Homework 4 solution

Problem 1: PCA and Feature Selection

SVMs and PCA

Consider the UCI Sonar data set.

- Perform PCA on the training data to reduce the dimensionality of the data set (ignoring the class labels for the moment). What are the top six eigenvalues of the data covariance matrix?

Solution: 63.56349684 36.76148171 16.41583154 10.71620451 9.68540952 8.94757052

- For each $k \in \{1, 2, 3, 4, 5\}$, project the training data into the best k dimensional subspace (with respect to the Frobenius norm) and use the SVM with slack formulation to learn a classifier for each $c \in \{1, 10, 100, 1000\}$. Report the error of the learned classifier on the validation set for each k and c pair.

Solution:

k = 1, c = 1, error: 0.40384615384615385
k = 1, c = 10, error: 0.40384615384615385
k = 1, c = 100, error: 0.40384615384615385
k = 1, c = 1000, error: 0.40384615384615385
k = 2, c = 1, error: 0.40384615384615385
k = 2, c = 10, error: 0.40384615384615385
k = 2, c = 100, error: 0.40384615384615385
k = 2, c = 1000, error: 0.40384615384615385
k = 3, c = 1, error: 0.3846153846153846
k = 3, c = 10, error: 0.3846153846153846
k = 3, c = 100, error: 0.3846153846153846

k = 3, c = 1000, error: 0.3846153846153846
k = 4, c = 1, error: 0.40384615384615385
k = 4, c = 10, error: 0.46153846153846156
k = 4, c = 100, error: 0.46153846153846156
k = 4, c = 1000, error: 0.46153846153846156
k = 5, c = 1, error: 0.23076923076923073
k = 5, c = 10, error: 0.2692307692307693
k = 5, c = 100, error: 0.1923076923076923
k = 5, c = 1000, error: 0.1923076923076923

- What is the error of the best k/c pair on the test data? How does it compare to the best classifier (with the same possible c choices) without feature selection? Explain your observations.

Solution: for the best k/c pair (k = 5, c = 1000), error on test set is: 0.23076923076923073; and the error with the best classifier (without feature selection) is: 0.23076923076923073. This result shows that even the original data has 60 dimensional features, but only 5 dimension are necessary to obtain the same performance.

- If you had to pick a value of k before evaluating the performance on the validation set (e.g., if this was not a supervised learning problem), how might you pick it?

Solution: Since there is no clear drop of the eigenvalues (sorted from high to low), I may pick all features (no selection) if I can afford the calculation expense. However, if learning with all features becomes too expensive, I may select the first 10 or 20 (for no reason, because there is no clear separation among the magnitudes of the eigenvalues).

PCA for Feature Selection

Again, using the UCI Sonar data set, train a Gaussian naïve Bayes classifier on the training data. What is its accuracy on the test set?
Solution: accuracy on test set is: 0.6923076923076923

If we performed PCA directly on the training data as we did in the first part of this question, we would generate new features that are linear combinations of our original features. If instead, we wanted to find a subset of our current features that were good for classification, we could still use PCA, but we would need to be more clever about it. The primary idea in this approach is to select features from the data that are good at explaining as much of the variance as possible. To do this, we can use the results of PCA as a guide. Implement the following algorithm for a given k and s:

1. Compute the top k eigenvalues and eigenvectors of the covariance matrix corresponding to the data matrix omitting the labels (recall that the columns of the data matrix are the input data points). Denote the top k eigenvectors as $v^{(1)}, \ldots, v^{(k)}$.
2. Define $\pi_j = \frac{1}{k} \sum_{i=1}^{k} v_j^{(i)2}$.
3. Sample s columns independently from the probability distribution defined by π.

- Why does π define a probability distribution?

Proof: To show that π defines a probability distribution, just need to show:

$$\sum_{j=1}^{n} \pi_j = 1$$

where n is the number of features. Since $v^{(1)}, \ldots, v^{(k)}$ are eigenvectors of a covariance matrix and as we know that covariance matrix is symmetric, we have $v^{(1)}, \ldots, v^{(k)}$ are orthonormal. i.e.:

$$\sum_{j=1}^{n} v_j^{(i)2} = 1, \forall i$$

Thus, we have:

$$\sum_{j=1}^{n} \pi_j = \sum_{j=1}^{n} \frac{1}{k} \sum_{i=1}^{k} v_j^{(i)2}$$
$$= \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{n} v_j^{(i)2}$$
$$= \frac{1}{k} \sum_{i=1}^{k} 1$$
$$= 1$$

- For each $k \in \{1,\ldots,10\}$ and each $s \in \{1,\ldots,20\}$, report the average test error over 100 iterations of naïve Bayes using only the s selected features (note that there may be some duplicates, so only include each feature once).

