CS731 Spring 2011 Advanced Artificial Intelligence

Dimensionality Reduction

Lecturer: Xiaojin Zhu *jerryzhu@cs.wisc.edu*

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^D$. It is convenient to assume that the points are centered $\sum_i \mathbf{x}_i = 0$ (one can always centered the data by subtracting the sample mean). We want to represent these points in some lower dimensional space $\mathbb{R}^d$ where typically $d \ll D$.

1 Principal Component Analysis (PCA)

PCA can be justified in several ways.

1.1 The Variance Preservation View

Let's consider a projection onto a line going through the origin. Such a line can be specified by a vector $\mathbf{w} \in \mathbb{R}^D$. The projection of $\mathbf{x}$ is

$$\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|}. \tag{1}$$

For simplicity, let us consider $\mathbf{w}$ with unit length. The variance of the projected dataset is

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i)^2 = \mathbf{w}^\top S \mathbf{w}, \tag{2}$$

where

$$S = \frac{1}{n}\sum_i \mathbf{x}_i \mathbf{x}_i^\top \tag{3}$$

is the sample covariance matrix since we assume the dataset is centered. The goal of PCA (in this 1D case) is to find the $\mathbf{w}$ that maximizes the variance, in the hope that it maximally preserves the distinction among points. This leads to the following optimization problem

$$\max_{\mathbf{w}} \quad \mathbf{w}^\top S \mathbf{w} \tag{4}$$

$$\text{s.t.} \quad \|\mathbf{w}\| = 1. \tag{5}$$

Let's solve it by forming the Lagrangian

$$\mathbf{w}^\top S \mathbf{w} + \lambda(1 - \mathbf{w}^\top \mathbf{w}). \tag{6}$$

The gradient w.r.t. $\mathbf{w}$ is

$$\nabla = 2S\mathbf{w} - 2\lambda\mathbf{w}. \tag{7}$$

Setting to zero, we find that

$$S\mathbf{w} = \lambda\mathbf{w}, \tag{8}$$

i.e., the desired direction $\mathbf{w}$ is an eigenvector of S! But which one? Recall the projected variance is

$$\mathbf{w}^\top S \mathbf{w} = \mathbf{w}^\top \lambda \mathbf{w} = \lambda, \tag{9}$$

we see that we want λ to be the largest eigenvalue of S and $\mathbf{w}$ the corresponding eigenvector. In other words, let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of S in non-increasing order, and $\phi_1, \ldots, \phi_n$ be the corresponding eigenvectors. Then ϕ_1 is the maximum variance preserving direction, and the resulting variance is simply λ_1. This is PCA with $d = 1$: a D-dimensional point $\mathbf{x}$ is projected to a scalar $\phi_1^\top \mathbf{x}$. Note that when S's top eigenvalue has multiplicity larger than one, i.e., $\lambda_1 = \lambda_2$, then PCA is not unique: any unit vector in $\text{span}(\phi_1, \phi_2)$ can be the PCA direction.

If we want $d > 1$, it can be shown that we want to project $\mathbf{x}$ onto the first d eigenvectors

$$\mathbf{x} \to (\phi_1^\top \mathbf{x}, \ldots, \phi_d^\top \mathbf{x})^\top. \tag{10}$$

Recall that one can view $\phi_1, \ldots, \phi_D$ as the D major-to-minor axes of an ellipsoid represented by the sample covariance matrix (NB this does not assume that the underlying distribution is Gaussian). Clearly, if $d = D$ then $\phi_1 \ldots \phi_D$ is a basis for $\mathbb{R}^D$, and this PCA projection amounts to a rotation of the coordinate system (align them with the eigenvectors) without any loss of information.

1.2 The Minimum Reconstruction Error View

Using *any* orthonormal basis $\mathbf{u}_1 \ldots \mathbf{u}_D$, a training point $\mathbf{x}_i$ (recall it has been centered) can be written as

$$\mathbf{x}_i = \sum_{j=1}^{D} \alpha_{ij} \mathbf{u}_j \tag{11}$$

where

$$\alpha_{ij} = \mathbf{u}_j^\top \mathbf{x}_i. \tag{12}$$

Consider the d-term approximation to $\mathbf{x}_i$:

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{d} \alpha_{ij} \mathbf{u}_j. \tag{13}$$

We want the approximation error to be small for all training points:

$$\frac{1}{n} \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 = \frac{1}{n} \sum_{i=1}^{n} \| \sum_{j=d+1}^{D} \alpha_{ij} \mathbf{u}_j \|^2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=d+1}^{D} \alpha_{ij}^2 \tag{14}$$

$$= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=d+1}^{D} \mathbf{u}_j^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{u}_j = \sum_{j=d+1}^{D} \mathbf{u}_j^\top S \mathbf{u}_j. \tag{15}$$

If $d = D-1$, i.e., we need to remove a single dimension, it is easy to see that $\mathbf{u}_D = \phi_D$ because $\phi_D^\top S \phi_D = \lambda_D$ is the smallest among all unit vectors. Similarly, the other dimensions to remove are subsequently the eigenvectors corresponding to the least eigenvalues.

