MSA220/MVE440 Statistical Learning for Big Data

Lecture 12

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Data representations

With big data we often need to find efficient data representations of a smaller dimension for both visualization and computation.

- Global/Linear: SVD, PCA, NMF, MDS
- Local/Nonlinear: isomap, kernelPCA, LLE, tSNE

Revisiting PCA

- Data matrix X of dimension $n \times p$
- Assume from now that rows are centered
- SVD $X = UDV'$, PCA $X'X = VD^2V'$. Principal components UD obtained by projecting X onto V: $XV = UD$.

Revisiting PCA

- Key: want to maximize variance along orthogonal projections.
- $max_v Var(vX)$ subject to $v'v = 1$
- Lagrangian formulation $max_v v'X'Xv - \lambda(v'v - 1)$
- Take derivatives: $X'Xv - \lambda v = 0$ or equivalently $Sv = \lambda v$ - the eigenproblem

Revisiting PCA

- What if we had instead focused on XX'?
- Let's apply PCA in this system: $XX' = WLW'$: equivalently $XX'W = WL$
- Multiple by X' from the left: $(X'X)(X'W) = (X'W)L$.
- This means $X'W$ is the eigenvectors of $X'X$
- Almost, since not orthonormal: $(X'W)'(X'W) = W'(XX')W = L$ not I
- Fix by renormalizing so: $V = (X'W)L^{-1/2}$ is the eigenvectors of $X'X$ obtained from XX'

Revisiting PCA

- Why do we care?
- Alternative approach to PCA from *object distances* rather than *feature correlations*
- $X'X \simeq Cov(X)$ is the p by p structure between features p.
- $XX' \simeq distance(X)$ is the n by n distance matrix between objects

Gram matrix and the distance matrix

- Let's look at the distance matrix $d_{ij} = dist(x_i, x_j)$
- $d_{ij} = ||x_i - x_j||^2 = (x_i - x_j)'(x_i - x_j) = x_i'x_i - 2x_i'x_j + x_j'x_j$
- Now

$$XX' = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \ldots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \ldots & & & \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{pmatrix} =$$

$$\begin{pmatrix} \sum_{j=1}^{p} x_{1j}^2 & \sum_{j=1}^{p} x_{1j}x_{2j} & \cdots & \sum_{j=1}^{p} x_{1j}x_{nj} \\ \sum_{j=1}^{p} x_{1j}x_{2j} & \sum_{j=1}^{p} x_{2j}^2 & \cdots & . \\ \ldots & & & \\ . & \ldots & . & \sum_{j=1}^{p} x_{nj}^2 \end{pmatrix}$$

Gram matrix and the distance matrix

- $d_{ij} = ||x_i - x_j||^2 = (x_i - x_j)'(x_i - x_j) = x_i'x_i - 2x_i'x_j + x_j'x_j$

$$XX' = \begin{pmatrix} \sum_{j=1}^{p} x_{1j}^2 & \sum_{j=1}^{p} x_{1j}x_{2j} & \cdots & \sum_{j=1}^{p} x_{1j}x_{nj} \\ \sum_{j=1}^{p} x_{1j}x_{2j} & \sum_{j=1}^{p} x_{2j}^2 & \cdots & . \\ \cdots & & & \\ . & \cdots & . & \sum_{j=1}^{p} x_{nj}^2 \end{pmatrix} =$$

$$\begin{pmatrix} x_1'x_1 & x_1'x_2 & \cdots & x_1'x_n \\ x_1'x_2 & x_2'x_2 & \cdots & . \\ \cdots & & & \\ . & \cdots & . & x_n'x_n \end{pmatrix}$$

- This means we can write the distance matrix $D = (d_{ij})$ as

$$D = 1diag(XX') - 2XX' + diag(XX')1'$$

where 1 is just a column of ones.

Gram matrix and the distance matrix

- The matrix $G = XX'$ is called the Gram matrix.
- The elements of G are the dot-products of the observations $x_i'x_j = < x_i, x_j >$
- From previous slide we see that the pairwise distances are just a function of the dot-products

$$D = 1diag(XX') - 2XX' + diag(XX')1'$$

Gram matrix and the distance matrix

- The matrix $G = XX'$ is called the Gram matrix.
- $$D = 1diag(XX') - 2XX' + diag(XX')1'$$
- Equivalently, define $H = I - \frac{1}{n}11'$ (also called the centering matrix), then
$$G = -\frac{1}{2}HD^2H$$

Revisiting Multi-dimensional scaling

- MDS: only use the pairwise distances
- MDS: not restricted to 2-dim space

Multi-dimensional scaling

- We compute all the pairwise distances between objects i and j: d_{ij}
- We can be clever about using appropriate distances here depending on the variable types (`daisy` package in R)
- We want to find observations z_i in a low-dimensional space such that

$$\sum_{i \neq i'} (d_{ii'} - ||z_i - z_{i'}||)^2$$

  is small.
- We can scale the mapping distance by $d_{ii'}$ which preserved small distances better
- We can also use rank-based mapping (called non-metric scaling) - depending if subsets of data are very spread out.

