Spectral Clustering for Social Media Analysis

Sanjay Krishnan

December 10, 2012

Abstract

Users in Social Media systems generate a vast amount of heterogeneous data containing numerical, textual, and implicit relationships. In this project, I will develop a single unsupervised learning framework to uncover patterns and relationships between different types of user data. The Laplacian Eigenmap algorithm is a spectral embedding algorithm that is flexible and promises to project data points on a lower dimension space in a way that preserves local structure of the manifold on which data points lie. I will focus my work on the responses in the Opinion Space dataset. Developed in the Berkeley Automation Laboratory, Opinion Space is a platform that uses the wisdom of the crowd to power brainstorming. In 2010, a version of Opinion Space was launched for the US Department of State to collect ideas about foreign policy. The project has a dataset of about 5700 users and associated textual ideas, responses to survey questions, and evaluations of others’ ideas.

1 Introduction

1.1 Opinion Space

Opinion Space [8] is an interactive social media tool that allows users to visualize the diversity of opinion surrounding a topic of interest. This section will introduce the system, and describe its key components.

1.1.1 Five baseline statements

When participants join the system, they express their opinions on five statements. These statements reflect basic issues, current events, and political views. For the dataset used in this project the following statements were used, and opinions were listed on a continuous scale from Strongly Agree to Strongly Disagree:

1. The most urgent security threat to the United States is a terrorist armed with a nuclear weapon.
2. Continuous diplomatic efforts are required to produce lasting, sustainable peace in Afghanistan and Pakistan.
3. Climate change poses a threat to political stability around the world.
4. Investing to increase food production in other countries will ultimately benefit me and my family in the future.
5. The best way to advance a country’s economic development is to empower its women.

Figure 1: An example of a statement and slider
1.1.2 Textual response

After users enter the system, they are prompted to answer a general discussion question with 1000 characters or less. In this dataset, the question is “If you met U. S. Secretary of State Hillary Clinton, what issue would you tell her about, why is it important to you, and what specific suggestions do you have for addressing it?”. Furthermore, users in the system evaluate each other’s responses and the dataset contains over 20,000 evaluations on a continuous [0,1] scale.

1.1.3 Prior work on this dataset

In the past, researchers have looked at two separate problems with this dataset: clustering users by baseline statement responses, and sentiment analysis (inferring statement responses from textual ones). These attempts to address the problem have relied on linear dimensionality reduction techniques such as Principal Component Analysis, and dataset alignment techniques such as Canonical Correlation Analysis[8, 1]. These techniques revealed some structure, however they were unable to produce interpretable clusters, and a rigorous inference framework. An underlying problem was these attempts lacked an appropriate statistical model of the data and this project will address this problem.

1.2 The Problem

Opinion Space users provide two principal forms of data: baseline statement responses and textual comments. I want to understand the statistical links between these two classes of data. In the abstract, this is a very unstructured problem, since I am comparing quantitative values to qualitative responses, and the space of all possible qualitative responses is immense.

All hope is not lost, intuitively political views have a lot of latent structure. For example, one can guess a person’s views on a given issue based on their party affiliation. This leads to a hypothesis that users can be parametrized by a small number of features and everything else they do in the system is simply a function of these features. Formally, the problem that this project addresses is given a group of users with similar baseline statement ratings, what features of the textual responses are also similar within the group.

What I hope to find is statistical justification for general intuition about politics and foreign policy. Our techniques should discover links between certain political views and the use of specific words. If a set of unsupervised techniques without prior knowledge about politics or the users can extract these divisions, it will be a meaningful contribution to the Opinion Space system. We can gain insight into how better to prevent biases, direct questions, and visualize the data.

1.2.1 Roadmap

To address this problem, I will first address 3 subproblems (Section 2, Section 3, and Section 4 respectively):

- Statistical featurization: A statistical model to convert the baseline statements and text into quantitative values
- User clustering: Find groups of users with similar responses to statement ratings.
- Word clustering: Rather than treating words independently, cluster them to find groups of co-occurring words (called topics in the rest of the project).

At the end, I will build upon these three components to develop a single model for associating a cluster of users with a topic in the textual comments. In a nutshell, given a clustering of users based on their responses to the baseline statements and another clustering of words mentioned by users; can we align the two datasets in a way that gives us insight into what clusters of users are saying in their textual comments.

1.2.2 Hardware resources

While the Opinion Space dataset is small, it is not a trivial dataset to analyze. For example, simple correlation matrix between all the users in the system would have 34 million entries. Social media poses a canonical Big Data problem and the techniques illustrated in this project will address scalability. To cope with the data, we will use
the BidMat library on an Intel Xeon CPU E5645 server. The algorithms presented in this project are implemented in a way that they can be run in a distributed environment such as Hadoop.

