Problem Set 4

CS 6375

Due: 3/31/2020 by 11:59pm

Note: all answers should be accompanied by explanations for full credit. Late homeworks will not be accepted.

Problem 1: PCA and Feature Selection (50pts)

In this problem, we will explore ways that we can use PCA for the problem of generating or selecting "good" features.

SVMs and PCA (25pts)

Consider the UCI Sonar data set (attached to this problem 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?
- For each $k \in \{1, 2, 3, 4, 5, 6\}$, 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.
- 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.
- 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?

PCA for Feature Selection (25pts)

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?
- Again, using the UCI Sonar data set, for each $k \in \{1, \dots, 10\}$ and each $s \in \{1, \dots, 20\}$, report the average test error of the SVM with slack classifier over 100 experiments. For each experiment use only the s selected features (note that there may be some duplicates, so only include each feature once).
- Does this provide a reasonable alternative to SVM with slack formulation without feature selection on this data set? What are the pros and cons of this approach?

Problem 2: Spectral Clustering (50pts)

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, $x_1, \dots, x_n \in \mathbb{R}^m$, 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 (20pts)

Write a function in MATLAB or Python 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.
2. Compute the eigenvectors of the Laplacian using `eig()` in MATLAB (numpy in Python).
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 $y_1, \dots, y_n$ denote the rows of V. Use the `kmeans()` algorithm in MATLAB (scikit-learn in Python) to cluster the rows of V into clusters $S_1, \dots, S_k$.
5. The final clusters $C_1, \dots, C_k$ should be given by assigning vertex i of the input set to cluster C_j if $y_i \in S_j$.

A Simple Comparison (15pts)

1. Use the spectral clustering algorithm above to compute the clustering for the matrix of two-dimensional points returned by the function `circs()` (attached to this problem set) above with $k = 2$ and different values of σ. Use the k-means algorithm in MATLAB/Python to compute an alternative clustering.
2. Use the `scatter()` function in MATLAB to output the points colored by which cluster that they belong to for both algorithms. Include your plot in your submission.

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 the output of your code in your submission.

Partitioning Images (15pts)

1. We can use the same spectral technique to partition images. Here, we consider each pixel of a grayscale image as a single intensity and construct a similarity matrix for pairs of pixels just as before.

2. Perform the same comparison of spectral clustering and k-means as before using the image `bw.jpg` that was attached as part of the homework. Again, set $k = 2$. You can use `imread()` to read an image from a file in MATLAB.