Dimensionality Reduction for Visualization of Text Similarity

Russell Yanofsky (rey4@columbia.edu)

December 13, 2007

Abstract

Techniques which express high dimensional data in a reduced number of dimensions while maintaining a useful amount of the information content of the original data are useful in data processing and analysis. This paper explores a somewhat unlikely application of dimensionality reduction techniques, applying techniques extremely well suited to visualization of data in two or three dimensions to a collection of text documents in English, which comprise a data set that is not normally subject to 2D plotting and spatial analysis. The techniques used are Semidefinite Embedding (SDE) and Minimum Volume Embedding (MVE), which are both based on the general technique of Kernel Principal Components Analysis (KPCA). A preprocessed representation of text based on word frequency (disregarding word order) is used as input to these algorithms. This type of representation has been shown in other work to be effective at text categorization with kernel methods, and in an experiment with a small number of related documents, gives good visualization results with a simple RBF kernel, though not much improvement with the more powerful SDE and MVE techniques.

1 Introduction

Textual data is among the most ubiquitous forms of data that exist today, and automated means of processing it, performing operations such as searching, categorization, translation, and correction, have had great success and are seeing constant improvements. But the sheer amount of text data means that even after text is automatically searched, categorized, translated, and corrected, navigating it may not be easy. To help manage this problem, visualization techniques have the potential to be applied to provide an aide to human navigation. The idea behind visualization for this purpose is to show documents which are related to each other as points clustered together in a two or three dimensional space.

Visual representations like this could have uses in certain types of research, for example, in helping to sort through large archives of historical documents. They could also be used on the Internet, as accompaniments to search results and the automated recommendations commonly given on book and movie websites. Finally, they might have use in the testing and development of other natural language processing algorithms, providing a tool to summarize large amounts of data, and to help evaluate experimental results.

Different visualization techniques have been applied in the past to the problem of document visualization, sometimes using ad-hoc methods. For example, one system [2] uses a feedback method, positioning points representing documents in a space, and iteratively moving the documents in space closer together or farther apart, depending on similarity scores. Another system [3] uses a visualization technique called Sammon mapping to compute a similar type of low dimensional representation.

This paper presents the application of two newer visualization techniques, Semidefinite Embedding (SDE) [10], and Minimum Volume Embedding (MVE) [8] to document visualization, and shows the results of applying these techniques to a small data set.

2 PCA Dimensionality Reduction Techniques

The dimensionality reduction techniques used in this paper can be understood in the context of Principal Component Analysis (PCA). PCA is a widely used technique for dimensionality reduction. In its ideal scenario, a Gaussian distributed (ellipsoid shaped) cloud of M points, $\overrightarrow{x_j}$, is distributed around some mean point $\overrightarrow{\mu}$. Computing the covariance matrix of the points by

$$\Sigma = \frac{1}{M}\sum_{j=1}^{M}\left(\overrightarrow{x_j} - \overrightarrow{\mu}\right)\left(\overrightarrow{x_j} - \overrightarrow{\mu}\right)'$$

gives the maximum likelihood estimate of the Gaussian distribution's covariance parameter. Given this covariance parameter, the shape and orientation of the Gaussian distribution is known. In particular, the orientation of the axes of the ellipsoidal shape described by the Gaussian distribution can be found by computing the eigenvectors of the covariance matrix,

$$\Sigma\overrightarrow{v_j} = \lambda_j\overrightarrow{v_j}$$

. The longest ellipsoid axis, pointing along the direction in the cloud where the data points have the largest variance, is described by the eigenvector with the largest corresponding eigenvalue. Axes pointing in directions where data has progressively less variance are described by eigenvectors that have progressively lower eigenvalues. Projecting data points onto axes that have the orientation of the eigenvectors, and that all intersect at the mean position of all the points, and then taking the position along each axis as a new set of coordinates for each point, gives an alternate representation of the data from which it is possible to reconstruct the original positions of the points. With the eigenvectors of C expressed as columns of a matrix $V = \begin{pmatrix} v_1 & \cdots & v_D \end{pmatrix}$, and the mean-centered, D-dimensional data points given as rows of a matrix $X = \begin{pmatrix} x_1 - \overline{x} \\ \vdots \\ x_M - \overline{x} \end{pmatrix}$, the alternate representation is given by

$$Y = XV$$

, where the transformed coordinates of each point can be found in rows of the matrix Y.

