

Principle Component Analysis and other data reduction techniques

Vincent A. Voelz

E-mail: vvoelz@gmail.com

[†]*Math Bio Boot Camp 2006*

University of California at San Francisco, San Francisco, CA 94143

August 28, 2006

How can we make meaningful sense of large data sets? The goals of this lecture are to:

1. Discuss some ways to use projection methods to analyze large data sets
2. Introduce some practical examples that we will work on using MATLAB (“eigenface” lab)

1 Data Reduction Techniques

1.1 Principal Component Analysis (PCA)

Principle Component Analysis is useful for large, multi-dimensional data sets. It seeks to find the eigenbasis that represents greatest amount of variation in the data. As we will see, it does this by diagonalizing a covariance matrix.

This basis can be used to discover trends in the data, as in: What linear combination of variables captures the greatest amount of variation in my data? PCA is also useful in finding the best low-dimensional projection of the data, by projecting onto a small subset of eigenvectors with the largest eigenvalues.

Suppose we have a number of T data points, $\mathbf{x}_j, j = 1..T$, where each data point represents N different variables in a high-dimensional space. We can organize the data into an $N \times T$ matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_T \end{pmatrix}$$

Next, we wish to consider the covariance matrix \mathbf{C} , where $\mathbf{C}_{ij} = \text{Cov}(X_i, X_j)$. Recall from basic statistics that the covariance of two variables is the expectation value of their (mean-centered) products:

$$\text{Cov}(X_i, X_j) = E((X_i - \mu_i)(X_j - \mu_j))$$

We can thus write the covariance matrix \mathbf{C} as:

$$\mathbf{C} = \begin{pmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_N) \\ \text{Cov}(X_2, X_1) & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \text{Cov}(X_N, X_1) & \dots & \dots & \text{Cov}(X_N, X_N) \end{pmatrix} = \frac{1}{T}(\mathbf{X} - \mathbf{M})(\mathbf{X} - \mathbf{M})^T$$

where $\mathbf{M}_{ij} = \frac{1}{T} \sum_j \mathbf{X}_{ij}$.

Note that because $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$, \mathbf{C} is symmetric, so it must have orthonormal basis vectors and real eigenvalues. Also, because \mathbf{C} is the inner product of a matrix $(\mathbf{X} - \mathbf{M})$ with itself, the eigenvalues must also be positive.

Transforming into eigenbasis coordinates, we have:

$$\mathbf{\Lambda} = \mathbf{U}^T \mathbf{C} \mathbf{U}$$

The column vectors of \mathbf{U} are the *principal components*. The significance of each principal component \mathbf{u}_j is given by its corresponding eigenvalue, λ_j , which characterizes the amount of variance along the direction of the eigenvalue.

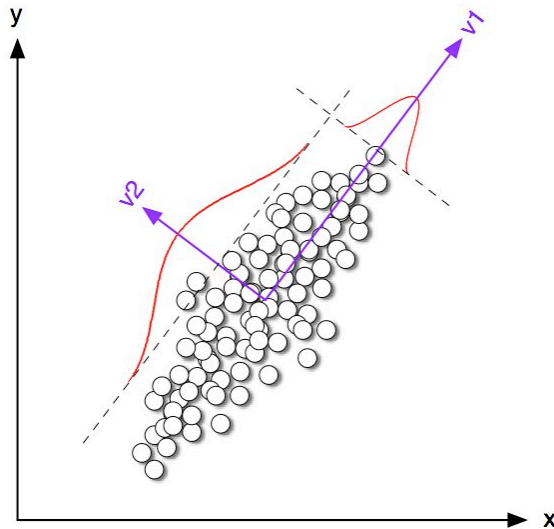


Figure 1: The principal components of some bogus fabricated data.

Projecting to the first few principal components. Suppose we want to take all our multidimensional data, and make a classy-looking 2-D plot for publication. To do this, we wish to project all of our data \mathbf{X} onto the first two principal components, often called PC1 and PC2. We can shuffle the column vectors of \mathbf{U} so the eigenvalues in $\mathbf{\Lambda}$ are in descending order: $\lambda_1 > \lambda_2 > \dots \lambda_N$. and only consider an $N \times 2$ matrix of just the first two column eigenvectors, setting the rest to zero.

$$\mathbf{U}' = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{pmatrix}$$

To get our (mean-centered) data into the PC1-PC2 coordinate system, we project onto these first two eigenvectors:

$$(\mathbf{U}')^T(\mathbf{X} - \mathbf{M})$$

resulting in a $2 \times T$ matrix \mathbf{T}' containing all the PC1-PC2 coordinates of the data.