1.3 Relationship to similar work

I am proposing a two-step manifold alignment problem [10], which is a framework that is well studied in other fields. Formally, we assume that there are two datasets $X$ and $Y$, which both lie on (or near) lower-dimensional manifolds. The manifold alignment problem is to find shared latent features of the two datasets. The simplest of these algorithms is Canonical Correlation Analysis which is a linear alignment. In a two-step method, we run a dimensionality reduction algorithm on each of the datasets first, and then align the resulting manifolds.

Many existing algorithms assume that $X$ and $Y$ are relatively clean datasets, and the difficult part is discovering the relationship between them. Based on intuition, this will not be true for the Opinion Space dataset. A public social media system will be filled with spam, missing information, and complex biasing processes of users. Furthermore, since we are working with data that is intuitively related: political views and textual comments. As a result, I will focus my effort on the first step with complex data modeling and dimensionality reduction procedure.

In addition, the idea of working with clusters instead of individual points is another well studied Data Science technique. The use of Core-sets [2] have been proposed as a way to reduce the computational complexity in Optimization and Machine Learning. As you will see later, our use of clusters allows us to solve a non-convex optimization problem for dataset alignment by exhaustive search.

Finally, a lot of work has been done in the field of topic modeling [13, 4, 9]. Some techniques even use dimensionality reduction to extract topics. However, I will take a different approach choosing a dimensionality reduction/clustering framework that can apply to both the baseline statements and the text. By choosing a single framework, the techniques shown in this project can generalize past the Opinion Space platform to other heterogeneous data systems.

2 Introduction to the Laplacian Eigenmap Algorithm

Spectral clustering uses the eigenvectors (the spectrum) of a similarity matrix of data to compute a lower dimensional embedding of the data. This similarity matrix can be constructed in a variety of ways to address the goals of the dimensionality reduction problem. In this section, I will provide motivation and description for the Laplacian Eigenmap[3] algorithm, one such example of spectral clustering.

2.1 Local Structure vs. Global Structure

When considering the similarity between data points, we must differentiate between global and local structure. We define local structure as a data point’s similarity to other data points in its immediate vicinity, and global structure as an aggregate measure of all data points.

A canonical technique that preserves global structure is Principal Component Analysis (PCA). This technique is one of the most intuitive dimensionality reduction methods, and is often the first technique tried by data scientists. PCA reduces the dimensionality of a dataset by finding two orthonormal axes in feature space which maximize variances in their respective directions. Variance, defined as the expected value of the square difference from the mean, so we can interpret PCA as optimally preserving a data point’s distance away from the center of mass. The Laplacian Eigenmap algorithm that I propose, is a technique that can flexibly preserve global or local structure based on a few parameters.
2.2 Similarity Matrix, Connectivity Graph, and Spectrum

With ideas about Local and Global structure in mind, we can figure out how to construct a similarity matrix $S$ (also called a kernel matrix) for our problem. To formalize the problem, given a data point $x$ the locality of a data point is defined as:

$$\{ y : \|x - y\|_m < \epsilon \forall y \in D \}$$

Let us use this idea of locality to create a connectivity graph that connects point to point traversing the manifold. Each data point is a node in this graph, and let us create an edge between points in the $\epsilon$-locality.

Given this graph construction, we want a way to convert this graph into similarity matrix. Define the Gaussian Kernel Function:

$$\kappa(x, y; \sigma) = e^{-(\frac{1}{2\sigma^2}\|x - y\|_m)}$$

Define an adjacency matrix $A$. Suppose for every edge $(i,j)$, we can set $A_{ij} = \kappa(x_i, y_j; \sigma)$, and set every other element to zero. This matrix encodes the graph consistently, where we can efficiently encode local relationships. Furthermore, our two parameters: $\sigma, \epsilon$ can control how look at this neighborhood. $\epsilon$ controls how big this neighborhood is, and $\sigma$ controls how much the kernel function drops off with distance. Define $D$ as a diagonal matrix where each $D_{ii} = \sum_{j=1}^{N} A_{ij}$. Define $L = D - A$, as the Laplacian matrix. We will use this matrix as the similarity matrix, and our dimensionality reduction will be based on the spectrum of this matrix. This matrix is well defined in spectral graph theory, and is widely used in analysis. Another to interpret this matrix is as a matrix of balanced network flows, and looking at the spectrum is a way to optimally represent these flows.

