Principal Tangent Data Reduction

Thomas Hunt and Arthur J. Krener
Department of Mathematics
University of California
Davis, CA 95616-8633, USA
and
Department of Applied Mathematics
Naval Postgraduate School
Monterey, CA 93943-5216, USA

***Abstract*— There is a need to be able to find patterns in high dimensional data sets. Often these patterns are described as lower dimensional manifolds possibly of varying dimension that more or less fit the data. We present a new algorithm for doing this. It is a form of nonlinear principle component analysis.**

Keywords: Nonlinear dimension reduction, manifold learning

I. Introduction

Suppose we are given a large number of data points in a high dimensional Euclidean space. Many algorithms have been proposed for reducing the apparently high dimensional data. Two of the best known are the Isomap technique of Tenenbaum, de Silva and Langford [1] and the Locally Linear Embedding technique of Roweis and Saul [3], [4]. The Isomap technique seeks to preserve the distances between data points while Local Linear Embedding seeks to preserve the linear relationships between nearby points. For a more complete review of the literature see Section 7 of [4] and the recent paper [2].

Our approach is to find a low dimensional piecewise linear manifold near which most of the points lie using a form of local principal component analysis. For ease of exposition we shall describe the algorithm when the low dimensional manifold is two dimesional and lies in three space but its extension to higher dimensions is straightforward at least in theory if not in computation.

II. PTDR

We start with an initial data point x_0 and compute its local covariance as follows. We choose a length scale l that depends on the density and curvature of the data. We define the neighbors of x_0 to be all data points x_d within $2l$ of x_0. The length scale l is chosen so that this neighborhood contains ten to twenty data points. The local covariance of the data at x_0 is the matrix P_0 defined to be

$$P_0 = \sum_d \frac{(x_d - x_0)(x_d - x_0)'}{|x_d - x_0|^2}$$

Research supported in part by AFOSR.

where the sun is over all neighboring data points.

The normalizing factor $|x_d - x_0|^2$ in the denominator in effect weights all points in the neighborhood equally. This is because we are primarily interested in the local directions of the data near x^0 not their magnitudes. An alternative definition that weights closest points the most is to make the denominator $|x_d - x_0|^\alpha$ where $\alpha > 2$.

We assume that the local covariance has two relatively large eigenvalues and its third eigenvalue is relatively small. This means that the data is locally two dimensional. We refer to the two large eigenvalues and their associated eigenvectors as the principal eigenvalues and principal eigenvectors.

Four triangles are constructed around the initial point x_0 using the two unit principal eigenvectors $v_1,\ v_2$ as follows. We find the four data points $x_1,\ x_2,\ x_3$ and x_4 that are closest to $x_0 + lv_1$, $x_0 + lv_2$, $x_0 - lv_1$ and $x_0 - lv_2$. These along with x_0 form the vertices of the four triangles. Lists of vertices, edges and triangles are stored with the boundary vertices, edges and triangles noted.

Then starting with a boundary edge which we call the active edge, we construct a new triangle as follows. Suppose the vertices of this outside edge are $x_1,\ x_2$, along with x_0 they are the vertices of one the four triangles that we have already constructed. We compute the local covariances $P_1,\ P_2$ at $x_1,\ x_2$ and find the point x that solves minimizes

$$(x - x_1)'P_1^{-1}(x - x_1) + (x - x_2)'P_2^{-1}(x - x_2) \tag{1}$$

subject to

$$l = |x - x_m| \tag{2}$$

and

$$(x - x_m)'(x_0 - x_m) \leq 0 \tag{3}$$

where $x_m = (x_1 + x_2)/2$.

A word of explanation is in order. Minimizing the objective (4) tends to put the new point in the directions from $x_1,\ x_2$ of the principal eigenvectors of the covariances. The first constraint (5) forces the next vertex to be of order l away from $x_1,\ x_2$. The second constraint (5) encourages the next vertex to be exterior to the already found triangles.

After finding the minimizing x we choose the data point x_k that is closest to it and construct new edges $[x_1, x_k]$, $[x_2, x_k]$ and a new triangle $[x_1, x_2, x_k]$.

One could solve the constrained minimization problem where x is restricted to be a data point but we found that this leads to very skewed triangles if l is not very small. In other words the minimizing data point tends to lie close to either x_1 or x_2 but not both.

