

# Principal Component Analysis for Semantic Classification

## AMATH 582 Final Project

Benjamin Liu, Kelsey Maass, and Riley Molloy

March 14, 2016

### Abstract

Principal component analysis (PCA) and classification by supervised learning are two popular topics in data science today. In this article, 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

$$A = USV^* \tag{1}$$

### 2.2 Robust PCA

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

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

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

where  $\sigma$  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  $\ell_1$  norm, treating  $S \in \mathbb{R}^{m \times n}$  as a vector in  $S \in \mathbb{R}^{mn}$ .

### 2.3 Classification Methods

## 3 Algorithm Implementation and Development

### 3.1 Article Data

ALGORITHM FOR TURNING ARTICLES INTO WORDCOUNT / WORDFREQUENCY DATA. STEMMING, LOW WORDCOUNT REMOVAL, STOPWORDS.

## 3.2 Robust PCA

## 3.3 Classification Schemes

### 3.3.1 Nearest Mean

### 3.3.2 K-Nearest Neighbors (KNN)

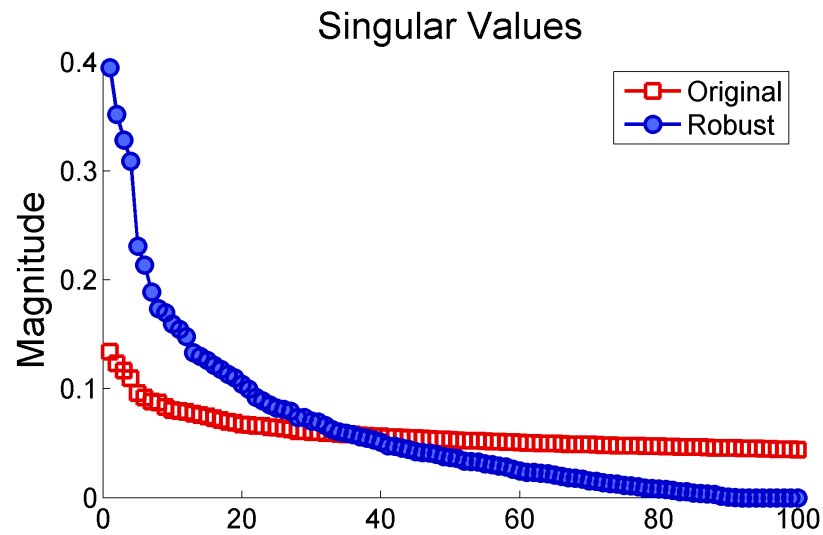
### 3.3.3 Logistic Regression

### 3.3.4 Neural Network

### 3.3.5 Support Vector Machine

## 4 Computational Results

### 4.1 PCA and Robust PCA



### 4.2 Classification

## 5 Summary and Conclusions

## 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