From spectral graph theory[7] we know that the Laplacian matrix of a simple connected graph is rank $N-1$. To find the lower dimensional embedding we have to look at the generalized eigenvectors corresponding to the smallest eigenvalues excluded the ones in the null space of the matrix. Since a Laplacian is necessarily symmetric, these generalized eigenvectors will be orthogonal, and thus represent well defined coordinates.

2.3 Algorithm

Using the concepts developed above we can formulate the basic Laplacian Eigenmap algorithm:

1. Form the adjacency matrix $A$ of the connectivity graph, where $A_{ij}$ is equal to $\kappa(x_i, x_j; \sigma) = e^{-(\frac{1}{2\sigma^2}\|x_i - x_j\|_m)}$ if $\|x_i - x_j\|_m < \epsilon$.
2. Form the Laplacian matrix $L$ from $D$, where each diagonal element is from the row sum of $i^{th}$ row of $A$, and $L = D - A$.
3. Solve the generalized eigenvector problem: $Lf = \lambda Df$, and get the $k$ smallest $f_i$ such that the corresponding $\lambda_i > 0$.

This algorithm relies on an explicit formulation of the similarity matrix, and in our case with 5711 users, that is a matrix with over 32,000,000 entries. I will reformulate the algorithm to take advantage of computer systems principles such as caching and rely on fast linear algebra libraries to do my computation.

There are many iterative techniques to solve eigenvalue problems. Typically these techniques are good at finding the dominant eigenvectors in well conditioned problems. Below we will formulate the Laplacian Eigenmap algorithm as an algorithm to solve for dominant eigenvectors. Furthermore, we will reformulate it in a way that we don’t have to store the $L, D, A$, matrices in memory, just one matrix that to which we repeatedly apply power iteration. Our original generalized eigenvector problem:

$$Lf = \lambda Df$$

Assuming a fully connected graph, $D$ is invertible so we can reformulate the problem as follows:

$$(D^{-1}L)f = \lambda f$$

This is a simple eigenvalue problem, however to quickly parallelize this problem using power iteration we want to calculate dominant eigenvectors not the smallest ones. Here is a useful property of eigenvectors:

$$Mv = \lambda v \Rightarrow (I - M)v = (r - \lambda)v$$
r is a regularization constant to ensure that the eigenvalues do not go negative in our calculation. Our power iteration will be as follows:

\[(I\tilde{r} - D^{-1}L)f^{(k)} = f^{(k+1)}\]

We can simplify this expression:

\[(I\tilde{r} - D^{-1}L) = (I\tilde{r} - I + D^{-1}A) = D^{-1}A\]

\[(Ir + D^{-1}A)f^{(k)} = f^{(k+1)}\]

\[D^{-1}A\] can be seen as a representation of the fraction of flows, and by finding the eigenvalues we are finding the best non-trivial representation of the graph that preserves the fraction of flows. This matrix can be treated as a black box, and we can cache this value in memory. We can choose our \(\epsilon\) parameter so that \(D\) is a full rank diagonal matrix, and thus easy to invert.

### 2.4 Embedding to clustering

So far, I have only derived a lower dimensional embedding of the data points, however I still need to describe a technique to assign cluster labels to points in the low dimensional space. My technique will be to project the points on a two dimensional space, and then use the k-means algorithmNg et al. [11] to assign clusters to each point. The first question to answer is why k-means will not work directly on the higher dimensional data.

Parts of this project deal with high-dimensional and sparse datasets. The data is inherently complex and without prior intuition it very difficult to select an appropriate k-means parameters to label the data. To address this problem, I make a basic assumption that users are similar to each other and the data they provide lies on a lower-dimensional manifold. Motivated by this, I can dimensionality-reduction as a pre-processing step to gain insight into the structure of the distribution of points. Less complex data will be easier to process, parameter selection will be easier, and we can use 2-D/3-D embeddings to visualize the results. Furthermore, lower dimensional embeddings can be stored in K-D trees, allowing these techniques to scale up to larger data.

In addition, if the dimensionality reduction can increase the efficacy of the clustering technique. If our dimensionality reduction technique can preserve the local structure (distances between points close to each other), then our clustering can work for data that lies on complex manifolds, where simple euclidian distance may lead to unexpected results. Within the same family of algorithms, it has been shownNg et al. [11] that this combination of techniques (embedding and then clustering) can be very effective at clustering complicated manifolds. I intuitively expect the featurized textual data to lie on (or near) such a manifold since words will have co-occurrence (or conversely negative correlation) relations with each other.