Multi-dimensional scaling

- We compute all the pairwise distances between objects i and j: d_{ij}
- We want to find observations z_i in a low-dimensional space such that

$$\sum_{i \neq i'} (d_{ii'} - ||z_i - z_{i'}||)^2$$

  is small.
- How? Spectral decomposition of centered $d:_{ii'}$ and use leading eigenvectors.
- With the alternative representation of PCA through the Gram matrix and thereby the distance matrix we see that MDS is in fact equivalent to PCA if the distance used is euclidean!!!

Multi-dimensional scaling

- Metric MDS: we want to minimize $min_z \sum_{ij}(d^x_{ij} - d^z_{ij})^2$
- We just saw the equivalence between the gram matrix and the distance matrix: minimizing D - max ”correlation” or min dotproduct.
- So MDS is equivalent to $min_z \sum_{ij}(x'_i x_j - z'_i z_j)^2$
- Write in matrix form as a Trace of the full matrices

$$min_Z Tr(XX' - ZZ')^2$$

- Use spectral decomposition of each

$$min_Z Tr(WLW' - QTQ')^2 = Tr(L - W'QTQ'W)^2 =$$

$$= Tr(L - RTR')^2 = Tr(L^2 - RTR'RTR' - 2LRTR')$$

Multi-dimensional scaling

- $$min_Z Tr(WLW' - QTQ')^2 = Tr(L - W'QTQ'W)^2 =$$
  $$= Tr(L - RTR')^2 = Tr(L^2 - RTR'RTR' - 2LRTR')$$
- For fixed T: minimize wrt R - solution $R = I$ and plug-in
  $$min_T Tr(L^2 - T^2 - 2LTR) = Tr(L^2 - T^2)$$
- Make small by matching the leading eigenvalues of L and T and since $I = R = W'Q$ this implies $Q = W$
- So MDS - leading eigenvectors of the Gram matrix.

Multi-dimensional scaling

- We compute all the pairwise distances between objects i and j: d_{ij}
- We want to find observations z_i in a low-dimensional space such that

$$\sum_{i \neq i'} (d_{ii'} - ||z_i - z_{i'}||)^2$$

  is small.
- The point being - with MDS we can exploit this and be more flexible with the distance used!
- Define $H = I - \frac{1}{n} 11'$ (also called the centering matrix), then we can obtain a gram matrix from a distance matrix through

$$G = -\frac{1}{2} H D^2 H$$

- and then solve the eigen problem

Multi-dimensional scaling

- PCA $=$ eigenvectors of $X'X$ (covariance of X) - scales poorly with dimensionality
- MDS $=$ eigenvectors of XX' (gram matrix, related to pairwise distances) - scales poorly with sample size
- Key is using other distance metrics in MDS for flexibility.
- In non-metric MDS you are even more adventurous - using monotone transformations of distances, qualitative distances, ranks - usually then solved by iterative procedures.

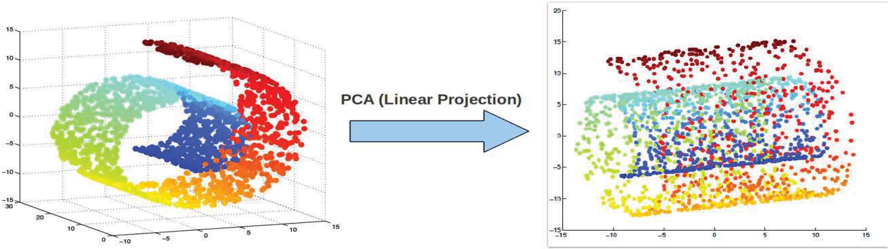
KERNELPCA

- PCA $=$ eigenvectors of $X'X$ (covariance of X) - scales poorly with dimensionality
- or equivalently eigenvectors of XX', obtain eigenvectors as $V = (X'W)L^{-1/2}$ and PCs as XV.
- Limitations of PCA: Global method, assuming X can be well represented by a linear projection/approximations - covariance is a linear association measure
- What if we could look at nonlinear dependencies? How?
- Transform data into a $M > p$ dimensional space: $\phi(x)$ and compute covariance in this space

  $$S^{\phi} = \phi(X)'\phi(X)$$

  (M by M matrix instead of p by p).
- Compute eigenvectors V: $S^{\phi}V = LV$ and project onto leading components here.