At the general step we pick up a boundary edge called the active edge with vertices $x_i,\ x_j$. Together with an interior vertex x_k they form an already found triangle. We compute the local covariances $P_i,\ P_j$ at $x_i,\ x_j$ and find the data point x that solves minimizes

$$(x - x_i)' P_i^{-1} (x - x_i) + (x - x_j)' P_j^{-1} (x - x_j) \tag{4}$$

subject to

$$l = |x - x_m| \tag{5}$$

and

$$(x - x_m)'(x_k - x_m) \leq 0 \tag{6}$$

where $x_m = (x_i + x_j)/2$. The constant l is set to

$$\sqrt{3}|x_i - x_j|/2$$

because this is what it would be if the newly constructed triangle were equilateral. We desire that our triangles be as close to equilateral as possible.

Suppose the minimizer is x then we find the data point x_d that is closest to x. The proposed new triangle is $[x_i, x_j, x_d]$. If the proposed triangle overlaps with an existing triangle, then mark the edge $[x_i, x_j]$ as problematic and start over by marking another boundary edge as active.

If the proposed triangle does not overlap with an existing triangle then check that the new vertex is close to any existing vertices. If it is closer than $l/2$ to an existing vertex which is no more than one edge away from the active edge then we move the proposed vertex to the existing vertex provided that the new triangle does not overlap any existing triangle and if the angle between the planes of the the new and old triangle is acceptably small

If the new vertex does not cause an overlap with any existing triangle and is not close to any existing vertices then we accept the proposed vertex and resulting triangle Then we go on to another boundary edge.

If the data lies in a higher dimensional space but the local covariances have only two principal eigenvectors the algorithm is essentially unchanged although the computational burden is increased. If there are $k > 2$ principal eigenvectors the instead of constructing triangles we construct k simplices in a similar fashion. There is no reason to presume that the number of principal eigenvectors is constant and we are working on extensions of the basic algorithm to handle this.

III. Example

We implement the above on an example. We generate $5,000$ data points more or less uniformly distributed on a torus in $|Real^3$ with horizontal radius 4 and vertical radius 1, see Figure 1. Figure 2 shows the results of the algorithm after 100 triangles have been constructed. The color coding is dictated by the height of the triangle. Upper triangle are shown in shades of purple while lower triangles are in shades of blue. The Figures 3-6 show the results of the algorithm after 200, 300, 400 and 500 triangles have been constructed.

Next we corrupted each component of the data points by a Gaussian random variable of standard deviation 0.1 and ran the algorithm again. Figures 7-12 show the results. As you can see the algorithm is robust to a reasonable amount of noise.

IV. References

[1] J. B. Tenenbaum, V. de Silva,and J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, *Science*, vol. 290, 2000, pp. 2319-2323.

[2] Y. Pan, S. S. Ge, A.Al Mamun, Weighted locally linear embedding for dimension reduction, *Pattern Recognition*, vol. 42, 2009, pp. 798–811.

[3] S. T. Roweis and L. K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, *Science*, vol. 290, 2000, pp. 2323-2326.

[4] L. K. Saul and S. T. Roweis, Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds, *Journal of Machine Learning Research*, vol. 4, 2003, pp. 119-155.

[5] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig Non-linear PCA: a missing data approach, *Bioinformatics*, vol. 21, 2005, pp. 3887-3895.