### 2.5 Parameter selection

A careful reader will note that there are many different parameters to choose: \(\sigma\), the Gaussian Kernel parameter; \(\epsilon\), the neighborhood around a point; \(k\), the k-means parameter for clustering. While there are proposed techniques for automatic parameter selectionNg et al. [11], it was not implemented in this project. Instead, I chose parameters that were mathematically sensible. The Gaussian Kernel was set to be the standard deviation of the feature, \(\epsilon\) was set to select about 30 neighbors, and \(k\) was set after visualizing the data.

### 3 Statistical models and feature selection

Before I can cluster the data, I must first convert the data in the system into a quantitative values. Continuous sliders and textual responses pose an interesting problem for feature selection. In general, we want to choose a set of features to compactly represent the data, induce tight clustering, and is consistent. This section will highlight all of the modeling assumptions made in this project, and how I will use those assumptions to select features.
3.1 A model for continuous slider ratings

Prior work on Opinion Space treated the slider ratings as a continuous random variable in \([0,1]\). I will, however, explore a different model treating a continuous slider rating as a noisy observation of a true rating. Let \(Y\) be the observed continuous value, and \(X\) be a discrete random variable taking values \(\in [0, \alpha, \beta, 1]\). This model is motivated by a few different user interface considerations. First, our argument is that a user who gives a statement a rating .85 is much more similar to a user who gave a rating .65 than to a user who gave the max rating of 1. A rating of .85 is a very conscious effort not to give a full rating and must be considered accordingly. Next, the sliders are dynamically sized by the user interface engine on a user’s computer, this means that on different machines users rate with a different effective resolution. Finally, exact slider ratings are difficult to reproduce thus giving rise to a noisy model. In this model, the probability of \(Y\) given \(X\) is:

\[
p(Y = y \mid X = x) = \begin{cases} 
  x = 1 & y \sim U[1 - \epsilon, 1] \\
  x = \beta & y \sim N(\beta, \sigma_B) \\
  x = \alpha & y \sim N(\alpha, \sigma_\alpha) \\
  x = 0 & y \sim U[0, \epsilon]
\end{cases}
\]

This model shows that there are 4 intended ratings, however due to the considerations mentioned above the ratings are noisy. However, there is one caveat, it is very easy for a user to set a maximum or minimum rating of 0 or 1, and thus we don’t consider noise in for these ratings other than an \(\epsilon\) to account for display resolution differences. In Figure 3, we show how well this model matches our observations.

3.2 Learning the parameters

We can apply the Expectation Maximization algorithm (Algorithm 1) to learn the parameters of this model for each statement. While we could apply this algorithm jointly to the entire set of statements, we a priori do not know how many clusters to expect. Furthermore, there are a maximum of \(4^5 = 1024\) clusters, too much to run a penalized model selection algorithm for each. Later in this project, we will find that the complexity of associating clusters with topics is dependent on the number of clusters, so we want to be careful not to over-estimate this parameter.

The result of this optimization is a most likely intended rating for a user. We will use these ratings as the statement features for all future learning algorithms.

3.3 Statistical text model and featurization

The textual responses in this dataset pose a few challenging problems: responses are short, a corpus of only 2000 responses, and spam comments. In this section, I will set up the problem of featurization and different approaches to converting a textual comment into a quantitative value. I will ignore the structure of the language by using a bag of words approach. In this approach, the order of the words in a response is unimportant, and each word is considered conditionally independent given the previous word. Furthermore, to cope with the noise in the dataset, we will only count words that appear in more than ten documents. These are ad-hoc assumptions, but they serve to limit the complexity of the problem.
Algorithm 1 Expectation Maximization for Statement Ratings

Initialize: $\pi_1, \pi_a, \pi_{\beta}, \pi_0, \alpha, \beta, \sigma_a, \sigma_{\beta}$

