centre	nadal	music	minimum	ink
	dallaglio	urban	forsythe	yahoo
		best	increase	gadget

4.2 Classification

In the four classification methods we tested, we found that the classification accuracy did not significantly increase when we added more than four principal components.
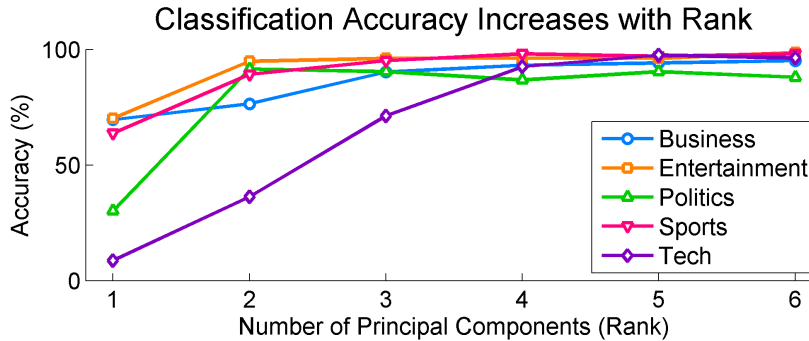
Figure 4: Comparison of classification accuracy for each article class as we vary the number of principal components (rank) used.

Rank	1	2	3	4	5
Nearest Mean	59%	77%	82%	96%	96%
KNN ($K = 5$)	56%	77%	86%	96%	97%
Logistic	45%	75%	78%	94%	94%
Neural Net	46%	75%	79%	95%	95%

Rank	1	2	3	4	5
Nearest Mean	48%	78%	89%	95%	96%
KNN ($K = 5$)	52%	77%	92%	96%	96%
Logistic	45%	74%	77%	93%	93%
Neural Net	45%	74%	79%	93%	93%

Table 1: Comparison of classification accuracy for four classification algorithms on our PCA space (top) and robust PCA spaces (bottom).

4.2.1 Misclassification

While projecting onto the principal components of the training set does a good job at separating the article by class, it does not separate them perfectly. Therefore articles near the decision boundaries may be misclassified.

For example, Figure 5 shows how articles from each class are classified by the K-nearest neighbor algorithm as we vary the number of principal components used. It is interesting to note that articles from business and politics are likely to be mistakenly cross-classified. Furthermore, technology articles are prone to misclassification if the fourth principal component, which contains many technology-related words, is ignored.
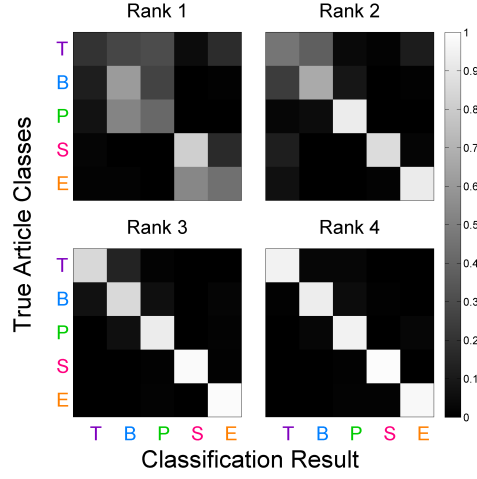
Figure 5: Classification of each article by type. Diagonal elements indicate percent of articles classified correctly; off-diagonal elements correspond to misclassified articles.

5 Summary and Conclusions

BEN

References

[1] D. Greene and P. Cunningham, *Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering*, Proc. ICML, 2006.

[2] Y. Ma, E. J. Candes, X. Li, and J. Wright. *Robust Principal Component Analysis?* Technical report, Stanford University , Stanford CA, 2009.

[3] J. N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems and Big Data.* Oxford University Press, 1 st edition, 20013.

[4] N. Parikh, S. Boyd, *Proximal Algorithms.* Foundations and Trends in Optimization, Vol. 1, No.3, 2013.

Appendix A: MATLAB Functions

ind = knnsearch(Xtrain,Xtest,k) Finds the **k** elements of **Xtrain** that are nearest to **Xtest**, and returns the indices of these elements in **ind**.

svmtrain

svmdecision

Appendix B: MATLAB Codes

All of our code can be accessed here.