The coordinates of the transformed data points for axes with low eigenvalues will be, intuitively, close to 0, because variances along these axes are low, and because the axes pass through the mean of the data. Because of this, a low dimensional representation of the data can be found by simply dropping these coordinates, and only keeping the coordinates of the alternate representation corresponding to axes with high eigenvalues. If the eigenvalues of the dropped coordinates are very low, it will still be possible to construct good approximations of the original points by substituting 0 for those values when reversing the transformation.

PCA gives the best possible low dimensional representation of a high dimensional data set, when the transformation between the two data sets has to be a linear transformation. Kernel PCA generalizes normal PCA to allow non-linear transformations between the low and high dimensional representations, providing better performance in many cases.

Kernel PCA

Kernel PCA [6] works, conceptually, by allowing the high-dimensional original data points, $\overrightarrow{x_j}$, to be mapped to an even higher dimensional intermediate form by an arbitrary function $\phi\left(\overrightarrow{x_j}\right)$, before finding the final low-dimensional representation through the normal process of PCA. To make this idea both computationally feasible and even more general, Kernel PCA recasts the PCA formulas for finding the alternate coordinate representation, so they are expressed in terms of dot products $\phi\left(\overrightarrow{x_j}\right)\cdot\phi\left(\overrightarrow{x_k}\right)$, and individual $\phi\left(\overrightarrow{x_j}\right)$ terms never appear alone. The use of dot products is important because for many useful ϕ mappings into high- (or infinitely-) dimensional spaces, it is possible to compute the dot product between two points from the lower dimensional $\overrightarrow{x_j}$ and $\overrightarrow{x_k}$ values without ever actually computing the high dimensional mappings. Functions which compute these types of dot products are called kernels, and are denoted:

$$k\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right) = \phi\left(\overrightarrow{x_j}\right)\cdot\phi\left(\overrightarrow{x_k}\right)$$

The process of performing Kernel PCA is not much more difficult than the process of performing normal linear PCA, although it does not lend itself as easily to geometric intuition. The first step in performing Kernel PCA is to compute a kernel matrix, K, whose elements are given by the following for all data points:

$$K_{ij} = k\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right)$$

The second step is to "center" the kernel matrix. This step can seem abstract, but is really just the equivalent of subtracting out the mean $\overrightarrow{\mu} = \sum_{i=1}^{M} \phi\left(\overrightarrow{x_i}\right)$ of the intermediate data points, so the dot products are computed as $\left(\phi\left(\overrightarrow{x_j}\right) - \overrightarrow{\mu}\right) \cdot \left(\phi\left(\overrightarrow{x_k}\right) - \overrightarrow{\mu}\right)$. The efficient way of centering the kernel matrix, without having to compute the high dimensional mappings, is to compute:

$$\widetilde{K}_{ij} = K_{ij} - \frac{1}{M}\sum_{m=1}^{M} K_{mj} - \frac{1}{M}\sum_{n=1}^{M} K_{in} + \frac{1}{N^2}\sum_{m,n=1}^{M} K_{mn}$$

Note that for a linear kernel, the centered $\widetilde{K}$ matrix is just the $M \times M$ Gram matrix of the centered data points. This contrasts with the computation of a $D \times D$ covariance matrix during the first step in Linear PCA, and can mean data storage requirements will be lower or higher in Kernel PCA depending on whether the number of points in the data set are lower or higher than the number of dimensions.

The next step in Kernel PCA is to find the eigenvalues and eigenvectors of the $\widetilde{K}$ matrix,

$$\widetilde{K}\overrightarrow{v_j} = \lambda_j \overrightarrow{v_j}$$

, with the eigenvectors normalized so $\lambda_j\left(\overrightarrow{v_j} \cdot \overrightarrow{v_j}\right) = 1$ for all j.