E-step: $\tau_n^{(t)} = I(1 - \epsilon < x \leq 1)$

$\pi_1 = I(0 \leq x < \epsilon)$

$I(0 \leq x < \epsilon) = 1 \parallel I(1 - \epsilon < x \leq 1) = 1$

otherwise

$\pi_a N(x_n | \alpha, \sigma_a)$

$\pi_{\beta} N(x_n | \beta, \sigma_{\beta})$

M-step: $\pi_1^{(t+1)} = \frac{\sum t_n^{(t)}}{N}$

$\alpha^{(t+1)} = \frac{\sum x_n^{(t)} x_n}{\sum t_n^{(t)}}$

$\sigma_{\alpha} = \frac{\sum (x_n - \alpha^{(t+1)})^2}{\sum t_n^{(t)}}$

$\beta^{(t+1)} = \frac{\sum x_n^{(t)} x_n}{\sum t_n^{(t)}}$

$\sigma_{\beta} = \frac{\sum (x_n - \beta^{(t+1)})^2}{\sum t_n^{(t)}}$

4 Clustering users based on statement ratings

Using our proposed model of a statements rating, a noisy discrete rating, we can apply a clustering algorithm to group like users together. As mentioned before, the EM algorithm used to featurize the statement ratings will produce 4^5 clusters and we want a technique to intelligently group these clusters together and that scales well with the number of statements. I will use the Laplacian Eigenmap algorithm from Section 2, to achieve this goal.

4.1 Results

Projecting the data onto a two-dimensional plane, I found that there are 22 main clusters, furthermore the distance between clusters relate to how different the clusters are from each other. In Figure 4, I illustrate the homogeneity and the consistency of the clusters. I plotted users who agreed with a statement with an “O” and those who disagreed with an “X”, we see that the clusters are quite consistent and in fact in the case of the climate change question linearly separable.

5 Clustering words in textual comments

We ultimately want to connect clusters in statement space to topics that are mentioned in textual comments. I will first address the sub-problem of extracting topics from the text. The document space is very high dimensional, there are 2000 documents and over 1000 distinct words. Furthermore, this space is very sparse and a simple k-means or EM based clustering will not perform well without well-tuned parameters. Like before we can apply dimensionality reduction to gain insight into how many clusters to expect, and use that to extract topics from the text. We can apply the same Laplacian Eigenmap clustering algorithm[12, 6] mentioned before to the words in the system. We can featureize each word by a vector of documents that in which it appears. Then, we can apply dimensionality reduction to discover clusters of correlated words.

5.1 Feature Vectors

We will define an inner product space over the words, so we can run our clustering algorithms on the words directly. Let $d$ be the number of documents and $N$ be the number of distinct words in the corpus of documents. Then the word vector, $x_i{\forall i : 1 \ldots N}$, is a $d$ dimensional vector such that the $j$ element of $x_i(j)$ is 1 if the document $j$ contains the word, 0 otherwise.

Given two words $x_i, x_j$, we define

$$< x_i, x_j > = \text{cov}(x_i, x_j)$$

In our case this is equal to the sample covariance:

$$< x_i, x_j > = \frac{1}{N} \sum_{p=1}^{N} (x_i - \bar{x}_i)(x_j - \bar{x}_j)$$
Algorithm 2 Removing Stop Words

\( M := \) correlation matrix of k-dimensional clustered statements and d-dimensional word vectors
for \( w \in \{k...d+k\} \)
\( r_i \leftarrow |M(i,w)| \forall \{i \in 1...k\} \)
\( p_j \leftarrow |M(j,w)| \forall \{j \in k...d+k\}\backslash w \)
if \( \max_i r_i < \epsilon \) \( \max_j p_j < \delta \)
ignore word \( w \)

Using this inner-product space we can define a similarity measure on \([-1,1]\), as:

\[
\cos(\angle x_i, x_j) = \frac{<x_i, x_j>}{(||x_i||)(||x_j||)}
\]

This measure is known in statistics as the Pearson Correlation Coefficient.

\[
\rho = \frac{\text{cov}(x_i, x_j)}{\sigma_i \sigma_j}
\]

5.2 Cleaning the dataset

In our corpus of text there are many “stop” words eg. a, an, the. To extract these words, we will make a statistical assumption that all meaningful words are correlated with the statement ratings. Words that are not correlated with the clustered statement ratings, ie. they occur everywhere in the text with the same frequency, are excluded. Furthermore, words that do not strongly co-occur with other words are excluded. To quantify these correlations we use Pearson Correlation Coefficient, as mentioned above. Algorithm 2 illustrates the cleaning algorithm used.