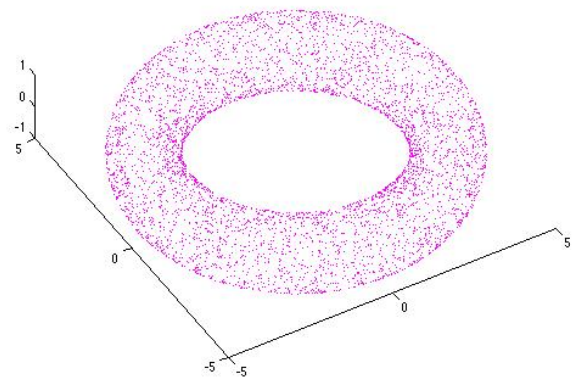

Fig. 1. Five Thousand Random Data Points on the Torus

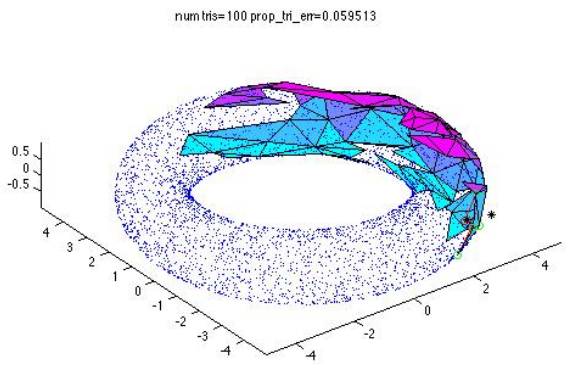

Fig. 2. One Hundred Triangles on the Torus

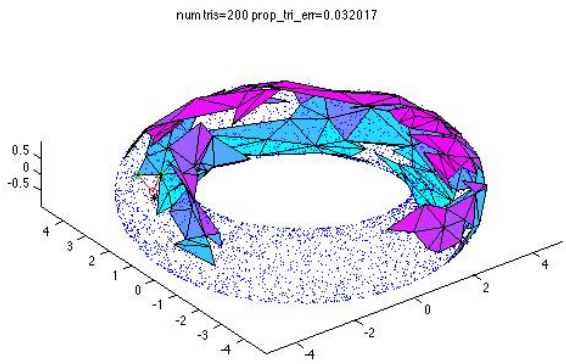

Fig. 3. Two Hundred Triangles on the Torus

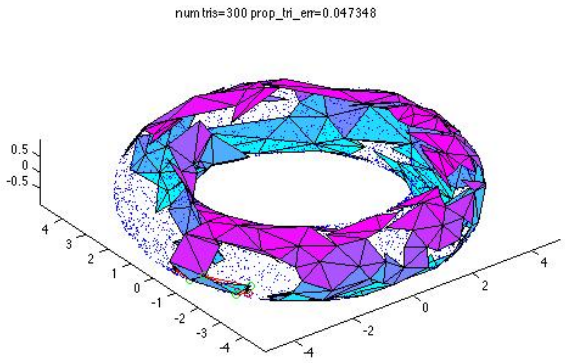

Fig. 4. Three Hundred Triangles on the Torus

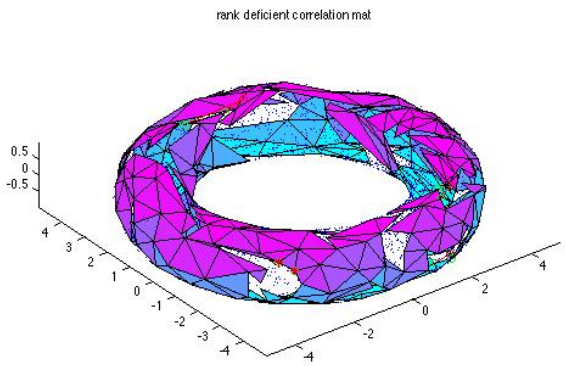

Fig. 5. Four Hundred Triangles on the Torus

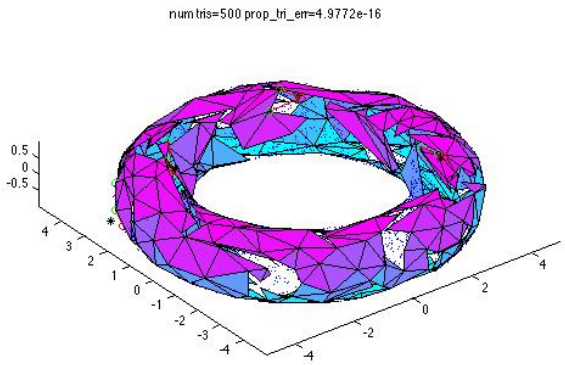

Fig. 6. Five Hundred Triangles on the Torus