1.1.1 Examples

Essential Dynamics. Essential Dynamics is PCA applied to molecular dynamics simulations [1]. Consider a set of T snapshots taken for a large macromolecular simulation. For an N -atom molecular simulation, there are $3N$ degrees of freedom, so the snapshot data can be organized into a $N \times T$ matrix, from which the covariance matrix is calculated and diagonalized.

The top principal components especially useful for finding large coordinated “breathing” motions of a protein that are separate from much of the tightly-constrained harmonic motion of more constrained atoms. One way to speed up molecular simulations is to rigidly constrain the smallest principal components and only simulate dynamics in a reduced space consisting of most important principal components.

1.1.2 Eigenfaces

Eigenfaces [2], [3] are an excellent example of how large data sets can be ‘compressed’ into the most meaningful principal components for the purposes of pattern recognition. In the eigenface problem, our data consist of a series of images of human faces. The pixel-by-pixel data defining an image can be thought of as a 1-D vector in a very high-dimensional space. Suppose we have T images each composed of N pixels. It is straightforward to organize the data into a $N \times T$ matrix \mathbf{X} , and perform PCA:

$$\mathbf{C} = \frac{1}{T}(\mathbf{X} - \mathbf{M})^T(\mathbf{X} - \mathbf{M})$$

where $\mathbf{M}_{ij} = \frac{1}{T} \sum_j \mathbf{X}_{ij} = \mu_i$. μ is the average of the all the image vectors, and is the *average face*.

Diagonalizing \mathbf{C} , we get:

$$\mathbf{\Lambda} = \mathbf{U}^T \mathbf{C} \mathbf{U}$$

The column vectors in \mathbf{U} are image vectors that represent the most variation across all the images. These are the *eigenfaces*. We call the eigenface with the largest eigenvalue the *first difference* eigenface, the second largest the *second difference* eigenface, and so on.

Face recognition. A PCA-based face recognition algorithm only stores in memory the top $P < T$ eigenfaces, and a library of face images projected down to P -dimensional eigenface-coordinates. Then, when presented with a new face image \mathbf{y} , the algorithm calculates the P -dimensional eigenface-coordinates of the image:

$$\mathbf{y}' = (\mathbf{V})^T(\mathbf{y} - \mu)$$

where \mathbf{V} is an $N \times P$ matrix whose column vectors are the top P eigenfaces. The face in the library \mathbf{f}' with coordinates closest to \mathbf{y}' is the best match.

Familiarity. Instead of projecting a new face into eigenface coordinates, we can project onto a basis consisting of other faces. Let's say you have a library of face images of all the other people in your class, and you want to determine who you most resemble. Consider an $N \times Q$ matrix \mathbf{V} whose column vectors consist of these Q other faces. Then you can project your face \mathbf{y} into the coordinate system of other (mean-centered) faces:

$$(\mathbf{V} - \mathbf{M})^T(\mathbf{y} - \mu)$$

The largest positive coordinate is along the face you most resemble.

Note that because (usually) $Q \ll N$, there is no inverse to this "familiarity" transformation, but if the data themselves can be well-described using only a few principal components, then it may be possible to reconstruct a recognizable version of your own face from a linear combination of your classmates' faces. We will do some hands-on testing of this idea in MATLAB.

Multi-Dimensional Scaling (MDS). Many times we have data that is defined in terms of inter-datum *distances*. For example: To determine protein and nucleic acid structures by nuclear magnetic resonance, a number of distance-dependent NOEs are measured between many pairs of atoms. The notion of 'distance' can be more abstract as well. Perhaps we have a set of genes that have various percent sequence similarity to with each other. Or perhaps we have a set of chemical ligands that each have a similarity metric to all the others, based on certain chemical descriptors.