5.3 Kernel Matrix

We will run the Laplacian Eigenmap algorithm with a different kernel matrix based on the Pearson Correlation Coefficient (PCC) described above. We will not use the Gaussian kernel function proposed in Section 3, instead we
will normalize the PCC to [0,1] by:

\[
 r_{i,j} := \frac{<x_i,x_j>}{(||x_i||)(||x_j||)}
\]

\[
 \hat{r}_{i,j} = \frac{r_{i,j} + 1}{2}
\]

Therefore our kernel matrix \(\kappa_{ij} = \hat{r}_{i,j}\). Let us confirm that this kernel matrix is a sensible measure of similarity. Suppose word i and word j are perfectly correlated, ie. they only occur or don’t occur in pairs. Then since \(x_i = x_j\):

\[
 r_{i,j} := \frac{<x_i,x_i>}{(||x_i||)(||x_i||)}
\]

\[
 \hat{r}_{i,j} := 1
\]

This is the same as the result with the Gaussian kernel when \(x_i = x_j, e^{-\frac{||x_i-x_i||^2}{2\tau}} = 1\). Suppose, the vectors are perfectly negatively correlated:

\[
 r_{i,j} := \frac{<x_i,-x_i>}{(||x_i||)(||x_i||)}
\]

\[
 \hat{r}_{i,j} := 0
\]

This is the same as: \(\lim_{x_j \to \infty} e^{-\frac{||x_i-x_i||^2}{2\tau}} = 0\)

5.4 Results

After data cleaning, there were 630 words to cluster. I ran the Laplacian Eigenmap algorithm to project down to 2 dimensions and then a k-means clustering to pick out clusters. To give intuition on what this algorithm does, Figure 5 and Table 1, shows a sampling, the word are sampled for the sake of visualization and presentation. In reality topics contain tens to hundreds of correlated words. As a result, all of the contained topics and words are not listed, but the graphics presented depict the general algorithm. In reality, there were 25 topics of variable length from 8 words to 50 words.

6 Alignment of statement clusters and topic clusters

The motivating problem is to understand what different groups of users are saying. This requires an alignment of the clusters we found in statement space and the topic clusters we found in word space.

<table>
<thead>
<tr>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
<th>Topic 4</th>
<th>Topic 5</th>
<th>Topic 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>china</td>
<td>gaza</td>
<td>global</td>
<td>iran</td>
<td>corps</td>
<td>drugs</td>
</tr>
<tr>
<td>taken</td>
<td>bank</td>
<td>change</td>
<td>states</td>
<td>health</td>
<td>billions</td>
</tr>
<tr>
<td>president</td>
<td>broker</td>
<td>production</td>
<td>nuclear</td>
<td>water</td>
<td>dollars</td>
</tr>
<tr>
<td>india</td>
<td>palestine</td>
<td>food</td>
<td>weapons</td>
<td>governance</td>
<td>technology</td>
</tr>
<tr>
<td>mutual</td>
<td>conflict</td>
<td>clean</td>
<td>weapon</td>
<td>women</td>
<td>markets</td>
</tr>
<tr>
<td>defense</td>
<td>palestinians</td>
<td>carbon</td>
<td>united</td>
<td>development</td>
<td>healthcare</td>
</tr>
<tr>
<td>european</td>
<td>israelis</td>
<td>climate</td>
<td>usaid</td>
<td>africa</td>
<td></td>
</tr>
<tr>
<td>illegal</td>
<td>sources</td>
<td>girls</td>
<td>economic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pakistan</td>
<td>energy</td>
<td>empowering</td>
<td>world</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>emissions</td>
<td>empowerment</td>
<td>abroad</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Topics selected through clustering (subset of clusters)
6.1 Mathematical Preliminaries

6.1.1 Sufficient Statistic

Let \( X := \{X_1, X_2..., X_n\} \) be \( n \) samples from a probability distribution \( p(x; \theta) \) parametrized by \( \theta \). A sufficient statistic is a function \( h(X) \) of the sample data that satisfies a condition of “sufficiency”, such that there is no other function of \( X \) that can provide additional information about the parameter \( \theta \).

Specific to this project, we can view our textual and statement data as generated from an underlying distribution \( p(x) \). The assumption of my work is that there is a hidden structure in the data that we are trying to uncover, a structure that is captured in the parameter \( \theta \). I use clustering as a sufficient statistic for the underlying parametrization of the distributions that generate the data.

6.1.2 Point-wise Mutual Information

Define mutual information as:

\[
H(x, y) = \log(p(x, y)) - \log(p(x)) - \log(p(y))
\]

Intuitively, this measure quantifies the difference between the co-occurrence of two random variables compared to their marginal occurrences. We can reformulate the \( H(x, y) \) to be slightly more useful:

\[
H(x, y) = \log\left(\frac{p(x, y)}{p(y)p(x)}\right)
\]

\[
H(x, y) = \log\left(\frac{p(y \mid x)}{p(y)}\right)
\]