Finally, it should be pointed out that PCA is an *unsupervised* technique. It could go terribly wrong as a preprocessing step for classification, e.g., when the decision boundary is orthogonal to the smallest eigenvector.

2 Classical Multidimensional Scaling (MDS)

MDS is not a dimensionality reduction method, but rather an *embedding* method. We have n items but we do not know their feature representation. Instead, we are given their pairwise squared distances

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{16}$$

Our goal is to *embed* these items in $\mathbb{R}^d$, i.e., finding a feature representation $\mathbf{x}_1, \ldots, \mathbf{x}_n$ such that it satisfies the given squared distances.

We introduce the $n \times n$ *centering matrix*

$$P = I_{n\times n} - \frac{1}{n}U \tag{17}$$

where U is the $n \times n$ all-1 matrix. This matrix is called the centering matrix because when it applies to a $n \times d$ matrix X where each row is a data point, the result

$$PX = [\mathbf{x}_i - \mu], \tag{18}$$

is to subtract μ the mean of X from each row. In addition,

$$P1 = 0 \tag{19}$$

where 1 here is the all-1 vector. Now consider

$$PXX^\top P = (PX)(PX)^\top \tag{20}$$

We see that the ij-th entry is $(\mathbf{x}_i - \mu)^\top(\mathbf{x}_j - \mu)$, i.e., the inner product between the centered i-th and j-th items. Furthermore, with this result

$$\begin{aligned} P[\|\mathbf{x}_i - \mathbf{x}_j\|^2]P &= P[\mathbf{x}_i^\top\mathbf{x}_i + \mathbf{x}_j^\top\mathbf{x}_j - 2\mathbf{x}_i^\top\mathbf{x}_j]P & (21)\\ &= P[\mathbf{x}_i^\top\mathbf{x}_i]P + P[\mathbf{x}_j^\top\mathbf{x}_j]P - 2P[\mathbf{x}_i^\top\mathbf{x}_j]P & (22)\\ &= -2P[\mathbf{x}_i^\top\mathbf{x}_j]P & (23)\\ &= -2[(\mathbf{x}_i - \mu)^\top(\mathbf{x}_j - \mu)] & (24)\\ & & (25)\end{aligned}$$

where we used the fact $P[a_i]_{n\times n}P = P[a_i]_{n\times 1}1_{1\times n}P = P[a_i]_{n\times 1}0 = 0$. This suggests that given a squared distance matrix $[\|\mathbf{x}_i - \mathbf{x}_j\|^2]$, the matrix

$$\bar{A} = -\frac{1}{2}P[\|\mathbf{x}_i - \mathbf{x}_j\|^2]P \tag{26}$$

gives us the centered inner product matrix $[(\mathbf{x}_i - \mu)^\top(\mathbf{x}_j - \mu)]$.

Theorem 1 *Consider the class of symmetric matrices $A \in S_n$ such that $A_{ij} \geq 0$ and $A_{ii} = 0$, $\forall i, j$. Then $\bar{A} = -\frac{1}{2}PAP$ is positive semi-definite if and only if A is a squared distance matrix. The minimal embedding dimension d is the rank of $\bar{A}$.*

Now that we have $\bar{A}$, we need to actually find an embedding. To this end, we perform eigen-decomposition

$$\bar{A} = \sum_{k=1}^{n} \lambda_k \phi_k \phi_k^\top. \tag{27}$$

where λ is sorted in decreasing order and the first d will be nonzero. Now consider the $n \times d$ matrix

$$Y = [\phi_1 | \ldots | \phi_d]\Lambda^{-1/2} \tag{28}$$

where $\Lambda^{-1/2}$ is the $d \times d$ diagonal matrix with the k-th diagonal element $\sqrt{\lambda_k}$. It is easy to see that

$$\bar{A} = YY^\top. \tag{29}$$

We take the rows of Y to be the embedding of the n items.

The minimum embedding dimension d is the dimension for which there is no distortion. One can ask for an even smaller dimensional embedding $d' < d$, where one simply uses the top d' eigenvector / eigenvalues in (28).

Classical MDS has been generalized to non-metric MDS, where the input is not squared distances but rather some general notion of dissimilarities. Ordinal MDS is an example of non-metric MDS where the input is the *ranking* of pairwise distances, but not the distances themselves.