In each of these examples, it would be desirable to find a low-dimensional space in which we can embed the data, that would optimally preserve the distances between data points. This is an especially useful way to visualize the data. The most straightforward way to do this is to consider a (squared) distance matrix, \mathbf{D} , where each element \mathbf{D}_{ij} is the (squared) distance from point i to j . MDS seeks to find an low-dimensional eigenbasis that minimizes any distortions to \mathbf{D} .

The eigenproblem is solved using an optimization procedure that minimizes some cost function, $\varepsilon(\mathbf{D}' - \mathbf{D})$. There are many variations of MDS, each with different cost functions. The cost function may enforce strict adherence to Euclidean distances, or perhaps only enforces the rank ordering of the distances, for visualization purposes.

Distance Geometry. In the case of NMR structure determination, the NOE measurements may be incomplete, and/or contain many errors. In general, n different distances can always be embedded in an n -dimensional space, so if we were to take the data at face value, the structure coordinates could only be realized in a high-dimensional space. Lucky for us, we know that the resulting structure resides in 3-dimensional space, and can solve the structure by finding the best three-dimensional projection of the data.

In practice, there is an effective MDS recipe for this [4], [5], consisting of 1) projecting the data to the best 3-dimensional coordinate system, and 2) further optimizing in 3-D, perhaps with a molecular forcefield.

For the first step, consider set of vectors $\{\mathbf{x}_i\}$ for which we know all the inter-datum (Euclidean) distances. We can construct a distance matrix \mathbf{D} , where each entry D_{ij} is the squared distance from \mathbf{x}_i and \mathbf{x}_j .

$$D_{ij} = d_{ij}^2$$

It turns out that we can define the inner product of vectors \mathbf{x}_i and \mathbf{x}_j entirely from distance measurements.

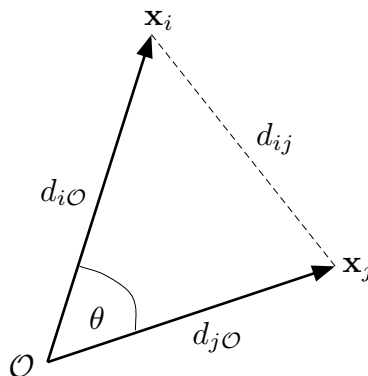


Figure 2: Distances that can be defined between two vectors \mathbf{x}_i and \mathbf{x}_j . The \mathcal{O} vector is the origin, which is arbitrary, although typically the mean $\mu = \langle \mathbf{x} \rangle$ is used.

To see this, consider the following identity (see Figure 2), due to the Law of Cosines:

$$d_{ij}^2 = d_{i\mathcal{O}}^2 + d_{j\mathcal{O}}^2 - 2|d_{i\mathcal{O}}||d_{j\mathcal{O}}|\cos\theta$$

The last term can be written in terms of the the inner product between \mathbf{x}_i and \mathbf{x}_j , and rearranged:

$$\begin{aligned} d_{ij}^2 &= d_{i\mathcal{O}}^2 + d_{j\mathcal{O}}^2 - 2\mathbf{x}_i^T \mathbf{x}_j. \\ \mathbf{x}_i^T \mathbf{x}_j &= \frac{1}{2}(d_{i\mathcal{O}}^2 + d_{j\mathcal{O}}^2 - d_{ij}^2) \end{aligned}$$

Next, we construct a matrix \mathbf{G} such that $\mathbf{G}_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. Now, if we *knew* the (mean-centered) molecular coordinates \mathbf{X} (a $3 \times N$ matrix containing all atomic coordinates), we could write \mathbf{G} as:

$$\mathbf{G} = \mathbf{X}^T \mathbf{X}$$

Alternatively, we can diagonalize \mathbf{G} into its eigenbasis:

$$\mathbf{G} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$$

where \mathbf{U} has column eigenvectors, and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues. Equating the last two equations, we get:

$$\mathbf{U} \mathbf{\Lambda} \mathbf{U}^T = \mathbf{U} \mathbf{\Lambda}^{1/2} \mathbf{\Lambda}^{1/2} \mathbf{U}^T = \mathbf{X}^T \mathbf{X}$$

and since $\mathbf{\Lambda}^{1/2} \mathbf{U}^T = (\mathbf{U} \mathbf{\Lambda}^{1/2})^T$,

