CS 178 Homework 5

Machine Learning and Data Mining, CS 178, Winter 2020

Due Date: 11:59pm Friday Mar 13th, submit via Gradescope

The submission for this homework, as for others so far, should be **one stand-alone PDF file** containing all of the relevant code, figures, and any text explaining your results. The code is not as important as your explanations of the trends and conclusions, so use the Markdown cells in Jupyter quite liberally. In this homework, we will be playing with a number of already implemented clustering and dimensionality reduction techniques.

Points: This homework adds up to a total of **100 points**, as follows:

Problem 1: Clustering	45 points
Problem 2: EigenFaces	50 points
Statement of Collaboration	5 points

Be sure to re-download and replace the `mltools` package; it contains a number of new algorithms for you.

1 Clustering, 45 points

The code this week provides the three clustering algorithms we discussed: k-means, agglomerative clustering, and EM for Gaussian mixture models; we will explore the first two here. (These functions are also provided in many 3rd party toolboxes; you are free to use those if you prefer.) In this problem, you'll do some basic exploration of the clustering techniques.

1. Load the usual Iris data restricted to the first two features, and ignore the class / target variable. Plot the data and see for yourself how "clustered" you think it looks. Include the plot, and mention how many clusters you think exist (no wrong answer here). *(5 points)*

2. Run k-means on the data, for $k = 2$, $k = 5$, and $k = 20$. Try a few (at least 5 each) different initializations and check to see whether they find the same solution; if not, pick the one with the best score. For the chosen assignment for each k, include a plot with the data, colored by assignment, and the cluster centers. You can plot the points colored by assignments using `ml.plotClassify2D(None,X,z)`, where z are the resulting cluster assignments of the data. You will have to additionally plot the centers yourself. *(15 points)*

3. Run agglomerative clustering on the data, using *single linkage* and then again using *complete linkage*, each with 2, 5, and then 20 clusters (using `ml.cluster.agglomerative` from `cluster.py`). Again, plot with color the final assignment of the clusters. (This algorithm has no initialization issues; so you do not have to try multiple initializations.) *(20 points)*

4. Describe similarities and differences in the results from the agglomerative clustering and k-means. *(5 points)*

2 EigenFaces, 50 points

Note: you may have to wait until after the Tuesday (March 3) lecture to understand this problem.

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

```
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. *(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 data is larger than the number of dimensions, there are at most 576 non-zero singular values; you can use `full_matrices=False` to avoid using a lot of memory. As in the slides, we suggest computing `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 $K = 1 \ldots 12$, compute the approximation to X_0 given by the first K eigendirections, e.g., $\hat{X}_0 = W[:, :K] \cdot Vh[:K, :]$, and use them to 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, for example, `2*np.median(np.abs(W[:,j]))`, to get a sense of the scale found in 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 $K = 5, 10, 50, 100$. *(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 being described. To visualize this, choose a few faces (say 25), and display them as images in a two-dimensional plane (as in the lecture), where the two-dimensional coordinates are given by their coefficients on the first two principal components:

```
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(); plt.hold(True);                     # you may need this for pyplot
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 can often help you get a "feel" for what the latent representation is capturing. *(10 points)*

Statement of Collaboration (5 points)

It is **mandatory** to include a *Statement of Collaboration* in each submission, with respect to 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, we encourage the students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, bugs in my 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 (i.e. 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.

Acknowledgements

This homework is adapted (with minor changes) from one made available by Alex Ihler's machine learning course. Thanks, Alex!