k\c	1	2	3	4	5	6	7	8	9	10
1	0.55519	0.55096	0.55885	0.54904	0.5625	0.56269	0.56442	0.56654	0.56827	0.57192
2	0.58231	0.57077	0.57596	0.57019	0.61923	0.58865	0.59442	0.59192	0.60058	0.60692
3	0.57808	0.57327	0.57942	0.56981	0.61212	0.60519	0.61808	0.59077	0.62577	0.60558
4	0.56558	0.57981	0.58538	0.58	0.63365	0.61231	0.61981	0.63154	0.63558	0.62058
5	0.58077	0.58442	0.58442	0.59288	0.635	0.6175	0.64423	0.62327	0.62731	0.64058
6	0.57423	0.58192	0.56885	0.58096	0.66077	0.63096	0.65	0.62865	0.64173	0.64519
7	0.57154	0.59269	0.59077	0.58731	0.65808	0.64596	0.65058	0.65096	0.66115	0.64885
8	0.58019	0.59904	0.6	0.58769	0.66904	0.64865	0.66404	0.64154	0.66692	0.64577
9	0.58385	0.59038	0.58654	0.59808	0.66577	0.65192	0.6625	0.66904	0.67135	0.65538
10	0.56635	0.59635	0.59654	0.58019	0.66712	0.66808	0.66481	0.66942	0.66442	0.67615
11	0.57769	0.58885	0.58635	0.59019	0.66981	0.64827	0.67923	0.67885	0.68942	0.67615
12	0.57231	0.59346	0.58885	0.59154	0.67058	0.67154	0.67635	0.67981	0.68615	0.67827
13	0.58673	0.59635	0.58577	0.59192	0.68615	0.68019	0.67962	0.66885	0.68538	0.67692
14	0.58231	0.59288	0.58635	0.58077	0.67635	0.66827	0.69558	0.69096	0.68942	0.67731
15	0.58404	0.59769	0.59	0.58077	0.68096	0.67808	0.69808	0.69596	0.68692	0.68731
16	0.58385	0.58577	0.58462	0.58481	0.68731	0.68173	0.68788	0.69212	0.69962	0.69096
17	0.57827	0.58519	0.58865	0.58519	0.68577	0.68596	0.69346	0.67808	0.68712	0.69731
18	0.57558	0.58481	0.58269	0.58327	0.68788	0.67962	0.69038	0.67231	0.69385	0.68846
19	0.58788	0.58212	0.58769	0.57981	0.69154	0.68904	0.69808	0.68731	0.70192	0.70308
20	0.57962	0.58538	0.58846	0.58058	0.69481	0.68385	0.69846	0.69423	0.70654	0.68442

best_Acc = ('k ', 9, 's', 20) 0.706538461538

Note:
The requirement is to report error, here the solution is accuracy. You can get error = 1 – accuracy.

And (This result might be different each time.)

- Does this provide a reasonable alternative to naïve Bayes without feature selection on this data set? What are the pros and cons of this approach?

Solution: It does provide a reasonable alternative to naïve Bayes without feature selection, they both have the test error rate about 30% (with a good pair of k and s).
Pros are obvious when there are too many features in the data set, it can reduce dramatically on the calculation expense. Cons should be how to pick a good k/s pair. Actually, running through all k and s values in this particular example is time consuming.

Problem 2: Spectral Clustering

In this problem, we will take a look at a simple clustering algorithm based on the eigenvalues of a matrix. This approach to clustering is typically referred to as spectral clustering. The basic approach is as follows, given a collection of n points, $x_1, \ldots, x_n \in \mathbb{R}^m$, we construct a matrix of $A \in \mathbb{R}^{n \times n}$ of similarities between them. Here, $A_{ij} = A_{ji} = e^{-\frac{1}{2\sigma^2}||x_i - x_j||^2}$ is the similarity between x_i and x_j for some $\sigma \in \mathbb{R}$.

The Basic Algorithm

Write a function in MATLAB that, given the matrix of similarities, performs the following operations.

1. Compute the "Laplacian matrix", $L = D - A$, where D is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$ for all i. Argue that this matrix is positive semidefinite.

Proof: since A is symmetric, let $A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{12} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}$ where $A_{ij} = e^{-\frac{1}{2\sigma^2}||x_i - x_j||^2}$, note that $A_{ij} > 0 \; \forall i, j$; let $D = \text{diag}\begin{bmatrix} \sum_j A_{1j} & \sum_j A_{2j} & \cdots & \sum_j A_{nj} \end{bmatrix}$.

Thus, $L = D - A = \begin{bmatrix} \sum_{j \neq 1} A_{1j} & -A_{12} & \cdots & -A_{1n} \\ -A_{12} & \sum_{j \neq 2} A_{2j} & \cdots & -A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -A_{1n} & -A_{2n} & \cdots & \sum_{j \neq n} A_{nj} \end{bmatrix}$.

Recall, a square matrix is said to be **diagonally dominant** if for every row of the matrix, the magnitude of the diagonal entry in a row is larger than or equal to the sum of the magnitudes of all the other (non-diagonal) entries in that row. One can check that the matrix L defined above is diagonally dominant.

Finally, a real symmetric diagonally dominant matrix with real non-negative diagonal entries is positive semidefinite.

