Lecture 12. Nonlinear dimension reduction

Prof. Alan Yuille

Spring 2014

Outline

1. Introduction
2. Isomap
3. Locally Linear Embedding

1 Introduction

Nonlinear dimensionality reduction is necessary when the data has some non-linear structure which linear methods cannot "unfold".

Spectral Methods. The basic idea is to assume that the data lies on a manifold/surface in D-dimensional space, see figure (1) Perform multi-dimensional scaling, or other dimension reduction method, using distances calculated on the manifold, see figure (2).

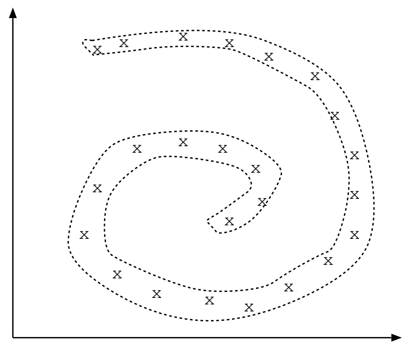

Figure 1: We assume that the data lies on a manifold. This is a surface which is locally flat.

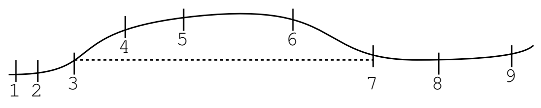

Figure 2: The distance is calculated on the manifold. I.e. the distance from x_3 to x_7 is the sum of the distances from x_3 to x_4, from x_4 to x_5, from x_5 to x_6, from x_6 to x_7. It is *not* the direct distance between x_3 and x_7.

A common strategy is: (i) define a sparse graph using kNN (nearest neighbors), (ii) derive a matrix from graph weights, (iii) derive the embedding from the graph weights.

In the following sections we describe two exapmles of nonlinear dimensional reduction methods.

2 Isomap

Isomap (Tenenbaum et al, 2000) stands for isometric feature mapping. It provides a simple method for estimating the intrinsic geometry of a data manifold and embeds the data in a lower-dimensional space.

We have a set of datapoints $\mathcal{X} = \{\vec{x}_i : i = 1, ..., N\}$.

Step 1. Construct an adjacency graph $(\mathcal{V}, \mathcal{E})$, where the datapoints are the vertices $\mathcal{V}$ (hence $|\mathcal{V}| = N$). The edges $\mathcal{E}$ connect each node to its k nearest neighbors, see figure (2).

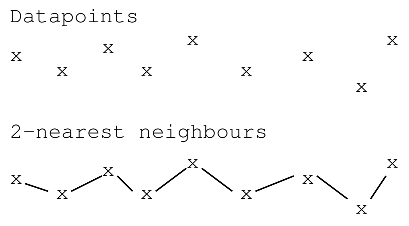

Figure 3: K-nearest neighbors with k=2.

Step 2. Estimate geodesics, the shortest distance between two nodes x_i, x_j of the graph on the manifold, see figure (2). Use dynamic programming to calculate the shortest path from A to B, see figure (2). This gives a geodesic distance between x_i and x_j measured on the manifold.

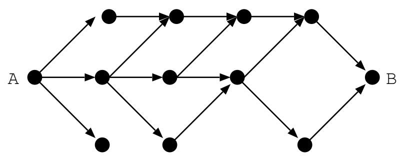

Figure 4: Dynamic programming: finding the shortest path.

Step 3. Apply Multi-dimensional scaling (MDS) to the Δ_{iJ}. MDS is described at the end of this lecture.

The complexity of this algorithm is as follows. (1) k nearest neighbor – complexity $O(n^2 D)$, (2) Shortest path by dynamic programming – complexity $O(n^2 k + n^2 \log n)$, (3) MDS – complexity $O(n^2 d)$. The overall complexity is high. Impractical for large numbers of datapoints. Possible solution – Landmark Isomap: identify a subset of datapoints as "landmarks", do Isomap on the landmarks, interpolate for other datapoints.

This is a simple and intuitive algorithm. There is a problem is the algorithm finds short-cuts, see figure (2). It only projects the datapoints and does not specify how to project new data.

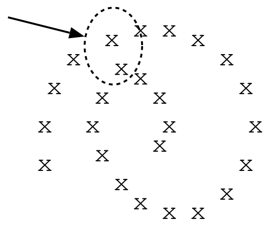

Figure 5: Short-cuts can cause a problems for Isomap. The short cut is a connection between two different spirals of the surface. It prevents the geodesic from lying on the surface and instead allow it to take short cutes.

The results are intuitive. For example, take two images of a face and interpolate in Isomap feature space. Isomap was the first of several methods which used similar strategies.

Theoretical guarantee. If the manifold is isometric to a convex subset of Euclidean space then Isomap recovers this subspace (up to rotation and translation). An isometry is a distance preserving map.

3 Locally Linear Embedding (LLE)

There are some advantages of LLE over Isomap, including faster optimization when implemented to take advantage of sparse matrix algorithms, and better results with many problems. However, LLE doesn't estimate the dimensionality, and offers no theoretical guarantees. It consists of the following steps.

Step 1. As for Isomap, construct adjacency graph using k-NN.

Step 2. Construct weights – characterize the local geometry by weights w_{ij} chosen to minimize:

$$\Phi(w) = \sum |\vec{x}_i - \sum_j w_{ij}\vec{x}_j|^2.$$

This is the error of reconstructing/predicting $\vec{x}_i$ from its neighbors. Here $w_{ii} = 0, \sum_j w_{ij} = 1$.

The solution $\hat{w} = \arg\min_w \Phi(w)$ has local invariance, see Figure (3). The optimal weights are invariant to rotation, translation, and scaling.

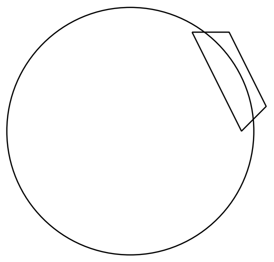

Figure 6: The geometry is characterized locally by $\hat{w}$. For example, if the data was lying on the surface of a sphere, it could be locally approximated by a plane.

Step 3. Linearization. This maps inputs to outputs $\vec{y}_i$ to minimize the reconstruction error:

$$\Psi(\vec{y}) = \sum_i |\vec{y}_i - \sum_j w_{ij}\vec{y}_j|^2$$

With the constraint $\sum_j \vec{y}_j = 0$. And impose $1/N \sum_i \vec{y}_i \vec{y}_i^T = I$ (the identity matrix).

Reformulate:

$$\Psi(\vec{y}) = \sum_{ij} \Psi_{ij} \vec{y}_i \cdot \vec{y}_j.$$

Where $\Psi = (I - W)^T(I - W)$. Use the bottom $d+1$ eigenvalues – where we are projecting onto d dimensions.

Both optimizations of $\Phi(w)$ and $\Psi(\vec{y})$ can be performed by linear algebra.

Properties. This has polynomial time optimization (no need to compute geodesics). It does not estimate the dimensions. It has no theoretical guarantees.