As a final step, the transformed representation, Y, of the original data points may be found by:

$$Y = V\Lambda$$

where the transformed representation of each point forms a row in Y, where $V = \begin{pmatrix} \overrightarrow{v_1} & \cdots & \overrightarrow{v_D} \end{pmatrix}$ is a matrix of the normalized eigenvectors as columns, and where Λ is a matrix with the square root of eigenvalues $\sqrt{\lambda_1} \ldots \sqrt{\lambda_M}$ along its diagonal [9, 7]. Low dimensional representations can be found by sorting the eigenvalues and eigenvectors in descending order of the eigenvalues, and skipping computation of the rightmost columns of matrix Y.

In addition to computing low dimensional representations of the original data points with Kernel PCA, it is also possible to compute representations of arbitrary points [6], although it is not necessary to do so in the application described in this paper.

Semidefinite Embedding

Kernel PCA provides a powerful visualization technique, but it leaves open the question of what type of kernel function should be used with it. The kernel function is what defines the mapping of the original high-dimensional data into the intermediate higher-dimensional space where PCA takes place. For data that is Gaussian distributed, in an elliptical cloud with low variance in some directions, and high variance in others, the linear kernel $k\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right) = \overrightarrow{x_j} \cdot \overrightarrow{x_k}$ may work very well. For data occupying more complicated shapes, polynomial $\left(k\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right) = \left(\overrightarrow{x_j} \cdot \overrightarrow{x_k} + 1\right)^p\right)$ and RBF kernels $\left(k\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right) = e^{-\left\|\overrightarrow{x_j} - \overrightarrow{x_k}\right\|^2 / 2\sigma^2}\right)$ may provide better results. There are also many domain specific kernels, designed to work on image, text, or speech data, for example.

One limitation of many kernels for the purpose of visualization is that they often provide poor measures of affinity between distant data points which are not closely related to each other, even when they do give useful measures for points that are closer together. For example, it is possible to imagine a text kernel that gives a useful measure of similarity between related documents that have the same topic and structure, but provide less meaningful scores between documents that are more different. This can be a problem any time a closed-form kernel is used on a high-dimensional, but highly constrained data sets like English text, or natural images. Data points in these types of data sets tend to occur on twisted, low dimensional manifolds within the high dimensional vector space used to describe them. Effective visualizations for these data sets need to show the relative position of data points along the manifolds in which they occur, instead of their positions in the larger

vector space, and the transformations done by closed-form kernels may not be able to accurately capture the shape of the manifold when giving the affinities for points which are not close together.

Semidefinite Embedding (SDE) [10] attempts to improve this situation by learning a Kernel matrix which effectively flattens out the manifold. It relies on normal dot product computation to find affinities between points which are closely related to each other and then it stretches, or unfolds the manifold by attempting to maximize the distance between all data points which are not linked together. As input, the SDE algorithm takes a connectivity matrix, η, denoting the points are presumed to lie close to each other on the manifold, and a Gram matrix, G, describing the affinity between points that are directly connected, or are connected through a neighbor. Semidefinite programming, a convex optimization technique, is used to an optimal matrix, which is plugged in as the Kernel matrix in Kernel PCA to yield the desired visualization.

The semidefinite programming problem is described by an objective function to be maximized, and a list of constraints. The first constraint is that the outputted kernel matrix K, is positive semidefinite ($K \succ 0$). This constraint is natural because by definition, all kernel matrices (matrices whose elements are expressed as dot products of high dimensional mappings, as in $K_{ij} = \phi(\overrightarrow{x_j}) \cdot \phi(\overrightarrow{x_k})$) are positive semidefinite. The second constraint is $\sum_{ij} K_{ij} = 0$, which limits the output to the set of "centered" kernel matrices which give meaningful results with Kernel PCA. The third constraint is expressed as $K_{ii} + K_{jj} - K_{ij} - K_{ji} = G_{ii} + G_{jj} - G_{ij} - G_{ji}$, and must hold for all points i and j which are neighbors of each other or of a common point. This third constraint is what preserves the distances and local relationships between sets of connected points. The objective function to be maximized is $tr(K)$. Maximizing this value maximizes the distance between all unconstrained points, and is what causes the manifold to be pulled apart, unfolding and flattening it.