When aligning our clusters, we will use point-wise mutual information as the objective function. We want to match the occurrence of a topic to a clusters of users in a way that maximizes the mutual information.
Algorithm 3 Aligning users with topics

\[ X := \text{Statement Matrix} \]
\[ Y := \text{Text Matrix} \]
\[ C \leftarrow \text{cluster}(X) \]
\[ T \leftarrow \text{cluster}(Y^T) \]

for each \( C \)

\[ \arg \max_j \log \left( \frac{1}{|C|} \sum_{i \in C} \Phi(T_j, D_i) \right) \]

6.2 Algorithm to align topics and user clusters

Using these mathematical tools defined in Section 5.1, we can develop an algorithm to align the data. Let us first define everything we have developed:

- \( C_i \) - the \( i \)th cluster of users \{1...U\}, a sufficient statistic for the statement responses
- \( T_i \) - the \( i \)th cluster of words \{1...W\}, a sufficient statistic for the textual responses
- \( D_n \) - the textual response vector from user \( n \), where entry \( k \) is 1 if the word is present 0 otherwise. The indexes are kept consistent with the word indexes.

Let us also define the function \( \Phi(T_i, D_n) \) as the indicator function of whether a topic has occurred in a given textual response or not. We will define occurrence as if 3 words from the topic are mentioned in the textual response. Then, for each cluster \( C_k \), I find the topic that that maximizes mutual information[5]:

\[ H(T, C) = \arg \max_j \log \left( \frac{1}{|C_k|} \sum_{i \in C_k} \Phi(T_j, D_i) \right) \frac{1}{|U|} \sum_{i=1}^{|U|} \Phi(T_j, D_i) \]

This optimization problem is non-convex and can only be solved by exhaustively search through all the clusters. This gives motivation to our clustering and dimensionality reduction in our first few sections, rather than working directly with the high dimensional data. Furthermore, dimensionality reduction gives us intuition on the parameter choice and the number of clusters that exist in the dataset. In Section 3, we discussed how there are 1024 possible combinations of statement ratings, a naive overestimation compared to the 22 clusters found would be costly in this step. One of the assumptions about the algorithm is that it is easy to determine cluster membership and topic membership. Depending on the way the data is stored, or its size this may affect the viability of this strategy. Also, this algorithm is general enough to work with any other clustering technique. The algorithm falls into a general family of two-step manifold alignment algorithms.

The motivation for this is that there are some words and topics that occur with very high frequency a priori. As a result, it is important to set expectations for this algorithm such as what kind of results are we expecting, and how meaningful are these results. This optimization problem finds topics that are mentioned in a user cluster at a higher probability than its corpus wide occurrence. In a sense, we are associating a topic that defines a user cluster; and issue that about which many in the cluster are interested. There is a danger, that this algorithm will extract rare topics, and this has to be addressed with parameter selection in the earlier steps to ensure that topics occur with similar frequency. Basically, the results from this algorithm are not a most likely topic, but rather a topic that occurs with an unexpectedly high rate within the cluster.

6.3 Results

After running the alignment algorithm on the clusters produced in Section 3 and Section 4, we find that there are intuitive relationships between the text data and the statement responses. Figure 6 shows the clusters of users and an associated topic from a list of 25 topics. Topics are labeled on the plot with their highest frequency word. Unintentionally, 5/6 questions were framed as Strongly Disagree is the politically conservative response and Strongly Agree is the liberal response, so as a rough heuristic clusters that have a negative X coordinate in (because of mean centering the data) can be considered conservative and on the positive side are liberal. First, we find that clusters that are politically more conservative tend to be related mentions of Iran or Israel. Clusters that are generally
liberal mention a wider variety of topics including climate change, technology, and energy. This is partially due to sample bias, that 71% of the users are on the liberal side.

Some results are less clear, but give us insight into the difficult of text analysis problems. We find that there are two clusters of users associated with discussions about energy one on the conservative side and on the liberal side. One possible explanation, is a discussion of oil with respect to the middle east and a discussion of renewable energy. The hard part about interpreting this algorithm is that we disregard sentiment, and only look at topics that are mentioned. A person who has a positive view on Israel and one who has a negative view are treated the same.

Characterizing the quality of results quantitatively is difficult, and is a subject for future work. A “good” clustering algorithmically, may not be interpretable or exactly what we are looking for.

7 Conclusion

This project demonstrated a sequence of algorithms that uncovered the latent statistical structure in the Opinion Space dataset. Without giving the algorithm human notions of liberal and conservative, and topics, it was able to recover this structure from noisy data. This analysis is not purely academic and it can help us design better versions of Opinion Space for the future. For example even though one of our questions food production, we don’t see topics about food or user clusters based on food views. A more informative question to ask would be about the Arab-Israeli conflict or Iranian nuclear weapons.

