CS 273a: Intro to Machine Learning: Winter 2019

Homework 5

Due Date: **Friday, March 15, 2019**

The submission for this homework should be ***a single PDF file*** containing all of the relevant code, figures, and any text explaining your results. In addition to including any code you write, be sure to use Markdown cells in Jupyter to clearly explain the trends you observe and conclusions you draw.

Problem 1: Clustering (40 points)

In this problem, you will experiment with two clustering algorithms implemented in the updated `mltools` package: k-means and agglomerative clustering.

1. Load the standard Iris dataset, select the first two features, and ignore the class (or target) variables. Plot the data and see for yourself how "clustered" you think it looks. Include the plot, say how many clusters you think exist, and briefly explain why. (There are multiple reasonable answers to this question.) *(5 points)*

2. Run k-means on the first two features of the Iris data, for $k = 2$, $k = 5$, and $k = 20$. Try multiple (at least 5) different initializations for each k, and check to see whether they find the same solution; if not, pick the one with the best score. For the best clustering for each candidate k, create a plot with the data colored by assignment, and the cluster centers. You can plot the points colored by cluster assignments z using `ml.plotClassify2D(None,X,z)`. (You will need to also plot the cluster centers yourself.) *(15 points)*

3. Run agglomerative clustering on the first two features of the Iris data, first using *single linkage* and then again using *complete linkage*, using the algorithms implemented in `ml.cluster.agglomerative` from `cluster.py`). For each linkage criterion, plot the data colored by their assignments to 2, 5, and 20 clusters. (Agglomerative clustering does not require an initialization, so there is no need to run methods multiple times.) *(15 points)*

4. Briefly discuss similarities and differences in the outputs of the agglomerative clustering and k-means algorithms. *(5 points)*

Problem 2: EigenFaces (50 points)

In class, we discussed how PCA has been applied to faces, and showed some example results. Here, you'll explore this representation yourself. First, load the data and display a few faces to better understand the data format:

```python
X = np.genfromtxt("data/faces.txt", delimiter=None)  # load face dataset
plt.figure()
# pick a data point i for display
img = np.reshape(X[i,:],(24,24))          # convert vectorized data to 24x24 image patches
plt.imshow( img.T , cmap="gray")          # display image patch; you may have to squint
```

1. Subtract the mean of the face images ($X_0 = X - \mu$) to make your data zero-mean. (The mean should be of the same dimension as a face, 576 pixels.) Plot the mean face as an image. *(5 points)*

2. Use `scipy.linalg.svd` to take the SVD of the data, so that

$$X_0 = U \cdot \text{diag}(S) \cdot V_h$$

Since the number of faces is larger than the dimension of each face, there are at most 576 non-zero singular values; use the `full_matrices=False` argument to avoid using a lot of memory. As in the slides, then compute `W = U.dot( np.diag(S) )` so that $X_0 \approx W \cdot V_h$. Print the shapes of W and V_h. *(10 points)*

3. For $K = 1, \dots, 10$, compute the approximation to X_0 given by the first K eigenvectors (or eigenfaces): $\hat{X}_0 = W[:,:K] \cdot Vh[:K,:]$. For each K, compute the mean squared error in the SVD's approximation, `np.mean( (X_0 - \hat{X}_0)**2 )`. Plot these MSE values as a function of K. *(10 points)*

4. Display the first three principal directions of the data, by computing μ+α V[j,:] and μ-α V[j,:], where α is a scale factor (we suggest setting α to `2*np.median(np.abs(W[:,j]))`, to match the scale of the data). These should be vectors of length $24^2 = 576$, so you can reshape them and view them as "face images" just like the original data. They should be similar to the images in lecture. *(10 points)*

5. Choose any two faces and reconstruct them using the first K principal directions, for $K = 5, 10, 50, 100$. Plot the reconstructed faces as images. *(5 points)*

6. Methods like PCA are often called "latent space" methods, as the coefficients can be interpreted as a new geometric space in which the data are represented. To visualize this, choose 25 of the faces, and display them as images with the coordinates given by their coefficients on the first two principal components:

```python
idx = ...      # pick some data (randomly or otherwise); an array of integer indices

import mltools.transforms
coord,params = ml.transforms.rescale( W[:,0:2] )  # normalize scale of "W" locations
plt.figure();
for i in idx:
    # compute where to place image (scaled W values) & size
    loc = (coord[i,0],coord[i,0]+0.5, coord[i,1],coord[i,1]+0.5)
    img = np.reshape( X[i,:], (24,24) )           # reshape to square
    plt.imshow( img.T , cmap="gray", extent=loc ) # draw each image
    plt.axis( (-2,2,-2,2) )                       # set axis to a reasonable scale
```

This plot is a good way to gain intuition for what the PCA latent representation captures. *(10 points)*

Problem 3: Statement of Collaboration (5 points)

It is **mandatory** to include a *Statement of Collaboration* in each submission, that follows the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed.

All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments in particular, I encourage students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, possible bugs in the support code, and the relevant technical content *before* they start working on it. However, you should not discuss the specific solutions, and as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Piazza as much as possible, so that there is no doubt as to the extent of your collaboration.

Problem 4: Course Evaluation (5 points)

What were your favorite parts of CS273a, and what machine learning topics do you wish you had learned more about? Please complete the official UC Irvine course evaluation to let us know. (You do *not* need to include any answer to this question in your submission; we will automatically determine which students complete the evaluation. Note also that we do *not* have access to the evaluation scores submitted by individual students; we can only see aggregate statistics of these scores.)