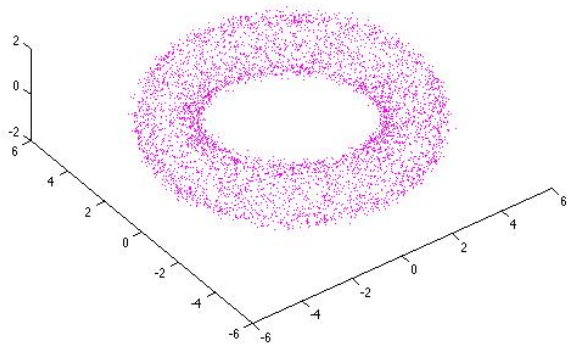

Fig. 7. Five Thousand Noisy Data Points on the Torus

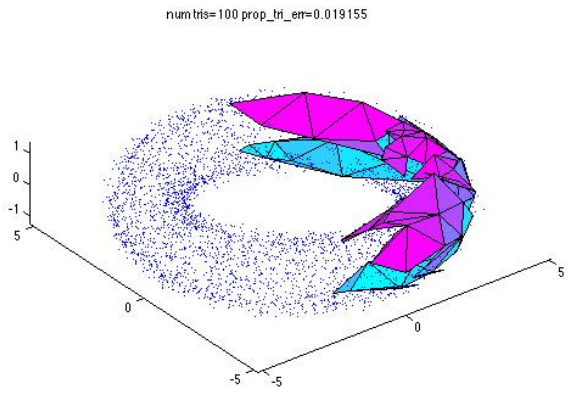

Fig. 8. One Hundred Noisy Triangles on the Torus

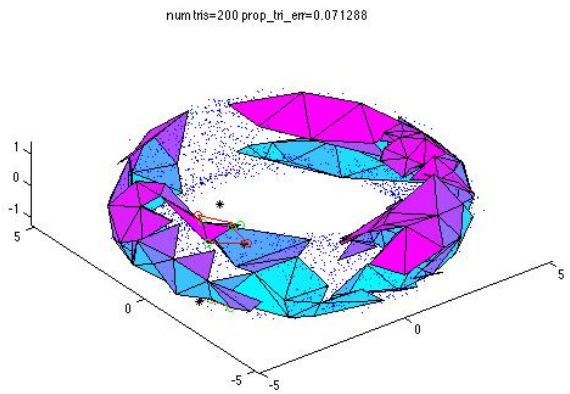

Fig. 9. Two Hundred Noisy Triangles on the Torus

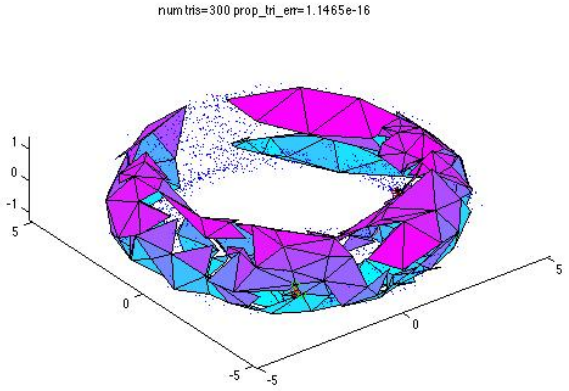

Fig. 10. Three Hundred Noisy Triangles on the Torus

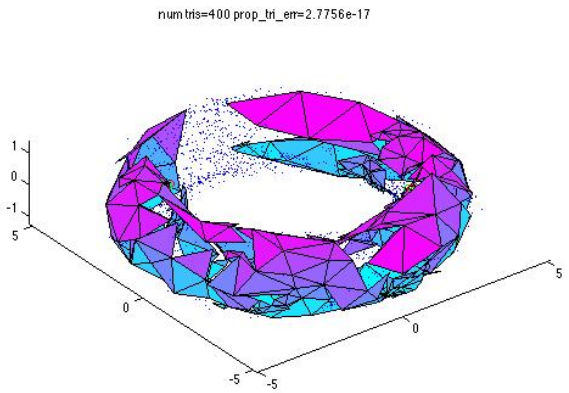

Fig. 11. Four Hundred Noisy Triangles on the Torus

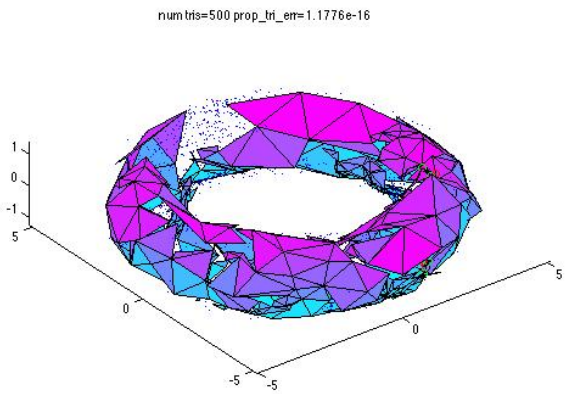

Fig. 12. Five Hundred Noisy Triangles on the Torus