3 Isomap

PCA and MDS both assume that the data lives in a Euclidean (sub)space. Isomap assumes that the data lives on a low dimensional manifold embedded in a Euclidean space. In this case, we are given the feature representation $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^D$. But instead of computing the Euclidean distances

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{30}$$

we should rather use the geodesic distance along the manifold between $\mathbf{x}_i$ and $\mathbf{x}_j$. Isomap approximates the geodesic distance by the shortest path length on a graph (e.g., kNN graph with small k) constructed on the data points. It then uses MDS on these (squared) shortest path lengths to find a low dimensional embedding.

4 Locally Linear Embedding (LLE)

LLE is an alternative to Isomap. It similarly starts with a kNN graph. Let $N(i)$ be the set of k nearest neighbors of $\mathbf{x}_i$. LLE works in two steps:

1. For each $\mathbf{x}_i \in \mathbb{R}^D$, fit itself by a linear combination of its neighbors. Each neighbor will have a weight in this fit.
2. Find new points $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^d$ such that each $\mathbf{y}_i$ is the same fit by its neighbors, using the same neighborhood and weights.

The fit in the first step is

$$\|\mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j\|^2 \tag{31}$$

where we introduced weights w_{ij}. This fit should be invariant by global translation $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta,\ \Delta \in R^D$, which induces the constraint

$$\sum_{j \in N(i)} w_{ij} = 1. \tag{32}$$

To find w, we solve

$$\min_{w} \quad \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j\|^2 \tag{33}$$

$$\text{s.t.} \quad \sum_{j \in N(i)} w_{ij} = 1,\ \forall i. \tag{34}$$

This problem has a closed form solution.

With the w_{ij}'s at hand, we now find $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^d$ to minimize a similar fit error:

$$\min_{\mathbf{y}} \quad \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{y}_i - \sum_{j \in N(i)} w_{ij} \mathbf{y}_j\|^2 \tag{35}$$

To avoid the trivial all-0 solution, and to make the $\mathbf{y}$'s truly span a d-dimensional space, we constrain the matrix $Y_{n \times d}$ whose rows are the $\mathbf{y}$'s to have rank d. This can be achieved by a constraint $Y^\top P Y = Z$ for any symmetric $P_{n \times n}$ and $Z_{d \times d}$ of full rank. This is because

$$d = \text{rank}(Z) = \text{rank}(Y^\top P Y) \leq \min(\text{rank}(Y^\top), \text{rank}(P), \text{rank}(Y)) = \min(\text{rank}(Y), n) = \text{rank}(Y) \leq d. \tag{36}$$

Note we required $n \geq d$, which is realistic. In particular, we require

$$\frac{1}{n} Y^\top Y = I_{d \times d}. \tag{37}$$

The solution can be shown to be the 2nd to the $(d+1)$-th unit eigenvectors of $(I - W)^\top (I - W)$.

5 Laplacian Eigenmaps

We assume a graph $W_{n\times n}$ is constructed on $\mathbf{x}_1 \dots \mathbf{x}_n \in \mathbb{R}^D$. This could be a kNN graph with $w_{ij} = 1$ if $\mathbf{x}_i, \mathbf{x}_j$ are neighbors, or a weighted fully connected graph with $w_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2\right)$. Note the weights have a very different meaning than those in LLE. The intuition is that we want to find $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^d$ such that if w_{ij} is large, $\mathbf{y}_i, \mathbf{y}_j$ should be similar to each other (because $\mathbf{x}_i, \mathbf{x}_j$ was similar, that was how we got a large weight w_{ij}). This can be formulated as

$$\min_{\mathbf{y}} \sum_{ij} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2. \tag{38}$$

Let $Y_{n\times d}$ be the matrix whose rows are the $\mathbf{y}$'s, and $D_{n\times n}$ is the diagonal degree matrix with

$$D_{ii} = \sum_{j=1}^{n} w_{ij}. \tag{39}$$

Our objective can be written as

$$2\text{trace}(Y^\top L Y), \tag{40}$$

where

$$L = D - W \tag{41}$$

is known as the *Laplacian matrix*, hence the name. Similar to LLE, to avoid trivial solutions, we require

$$Y^\top D Y = I. \tag{42}$$

This results in the generalized eigenvalue problem

$$L\mathbf{y} = \lambda D\mathbf{y}. \tag{43}$$

The optimal Y is constructed with columns being the eigenvectors of the above generalized eigenvalue problem, with the smallest eigenvalues (but excluding the smallest one which is zero, corresponding to a constant eigenvector).