kernelPCA



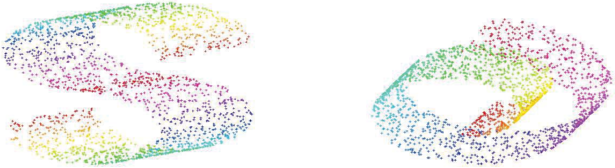
PCA (Linear Projection)

kernelPCA

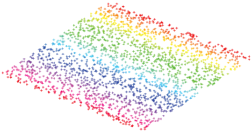
- What do we want the nonlinear transformation ϕ to do?
- Preserve local information: e.g. locally linear relationships or pairwise distance information

kernelPCA

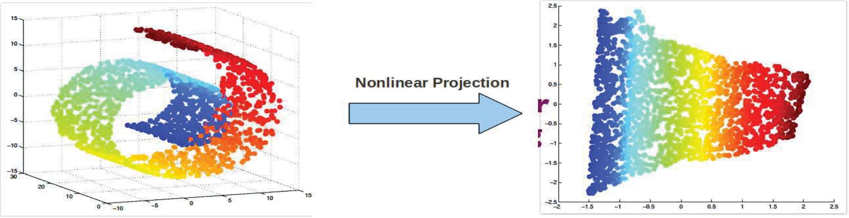
- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**



- Goal: Recover the low-dimensional surface



kernelPCA



Figures from slides by P. Rai

kernelPCA

- What do we want the nonlinear transformation ϕ to do?
- Preserve local information: e.g. locally linear relationships or pairwise distance information
- ”Unrolling” or ”unwrapping” the low dimensional structure embedded in a high dimensional space

kernelPCA

- We would rather not have to compute the data transformation for two reasons
- Increasing dimensionality - bad
- Having to explicitly define a transformation that works
- Idea: the kernel-trick. ”Easier” to define a nonlinear or local distance and we know from before there is an implicit relationship between the PCA decomposition based on structure and distance.

kernelPCA

- We have $S^{\phi} = \frac{1}{n}\sum_i \phi(x_i)\phi(x_i)'$ as the M by M covariance matrix. (assume centered after transformation)
- PCA: $S^{\phi} v_k = \lambda_k v_k$ for components $k = 1, \cdots, M$
- Plug in: $\frac{1}{n}\sum_i \phi(x_i)(\phi(x_i)' v_k) = \lambda_k v_k$
- Eigenvectors are of the format $v_k = \sum_i a_{ki}\phi(x_i)$ and so $\frac{1}{n}\sum_i \phi(x_i)(\phi(x_i)' \sum_j a_{kj}\phi(x_j) = \lambda_k \sum_j a_{kj}\phi(x_j)$
- Multiply this by $\phi(x_l)'$ on both sides $\frac{1}{n}\sum_i phi(x_l)'\phi(x_i)(\phi(x_i)' \sum_j a_{kj}\phi(x_j) = \lambda_k \sum_j a_{kj}\phi(x_l)'\phi(x_j)$

kernelPCA

- Multiply this by $\phi(x_l)'$ on both sides
  $\frac{1}{n}\sum_i phi(x_l)'\phi(x_i)(\phi(x_i)'\sum_j a_{kj}\phi(x_j) = \lambda_k \sum_j a_{kj}\phi(x_l)'\phi(x_j)$
- The dot products are scalars, define as $\phi(x_i)'\phi(x_j) = k(x_i, x_j)$
- and so we have
  $\frac{1}{n}\sum_i k(x_i, x_l)\sum_j a_{kj}k(x_i, x_j) = \lambda_k \sum_j a_{kj}k(x_i, x_l)$
- Define the kernel matrix K comprising all the dot products (see back at PCA via Gram) which captures pairwise distance/similariy in ϕ space

$$K^2 a_k = \lambda_k N K a_k \rightarrow K a_k = \lambda_k a_k$$

- We need to normalize the vs:
  $v_k' v_k = 1 = \sum_{i,j} a_{ki} a_{kj}\phi(x_i)'\phi(x_j) \rightarrow a_k' K a_k = 1$
- A projection onto the k-th PCs is thus

$$\phi(x)' v_k = \sum_i a_{ki} K(x, x_i)$$

kernelPCA

- The kernel matrix has to be centered prior to this
- Use the centering matrix from above
- Popular kernels: gaussian, polynomial