2. Compute the eigenvectors of the Laplacian using **eig()** in MATLAB.

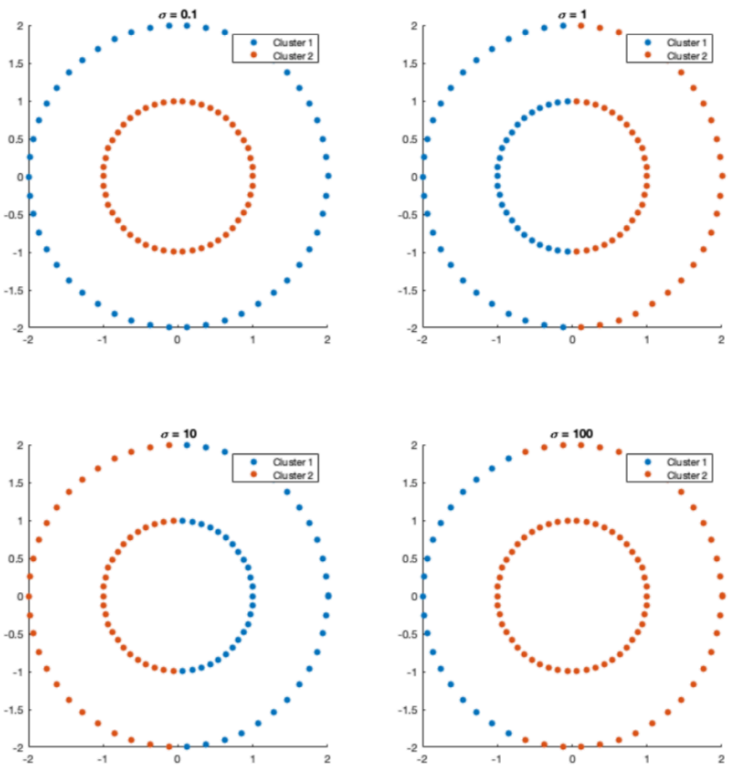
3. Construct a matrix $V \in \mathbb{R}^{n \times k}$ whose columns are the eigenvectors that correspond to the k smallest eigenvalues of L.
4. Let $y_1, \ldots, y_n$ denote the rows of V . Use the **kmeans()** algorithm in MATLAB to cluster the rows of V into clusters $S_1, \ldots, S_k$.
5. The final clusters $C_1, \ldots, C_k$ should be given by assigning vertex i of the input set to cluster C_j if $y_i \in S_j$.

Solution: the source code for this part is: *spectral.m*

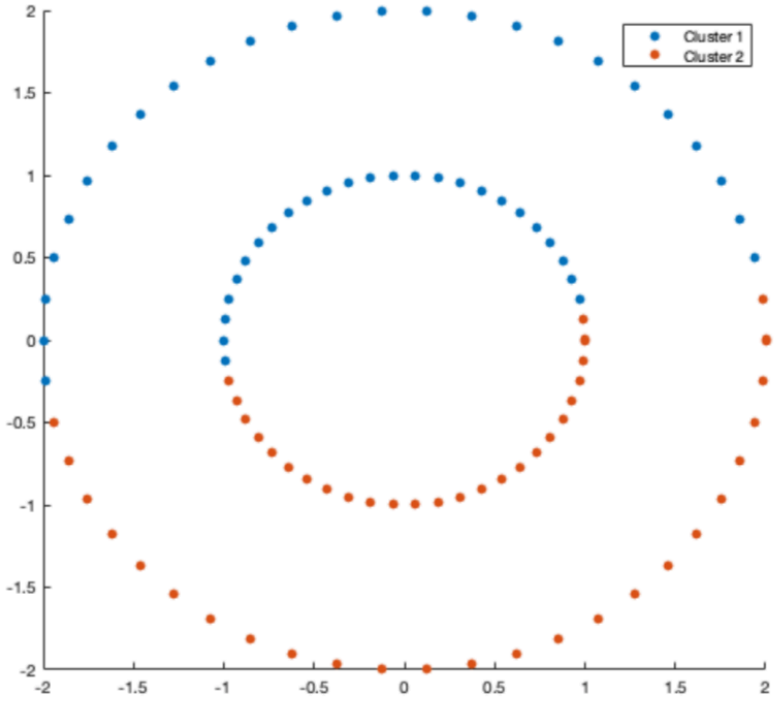
A Simple Comparison

2. Use the **scatter()** function in MATLAB to output the points colored by which cluster that they belong to for both algorithms. Include your MATLAB output in your submission.

Solution: the following are the outputs for spectral clustering with $\sigma = 0.1, 1, 10, 100$:



and the following is the output for k-means clustering:



3. Find a choice of σ such that the spectral method outperforms k-means. How do you know that there is no k-means solution (i.e., a choice of centers and clusters) that performs this well? Include your MATLAB output in your submission.

Solution: As we can see from the figures shown above, when $\sigma = 0.1$ the spectral method outperforms k-means. It is obviously that the data points given by the function circs.m are two concentric circles. With the result given by spectral algorithm with $\sigma = 0.1$, we have the points on the inner circle belongs to one clustering and the points on the outer circle belongs to the other clustering. This is simply impossible for the regular k-means with euclidean distance, since the two clusters have the same center.

Partitioning Images

the following are the outputs with $\sigma = 0.1, 1, 10, 100$:

and the output is: