Machine Learning (CS 6140)
Homework 3 Solutions

Instructor: Ehsan Elhamifar
Due Date: November 27, 2017, 11:45am

1. PCA Objective Value. Show that for the optimal solution of fitting a d-dimensional subspace to data using PCA, the objective function value is equal to $\sum_{i \geq d+1} \sigma_i^2$. Here, σ_i denotes the i-th largest singular value of the mean subtracted data matrix, $\bar{\boldsymbol{Y}} = \begin{bmatrix} \boldsymbol{y}_1 - \bar{\boldsymbol{y}} & \cdots & \boldsymbol{y}_N - \bar{\boldsymbol{y}} \end{bmatrix}$, where $\bar{\boldsymbol{y}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i$.

Solution: Denote the mean subtracted data by $\bar{\boldsymbol{Y}}$. Take the SVD of it to be $\bar{\boldsymbol{Y}} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^T$, where $\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}_N$ and $\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{I}_D$. Denote the d-dimensional data by $\boldsymbol{X}$ and let $\boldsymbol{U}_d = \boldsymbol{U}(:, 1:d)$. From class, we have $\boldsymbol{X} = \boldsymbol{U}_d^\top \bar{\boldsymbol{Y}} = \boldsymbol{\Sigma}(1:d,:) \boldsymbol{V}^T$. The PCA objective function can be written as

$$\begin{aligned} ||\bar{\boldsymbol{Y}} - \boldsymbol{U}_d \boldsymbol{X}||_F^2 = ||\bar{\boldsymbol{Y}} - \boldsymbol{U}(1:d,:)\boldsymbol{\Sigma}(1:d,:)\boldsymbol{V}^T||_F^2 &= ||\sum_{i=1}^{D} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T - \sum_{i=1}^{d} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T||_F^2 \\ &= ||\sum_{i=d+1}^{D} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T||_F^2 \\ &= \sum_{i=d+1}^{D} \sigma_i^2 \end{aligned}$$

To get the last equality, we used the fact that $\boldsymbol{u}_i^\top \boldsymbol{u}_j = 0$ for every $i \neq j$ and

$$\begin{aligned} ||\sum_{i=d+1}^{D} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T||_F^2 &= \operatorname{tr}\left((\sum_{i=d+1}^{D} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T)^\top (\sum_{i=d+1}^{D} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T) \right) \\ &= \operatorname{tr}\left(\sum_{i=d+1}^{D} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^\top \right) = \operatorname{tr}\left(\sum_{i=d+1}^{D} \sigma_i^2 \boldsymbol{v}_i^\top \boldsymbol{v}_i \right) \\ &= \operatorname{tr}(\sum_{i=d+1}^{D} \sigma_i^2) = \sum_{i=d+1}^{D} \sigma_i^2. \end{aligned} \tag{1}$$

2. Locally Linear Embedding. As discussed in the class, the first step of LLE involves solving the optimization

$$\min_{\boldsymbol{w}_i} \frac{1}{2} \|\boldsymbol{Y}_i \boldsymbol{w}_i\|_2^2 \ \text{ s.t. } \ \boldsymbol{1}^\top \boldsymbol{w}_i = 1,$$

where $\boldsymbol{Y}_i = \begin{bmatrix} \cdots \boldsymbol{y}_j - \boldsymbol{y}_i \cdots \end{bmatrix}$ for all j's that are neighbors of point j and $\boldsymbol{1}$ is a vector of all 1's of appropriate dimension. a) Prove that the solution of this optimization is given by $\boldsymbol{w}_i =$

$\frac{(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}}{\mathbf{1}^\top(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}}$. b) Assume that we have N data points in the D-dimensional ambient space. We set the number of nearest neighbors in LLE to be K. We try to find a d-dimensional representation of the data. What is the computational cost of step 1 and step 2 of LLE, in terms of big O notation (for example, $O(D^p N^q)$)? Explain your answer in details.

Solution:

Part A: We form the Lagrangian function as

$$L(\boldsymbol{w}_i, \lambda) = \frac{1}{2}\|\boldsymbol{Y}_i \boldsymbol{w}_i\|\|_2^2 - \beta(\mathbf{1}^\top \boldsymbol{w}_i - 1).$$

We set the partial derivative of $L(w_i, \lambda)$ w.r.t. w_i and β to zero respectively.

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{w}_i} = \boldsymbol{Y}_i^\top \boldsymbol{Y}_i \boldsymbol{w}_i - \beta \mathbf{1} & = 0 \\ \dfrac{\partial L}{\partial \beta} = \mathbf{1}^\top \boldsymbol{w}_i - 1 & = 0 \end{cases}$$

$$\Rightarrow \begin{cases} \boldsymbol{w}_i = \beta(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1} & ① \\ \mathbf{1}^\top \boldsymbol{w}_i - 1 = 0 & ② \end{cases}$$

Plug $\boldsymbol{w}_i$ in ① into ②. We obtain

$$\Rightarrow \mathbf{1}^\top(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}\,\beta = 1$$

$$\Rightarrow \beta = [\mathbf{1}^\top(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}]^{-1}$$

Plug the above β back into ①. We obtain

$$\boldsymbol{w}_i = \beta(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1} = \frac{(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}}{\mathbf{1}^\top(\boldsymbol{Y}_i^\top \boldsymbol{Y}_i)^{-1}\mathbf{1}}.$$

Part B:
Step 1) Find $\{w_{ij}\}_{j\in N_i}$ for $i = 1, ..., N$:

1. Computing nearest neighbors scales (in the worst case) as $O(DN^2)$, or linearly in the input dimensionality, D, and quadratically in the number of data points, N.
2. Computing the reconstruction weights scales as $O(N(k^3 + k^2 D))$; this is the number of operations required to solve $\boldsymbol{w}_i$ for each data point.

Step 2) Given $\{w_{ij}\}_{j\in N_i}$, recover $\{x_i \in \mathcal{R}^d\}_{i=1}^N$:

Computing the bottom eigenvectors scales as $O(dN^2)$, linearly in the number of embedding dimensions, d, and quadratically in the number of data points, N.

The overall complexity of standard LLE is $O(DN^2 + k^3N + k^2DN)$.

N : number of training data points

D : input dimension

k : number of nearest neighbors

d : output dimension

3. Laplacian Matrix. The Laplacian matrix of a graph on N nodes is defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{W}$ is the matrix of weights of the graph, where W_{ij} is the weight between nodes i and j. The degree matrix $\boldsymbol{D}$ is also defined as a diagonal matrix whose i-th diagonal entry is the sum of the weights of the nodes connected to node i, i.e., $D_{ii} = \sum_{j=1}^{N} W_{ij}$. a) Let $\boldsymbol{x} = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix}^\top$. Compute a closed-form expression for $\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}$ that only depends on W_{ij}'s. b) Show that the Laplacian matrix is a positive semi-definite matrix. c) Is L an invertible matrix? Prove. d) How many zero singular values does $\boldsymbol{L}$ have?

Solution:

Part A:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} = \begin{bmatrix} D_{11} & -W_{12} & ... & -W_{1N} \\ ... & ... & ... & ... \\ -W_{N1} & -W_{N2} & ... & D_{NN} \end{bmatrix}$$

$$\begin{aligned}
\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} &= \boldsymbol{x}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x} \\
&= \sum_{i=1}^{N} D_{ii}x_i^2 - \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_ix_j \\
&= \frac{1}{2}(\sum_{i=1}^{N} D_{ii}x_i^2 - 2\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_ix_j + \sum_{j=1}^{N} D_{jj}x_j^2) \\
&= \frac{1}{2}(\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_i^2 - 2\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_ix_j + \sum_{j=1}^{N}\sum_{i=1}^{N} W_{ij}x_j^2) \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(x_i - x_j)^2
\end{aligned}$$

Part B: $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}(x_i - x_j)^2 \geq 0, \forall \boldsymbol{x} \neq \boldsymbol{0}$. Hence, $\boldsymbol{L}$ is a positive semi-definite matrix.

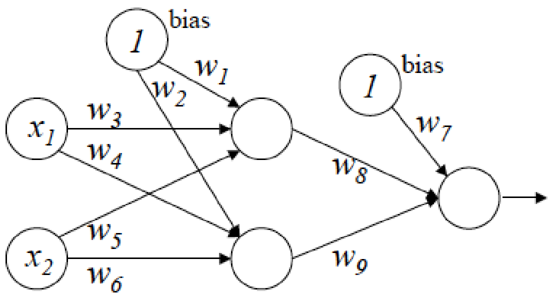
Part C: $\boldsymbol{L}$ is not an invertible matrix. Denote $\boldsymbol{1} = [1, ..., 1]^T$, where the dimension of $\boldsymbol{1}$ is $N \times 1$. We have

$$\boldsymbol{L}\boldsymbol{1} = \begin{bmatrix} D_{11} - \sum_{j=1}^{N} W_{1j} \\ ... \\ D_{NN} - \sum_{j=1}^{N} W_{Nj} \end{bmatrix} = \boldsymbol{0}$$

Thus, $\boldsymbol{L}$ has an eigenvalue of 0 and is not full rank, hence, it is not invertible.

Part D: The number of zero singular values equals to the number of connected components of the graph built using $\boldsymbol{W}$.

4. Neural Network. Consider a neural net for a binary classification which has one hidden layer as shown in the figure below. We use a linear activation function $a(z) = cz$ at hidden units and a sigmoid activation function $a(z) = 1/(1+e^{-z})$ at the output unit to learn the function for $P(y = 1|\boldsymbol{x}, \boldsymbol{w})$ where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2, \ldots, w_9)$. A) What is the output $P(y = 1|x, w)$ from the above neural net? Express it in terms of x_i, c and weights w_i. What is the final classification boundary? B) Draw a neural net with no hidden layer which is equivalent to the given neural net, and write weights $\tilde{\boldsymbol{w}}$ of this new neural net in terms of c and w_i. C) Is it true that any multi-layered neural net with linear activation functions at hidden layers can be represented as a neural net without any hidden layer? Explain your answer.



Solution:

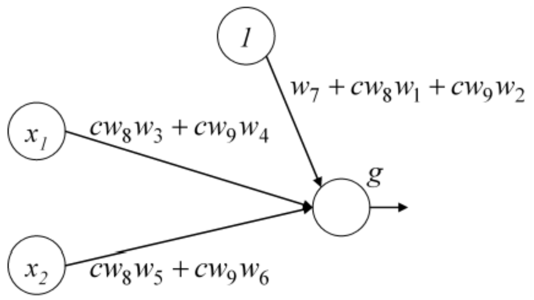
Part A:

$$g(w_7 + w_8 h(w_1 + w_3x_1 + w_5x_2) + w_9 h(w_2 + w_4x_1 + w_6x_2))$$
$$= \frac{1}{1 + \exp(-(w_7 + cw_8w_1 + cw_9w_2 + (cw_8w_3 + cw_9w_4)x_1 + (cw_8w_5 + cw_9w_6)x_2))}$$

The classification boundary is :

$$w_7 + cw_8w_1 + cw_9w_2 + (cw_8w_3 + cw_9w_4)x_1 + (cw_8w_5 + cw_9w_6)x_2 = 0$$

Part B:



Part C: Yes. If linear activation functions are used for all the hidden units, output from hidden units will be written as linear combination of input features. Since these intermediate output serves as input for the final output layer, we can always find an equivalent neural net which does not have any hidden layer as seen in the example above.

4. PCA Implementation. Implement the PCA and the Kernel PCA algorithm. The input to the PCA algorithm must be the input data matrix $Y \in \mathbb{R}^{D\times N}$, where D is the ambient dimension and N is the number of points, as well as the dimension of the low-dimensional representation, d. The output of the PCA must be the basis of the low-dimensional subspace $U \in \mathbb{R}^{D\times d}$, the mean of the subspace $\mu \in \mathbb{R}^D$ and the low-dimensional representations $X \in \mathbb{R}^{d\times N}$. For KPCA, the input must be the Kernel matrix $K \in \mathbb{R}^{N\times N}$, where N is the number of points, as well as the dimension of the low-dimensional representation, d. The output of KPCA must be the low-dimensional representations $X \in \mathbb{R}^{d\times N}$.

5. Kmeans Implementation. Implement the Kmeans algorithm. The input to the algorithm must be the data matrix $X \in \mathbb{R}^{D\times N}$, the desired number of clusters, k, as well as the number of repetitions of kmeans with different random initializations, r. The output of the algorithm must be the clustering of the data (indices of points in each group) for the best run of the kmeans, i.e., the run of kmeans that achieves the lowest cost function among all different initializations.