T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING

- tSNE is a local extension of MDS. (Paper can be found here).
- Here we use a kernel based distance between observations i and j and interpret this as a probability

  $$p_{j|i} = Gaussian - pdf(d_{ij}, \sigma_i^2) / \sum_{k \neq i} Gaussian - pdf(d_{ik}, \sigma_i^2)$$

  where σ_i is the bandwidth of the kernel around reference point i. You create a symmetric distance by taking the average of the two conditional distributions
- Even more simple if you use the same bandwidth everywhere

  $$p_{ij} = Gaussian - pdf(d_{ij}, \sigma^2) / \sum_{k, l \neq k} Gaussian - pdf(d_{kl}, \sigma^2)$$

- We now try to construct a d-dimensional space y that mimics these densities where we define the pdf in this space as

tSNE

- How do we measure distance in the y-space? Natural thing would be to use gaussian densities there too (called SNE)
- In the SNE, the authors observed that the y-space got "crowded" in that slightly similar observations were forced to be very similar in the low-dimensional space
- To remedy this, tSNE uses a more long-tailed distribution to describe the densities in y-space (Cauchy distribution)

  $$q_{ij} = \frac{(1 + d(y_i, y_j))^{-1}}{\sum_{k \neq i}(1 + d(y_i, y_k))^{-1}}$$

  where d is the squared euclidean distance
- We match p and q by minimizing the Kullback-Leibler distance $\sum_{i \neq j} p_{ij} \log(\frac{p_{ij}}{q_{ij}})$
- How? Gradient descent.
- So it's related to MDS, but with a different treatment of distances and a different cost function.

Isomap

- Isomap is a local version of MDS - we work with a matrix of distances between observations
- Use distances based on a shortest path in a graph connecting observations
- The graph is produced by connecting only objects that are within a certain euclidean distance of eachother, or is within a set of k nearest neighbors.
- This can capture quite local behaviour - nonlinear transformation of data
- Spectral decomposition of this matrix

Local linear embedding

- LLE - "fix" the problem with global PCA by only approximating each X by a linear combination of nearest neighbors!
- Find the L nearest neighbors of each observation
- Assume x_i can be explained by a weighted linear combination of only the neighbors

$$x_i = \sum_{j \in N_i} w_{ij} x_j, min_W \sum_i ||x_i - \sum_{j \in N_i} W_{ij} x_j||^2$$

where we normalize the weights to add to 1

- Now consider a K-dim space ($K < L$) where the same weights could approximate local behaviour

$$Z = \arg\min_Z \sum_i ||z_i - \sum_{j \in N_i} W_{ij} z_j||^2$$

where we want $Z'Z = I$ and centered Z.

Local linear embedding

- Now consider a K-dim space ($K < L$) where the same weights could approximate local behaviour

  $$Z = \arg\min_{Z} \sum_{i} ||z_i - \sum_{j \in N_i} W_{ij} z_j||^2$$

  where we want $Z'Z = I$ and centered Z.
- Can rewrite the approximation error in Z-space as $\sum_{ij} M_{ij} Z_i' Z_j$ where $M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$
- The bottom eigenvectors of M solves the problem

Local linear embedding

- Assume x_i can be explained by a weighted linear combination of only the neighbors

$$x_i = \sum_{j \in N_i} w_{ij} x_j, min_W \sum_i ||x_i - \sum_{j \in N_i} W_{ij} x_j||^2$$

where we normalize the weights to add to 1

- Lagrangian

$$\sum_i (||x_i - \sum_{jinN_i} W_{ij} x_j||^2 + \lambda_i (sum_j W_{ij} - 1))$$

solve separately for each observation

Local linear embedding

- Now consider a K-dim space ($K < L$) where the same weights could approximate local behaviour

$$Z = \arg\min_{Z} \sum_i ||z_i - \sum_{j \in N_i} W_{ij} z_j||^2$$

where we want $Z'Z = I$ and centered Z.

$$\sum_i ||z_i - \sum_{j \in N_i} W_{ij} z_j||^2 - \sum_{ab} \lambda_{ab} (\sum_i z_{ia} z_{ib} - \delta ab)$$

which after some manipulation can be written as

$$(I - W)'(I - W)Z = Z\Lambda$$

an eigenvalue problem (details in the paper).

Global/Local

- Global: PCA, SVD, MDS, NMF
- Local: kPCA, isomap, tSNE, LLE
- many many more.... I have posted some review papers.
- Of course, be careful about applying to real data! "Overfitting", noisy data, big/small n, big/small p, mixed data types...