$$\mathbf{X} = \mathbf{\Lambda}^{1/2} \mathbf{U}^T$$

$\mathbf{\Lambda}^{1/2}$ is a diagonal matrix with $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots$ along the diagonal. If the distance data is good enough, there will be three eigenvalues in $\mathbf{\Lambda}$ that are much larger than the rest. These correspond to the three principal dimensions of our three-dimensional world. We then project our $N \times N$ candidate matrix \mathbf{X} of atomic coordinates down to these first three principal components by computing

$$\mathbf{X}' = \mathbf{V}^T \mathbf{X}$$

where \mathbf{V} is an $N \times 3$ matrix consisting of the three principal eigenvectors as column vectors. The resulting \mathbf{X}' is a $3 \times N$ matrix containing the best guess of the 3D coordinates of the molecule. These coordinates can be further optimized using computer algorithms, possibly with molecular forcefields.

1.2 Local Linear Embedding (LLE)

Of course, there are limitations to PCA-based techniques for data reduction. One drawback of PCA is that it assumes that the underlying data are spread like a multi-variate Gaussian throughout the high-dimensional space, such that the covariance measure accurately captures the variation in the data. But what if the data reflect a more complicated non-linear relationship? A good example of this is a data set like the kind in Figure 1 of ref. [6]. The data lie on a two-dimensional surface, but this manifold is curled up in three dimensions. PCA would predict three principle components, but the effective dimensionality of the data is 2D.

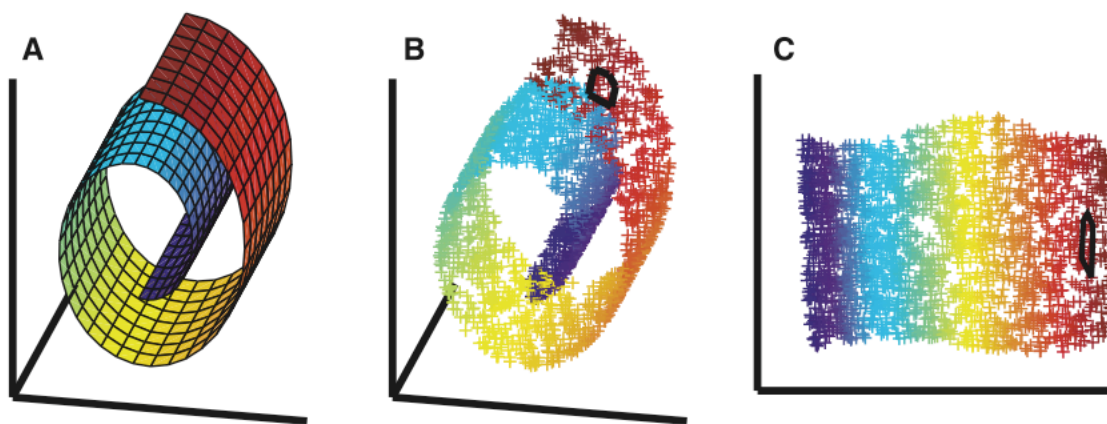


Figure 3: Figure 1 of ref. [6].

Local Linear Embedding (LLE) is a method that deals with this issue by finding principal components in very local neighborhoods of the data, and then finding the best set of low-dimensional eigenvectors to embed the data [7].

References

- [1] Andrea Amadei, Antonius B. M. Linssen, and Herman J.C. Berendsen. Essential dynamics of proteins. *PROTEINS: Structure, Function, and Genetics*, 17:412–425, 1993.
- [2] M. Turk and A. Pentland. Face recognition using eigenfaces. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.
- [3] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Computational Neuroscience*, 3(1):71–86, 1991.
- [4] Timothy F. Havel, Irwin D. Kuntz, and Gordon M. Crippen. The theory and practice of distance geometry. *Bulletin of Mathematical Biology*, 45(5):665–720, 1983.
- [5] Andrew R. Leach. *Molecular modelling: principles and applications. Chapter 9.5*. Pearson Education Ltd., 2nd edition, 1996.

- [6] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.
- [7] Lawrence K. Saul and Sam T. Roweis. An introduction to locally linear embedding. Technical report, AT&T Labs – Research, 2001.