Because SDE takes an arbitrary Gram matrix as input, any kernel function can be used to compute the local affinities that SDE preserves in the final embedding. To compute the connectivity matrix input, k-nearest neighbors, or any other clustering algorithm may be used.

Minimum Volume Embedding

One weakness of SDE is that the objective function it maximizes, $tr(K)$, increases the distance between unconnected points in every dimension, instead of just the two or three dimensions being visualized. Since we only ever see the distances between points in the dimensions associated with the highest eigenvalues of the kernel matrix, this is not the objective function we really seek to maximize. As such, there are cases where SDE will not do an optimal job reshaping the manifold for visualization, such as when the connectivity graph has a hub-and-spokes structure.

Minimum Volume Embedding (MVE) [8] can provide improved results by maximizing distances between points in the first d visible dimensions while simultaneously minimizing the distances between points in all other dimensions. This corresponds to maximizing the value of

$$\sum_{i=1}^{d} \lambda_i - \sum_{i=d+1}^{N} \lambda_i$$

, which is a modification of the objective function used in SDE, recalling that $tr(K) = \sum_{i=1}^{N} \lambda_i$.

Semidefinite programming combined with an iterative alternative minimization scheme make MVE more computationally expensive to implement than SDE, but with better results for certain types of visualizations.

3 Experiment

Text Representation

A relatively simple text representation, which has been shown to give good results for document classification with kernel methods [4], was used in this project, and gave good results for document visualization with Kernel PCA techniques. Each document was represented as a vector of word counts. During initial parsing, stop-words like "and", "or", and

"the" were removed using an English stop-word list [5], and a stemming algorithm [1] was used to map the related words with different suffixes (for example, "computes", "computing", and "computer"), into a canonical word-stem form, so their counts would be added up in the resulting feature vectors. The resulting word count vectors were normalized to sum to one to allow for more meaningful comparisons of documents with different lengths. The word counts were then weighted by *inverse document frequency* (IDF) values computed by

$$IDF(w_i) = \log\left(\frac{n}{DF(w_i)}\right)$$

for each word w_i where n was the total number of documents in the set, and $DF(w_i)$ was number of documents the word appeared in (the *document frequency*). Weighting by IDF helps give words which only occur in some documents, which can be good indicators of subject matter, more influence over the visualization results.

This vector representation was used with an RBF kernel to compute document affinities. The RBF kernel was chosen because it has been shown to work well with similar text representations for text classification. A k-nearest neighbors algorithm was used to compute the connectivity matrices passed as input the SDE and MVE implementations.

Data Set

The data set used was a collection of issue documents from the 2008 presidential candidates' campaign web sites. Each document was labeled for the its topic (environment, health care, foreign policy, etc.), and the candidate whose views it expressed (Clinton, Giuliani, etc.) Having a data set with two distinct labels for each point was expected to make visualization more interesting, and to make it easier to look for patterns in output.

Results

The results of using Kernel PCA with an RBF kernel, and Kernel PCA with SDE and MVE kernels are shown in Figure 1. The visualizations produced by these methods were all very similar, with the SDE and MVE visualizations being completely indistinguishable. In the graphs, documents are pretty clearly clustered together by topic. It was harder to find any clear correlation between the position of documents in the graph and the candidates' websites they originated from.

Adjusting the sigma parameter used in the RBF kernel did not seem to have a discernible impact on visualization, unless the values were extremely large or extremely small, in which cases the visualization ceased to provide meaningful results. Adjusting the number of neighbors found by the k-nearest neighbor algorithm also did not have much effect on results once they were set above around 20% of the total number of documents.

4 Conclusion

The Kernel PCA methods used in this project were able to generate meaningful visualizations of text documents using a very simple documentation which was based solely on individual word counts, and did not take into account word order or document structure. It would be interesting to see if visualization could be improved with a more sophisticated representation that did take these things into account, and to see if the manifold-flattening SDE and MVE techniques would give better results than the simple RBF kernel, given more complex and constrained representations.

