

Low-dimensional Representations

Often we want to find a representation of data from \mathbb{R}^D in some lower-dimensional space, \mathbb{R}^F , for $F \ll D$. For $F \in \{2, 3\}$, this is useful for visualization. For other F , it's useful if we believe that the data is noisy, or not ideal for our learning algorithm (eg., k NN).

There are two varieties of dimensionality reduction techniques: linear and non-linear. We will talk about one example of each.

Linear Dimensionality Reduction: PCA

Have data matrix in $X \in \mathbb{R}^{N \times D}$. Want to linearly *project* X into some $Y \in \mathbb{R}^{N \times F}$. We don't want to lose much "information."

Two ways of deriving PCA:

1. Project X onto basis vectors of highest variance
2. Imagine data was generated by an F -dimensional Gaussian and then noisified into D dimensions

(1) is standard, hence the name "principle component analysis" (a component is a basis vector).

First, center X so it has mean 0.

Now, what do we want? We want Y such that YY^\top is diagonal (so the basis is orthogonal) and so that $Y = ZX$. From this, we get:

$$\begin{aligned} YY^\top &= (ZX)(ZX)^\top \\ &= ZXX^\top Z^\top \\ &= Z(XX^\top)Z^\top \end{aligned}$$

Now, linear algebra (i.e., magic) tells us that if A is a symmetric matrix, then $A = EDE^\top$, where E is a matrix of eigenvectors of A and D is a diagonal matrix of eigenvalues.

Letting $A = XX^\top$, we select Z to be the matrix of eigenvectors of XX^\top and D be the eigenvalues, so:

$$\begin{aligned} Z(XX^\top)Z^\top &= Z(Z^\top DZ)Z^\top \\ &= (ZZ^\top)D(ZZ^\top) \\ &= (ZZ^{-1})D(ZZ^{-1}) \\ &= D \end{aligned}$$

using the fact that the inverse of an orthogonal matrix is its transpose.

All you really need to know is that you first center your data, find the eigenvectors corresponding to the top F eigenvalues, and then use these as the new basis for the data.

You can additionally show that PCA minimizes the reconstruction mean-squared error, within the constraint of being an orthogonal linear projection.

Nonlinear Dimensionality Reduction: LLE

Locally linear embedding is a “manifold learning” algorithm. A manifold is like a F dimensional space warped to fit into a $D > F$ dimensional space. Think about a partially folded piece of paper (swiss roll).

We want to “unfold” the manifold so that it lies in its true dimensionality, F . Of course, the problem is that (a) we only have data from the manifold and (b) the data is noisy.

LLE attempts to unfold the manifold by assuming local linearity.

The algorithm works by considering each data point independently, and only in the context of its k nearest neighbors. Then, we want to be able to *reconstruct* the original data point based only on its neighbors, using a linear function. We then use these linear functions to project the data into low dimensional space.

Algorithm:

1. For each data point x_n , calculate the set S_n of its k nearest neighbors (excluding itself).
2. For each n , compute a weight vector w_n that minimizes:

$$\left\| x_n - \sum_{m \in S_n} w_{nm} x_m \right\|^2$$

subject to $\sum_m w_{nm} = 1$ for all n .

3. For each n , compute an embedded y_n to minimize:

$$\sum_n \left\| y_n - \sum_{m \in S_n} w_{nm} y_m \right\|^2$$

The optimal weights (from (2)) are invariant to rotations, rescalings and translations of a data point and its neighbors.

The second step is a bunch of least squares problems, one for each data point. The third step can be solved using eigen techniques. (Details in the paper.)