6. Spectral Clustering Implementation. Implement the Spectral Clustering algorithm. The input to the algorithm must be the similarity matrix $\boldsymbol{W} \in Re^{N\times N}$, where W_{ij} is the similarity between point i and j, and the desired number of clusters, k. The output of the algorithm must be the clustering of data (indices of points in each group).

7. Testing Algorithms on Data.

Part A. Take the dataset1, which consists of 200 points in 40-dimensional space ($\boldsymbol{Y}$ is 40×200 dimensional matrix).

i) Plot the data by using the first and second features (i.e., first and second rows) of $\boldsymbol{Y}$. Can you see any structure in data? Explain.

ii) Plot the data by using the second and third features (i.e., first and third rows) of $\boldsymbol{Y}$. Can you see any structure in data? Explain.

iii) Apply PCA to data and reduce the dimension of data to $d = 2$. Your output should be a 2×200 dimensional matrix. Plot the 2-dimensional representation of data using PCA. Can you see any structure in data? Explain.

iv) Apply Kmeans to the 40-dimensional data. Decide about K based on the PCA plot of the data. Show the results by plotting the 2-dimensional data and indicating data in each cluster by a different color. Explain why Kmeans is or is not successful in recovering the true clustering/grouping of the data.

v) Apply Kmeans to the 2-dimensional data. Decide about K based on the PCA plot of the data. Show the results by plotting the 2-dimensional data and indicating data in each cluster by a different color. Explain why Kmeans is or is not successful in recovering the true clustering/grouping of the data.

vi) Explain the similarities and differences of results between part iv and v? Looking at the plots, how much the results for applying Kmeans to 40-dimensional original data differ from the results of applying Kmeans to 2-dimensional representations of data? For general problems, is there any advantage of first applying PCA and then Kmeans to data?

Part B. Take the dataset2, which consists of 200 points in 40-dimensional space ($\boldsymbol{Y}$ is 40×200 dimensional matrix).

i) Plot the data by using the first and second features (i.e., first and second rows) of $\boldsymbol{Y}$. Can you see any structure in data? Explain.

ii) Plot the data by using the second and third features (i.e., first and third rows) of $\boldsymbol{Y}$. Can you see any structure in data? Explain.

iii) Apply PCA to data and reduce the dimension of data to $d = 2$. Your output should be a 2×200 dimensional matrix. Plot the 2-dimensional representation of data using PCA. Can you see any structure in data? Explain.

iv) Apply Kmeans to the 2-dimensional data obtained by PCA. Show the results by plotting the 2-dimensional data and indicating data in each cluster by a different color. Explain why Kmeans is or is not successful in recovering the true clustering/grouping of the data.

v) Repeat part iv by applying Kernel PCA to data to reduce the dimension of original data to 2. For KPCA, use Gaussian kernel ($k_{ij} = \boldsymbol{e}^{-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2/(2\sigma^2)}$) and try several values of σ (Hint: 2-dimensional PCA plots can help you to pick relatively good σ values). Show the results (for the best σ you could find) by plotting the 2-dimensional data and indicating data in each cluster by a different color. Explain why KPCA is or is not successful in recovering the true clustering/grouping of the data.

vi) Repeat part iv by applying Spectral to data to reduce the dimension of original data to 2. For spectral clustering, connect each point to its K nearest neighbors using Euclidean distance between points and for any two points i and j connected, set the weights to be $w_{ij} = \boldsymbol{e}^{-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2/(2\sigma^2)}$. Try several values of K and σ (Hint: 2-dimensional PCA plots can help you to pick relatively good K and σ values). Show the results (for the best K and σ you could find) by plotting the 2-dimensional data and indicating data in each cluster by a different color. Explain why Spectral Clustering is or is not successful in recovering the true clustering/grouping of the data.

Homework Submission Instructions:
– Submission of Written Part: You must submit your written report in the class BEFORE CLASS STARTS. The written part, must contain the plots and results of running your code on the provided datasets.
–Submission of Code: You must submit your Python code (.py file) via email, BEFORE CLASS STARTS. For submitting your code, please send an email to me and CC both TAs.
- The title of your email must be "CS6140: Code: HW3: Your First and Last Name".
- You must attach a single zip file to your email that contains all python codes and a readme file on how to run your files.
- The name of the zip file must be "HW3-Code: Your First and Last Name".