It would also be better to test these visualization techniques on larger data sets than the one used in this experiment, and to test with data sets containing different types of text, for example newspaper articles, encyclopedia articles, web pages, to see how well the visualization is able to cluster related documents written in different styles and formats.

References

[1] The snowball project. http://snowball.tartarus.org/, 2007.

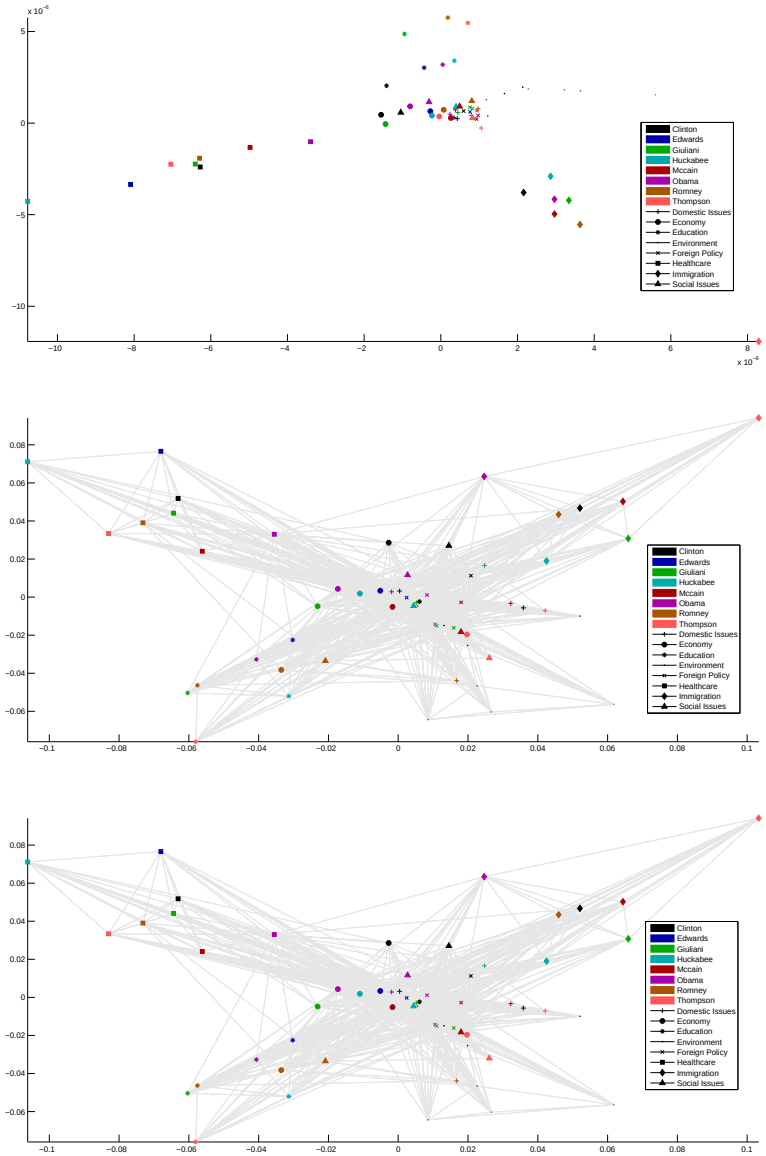

Figure 1: Visualization Results using an RBF kernel with Kernel PCA (top), Semidefinite Embedding (middle), and Minimum Volume Embedding (bottom)

[2] J. Allan, A. Leouski, and R. Swan. Interactive cluster visualization for information retrieval. Technical Report IR-116, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, MA, 1997.

[3] M. Carey, D.C. Heesch, and S.M. Ruger. Info navigator: A visualization tool for document searching and browsing. 2003.

[4] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, 1998.

[5] M. Sanderson. Stop word list. http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/, 2007.

[6] B. Scholkopf, A.J. Smola, and K.R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[7] B. Shaw. Mve v0.2. http://www.metablake.com/mve/, 2007.

[8] B. Shaw and T. Jebara. Minimum volume embedding. In *Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[9] L. van der Maaten. Matlab toolbox for dimensionality reduction. http://www.cs.unimaas.nl/l.vandermaaten/Laurens_van_der_Maaten/.

[10] K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.