From a technical perspective, I demonstrated the robustness of spectral embedding, chosen because we can use the same technique on multiple classes of data just by changing the kernel matrix. Dimensionality reduction helped me reduce the complexity of the problem, visualize my results, and justify my clustering parameter choice. It is important to note, that none of the techniques shown were completely autonomous since all required model selection. Model selection and then validation of the model is simpler in a lower dimensional space. Our results (Figure 7) also hint at a complex manifold structure in the textual data (words correlated with seemingly unrelated topics) with loops and other features that can not be represented with a linear dimensionality reduction technique; a further justification for the Laplacian Eigenmap algorithm—a neighborhood preserving technique. Future work can explore higher dimensional clustering of the data.

Future extensions of this work include: scaling the techniques to larger datasets, sensitivity analysis, and using
the technique for inference. Opinion Space is a rare Social Media system with only 5711 users with very complete data. A possible extension is to a system with missing data and with a much larger amount of users. Many of the techniques I presented are matrix techniques with $O(n^2)$ or $O(n^3)$ time complexity, so a larger system may pose problems. As mentioned repeatedly in this project, the motivation for the Laplacian Eigenmap algorithm is that it is a single flexible framework that can be applied to multiple classes of data. In the future, we can design a system that automatically selects the kernel and parameters that are best suited for the underlying data. A drawback to this algorithm is its sensitivity. Since the technique preserves local structure, intuition suggests that it is very sensitive to noise in a neighborhood around a point. This deserves a further exploration in future work. Finally, we can use this framework to perform inference between clusters.

References


Appendix

Laplacian Eigenmap Algorithm

The intuition behind the Laplacian Eigenmap algorithm is a bit unclear, and it is hard to give meaning to the spectrum of a graph; especially a graph that is artificially constructed from data points. Here, I will work through a toy example of how a spectral embedding can help solve a facility location problem.

Example: Embedding for Facility Location

To gain intuition about the spectral embeddings of a graph, we will introduce a toy example using this algorithm. In this example, Company A has 10 nodes in a distribution network and has constructed a graph representing connections between these nodes. Each edge is weighted by a cost of transit between the nodes. The company decides to add two high speed links between (A,C) and (A,I), and they would like to know where they should locate their warehouse to service all the nodes. To solve this problem we will find an embedding to represent each node in the X,Y plane, and use this embedding to gain insight on this problem.

After running the Laplacian Eigenmap Algorithm we can calculate a 2D embedding. This embedding gives us a snapshot of the entire network reduced to the 2D plane. We see clustering of nodes, outliers, and nodes that are centrally located. With this intuition, we will apply the same algorithm to abstract data where edges represent a similarity metric between two data points.

Runtime of the Algorithm

The most expensive part of the algorithm is an explicit calculation of the kernel matrix. For the Gaussian kernel and the correlation kernel, the time complexity of this step is $O(dn^2)$, where $d$ is the number of features and $n$ is the number of observations to cluster. Note that calculating the Pearson correlation required 2 passes through the data to calculate the means, but this serves to add only a constant factor of additional compute time.

Once the kernel is formed, all other operations can be completed with sparse matrix algebra. Calculation of the eigenvectors is dependent on the condition number of the kernel matrix. When clustering the statements, mixing rate was quite fast. The words on the other hand required over 1000 iterations to converge. Intuitively this makes sense, political views can be parametrized by a small number of dimensions. Conversely, topics are a little bit more complex.

Mixing rate surprising results

As mentioned before the eigenvalue step depends on the condition number of the kernel matrix. I found surprising results by limiting the number of iterations and using results before they converged. The coordinates were less
interpretable, but k-means could still extract the structure from the problem with close to convergent results. I have no mathematical justification for this, and it is worth exploring in future work.

**BidMat Library**

To achieve fast sparse matrix operations, I used the BidMat Library (http://bid.berkeley.edu/BIDMat/index.php/Main_Page). The library exploits native processor and gpu libraries to increase performance. The library is a work in progress and some of the future additions such as “butterfly mixing” will be useful in my research. Initially, I was skeptical about the performance gains, but it turned out to be very effective. MATLAB requires almost 10 minutes of computation to cluster 5711 points, while this requires 9.78 secs.

**Word Stemming**

I also attempted stemming words in an effort to remove noise from the text processing, this gave me very poor results. My intuition is that similar words are referred in multiple contexts. For example, west and western, the word west was strongly correlated with the word bank and consequently the Arab-Israeli topic. Western was strongly correlated with the word culture, and the topic about empowerment of women. Stemming would break these relationships.