CSCI 447/547: Assignment 3

March 9, 2018

Due April 6th. Problems marked with an asterisk are only required for grad students. However, undergrads may undertake them at no penalty, and will receive extra credit for successful completion. Turn in responses and plots, as well as code.

1 Support Vector Machine

1A 10 pts

Download the Wine data set at https://archive.ics.uci.edu/ml/datasets/wine. The Wine data set has 3 classes, with 13 attributes, all of which are real valued. Implement a support vector machine classifier (probably using sklearn). To begin with, use a linear kernel function. Split the data into training (2/3) and test (1/3) sets using the function sklearn.model_selection.train_test_split. Train the model, and then report your test set and trial set accuracy.

1B 10 pts

train_test_split is randomized; every time you run it it will partition the data into test and training sets differently. As such, the classification error can vary for different random selections of the members of these sets. To explore the sensitivity of the model to this randomness, train your classifier 100 times, with different random selections of the training and test set each time (ensure that you don't seed the random number generator!). Produce a histogram of *test set* accuracy, and report the mean and standard deviation.

1C 10 pts

Repeat the previous section twice more, but with polynomial and radial basis function kernels. Comment on which of these kernel functions is the most robust (i.e. for which kernel is the test set accuracy least sensitive to randomness in the data sets?), and which one is the most accurate (i.e. which has mean test set accuracy closest to 100%).

1D 10 pts

If you did not normalize your input features in the previous section, normalize them such that they have mean zero and standard deviation one. If you already normalized your data, then unnormalize. Re-run sections B and C. Does normalization affect the accuracy of your classifier? If so, which kernel is most sensitive to scaling the data? Speculate on why this is.

2 Principal Component Analysis 1

2A 10 pts

Load the datasets pca1.npz and pca2.npz. These data sets each have three features. Plot these features in 3D (See https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html, for a tutorial about 3D plotting in matplotlib). Just by visual inspection, how many significantly non-zero principal components does each data set have?

2B 10 pts

Verify the results of the previous section by performing as PCA on each of these data sets, and reporting on the significance of each principal component. Assuming that you use sklearn for this analysis, the amount of variance explained by a given principal component can be queried with pca.explained_variance_

3 Principal Component Analysis 2

3A 5 pts

Download a subset of the data set 'labelled faces in the wild' (http://vis-www.cs.umass.edu/lfw/ for a description). sklearn makes this easy with the function

```
sklearn.datasets.fetch_lfw_people(min_faces_per_person=50,resize=0.7).
```

Utilize only faces for which there are more than 50 training examples with the keyword argument min_faces_per_person=50. Resize the images to 0.5 times their native resolution with the keyword argument resize=0.7.

3B 10 pts

Perform a principal component analysis on these observations (HINT: this is identical to performing PCA on MNIST as we did in class). Do not whiten the data at this stage (whitening scales the data to unit variance, but the resulting transformation is not invertible). Keep the first 100 principal components. Plot the first ten principal components as images (reshaping to the original image size). These resulting basis vectors are called 'eigenfaces'.

3C 10 pts

Plot the reconstruction of a training set example of your choice using the first principal component, ten principal components, and 100 principal components. More concretely, make three plots. The first plot should be the first coefficient of your training example after transformation times the first principal component. The second plot should be the linear combination of the first ten principle components, with coefficients given by your chosen training example (after being transformed). Similarly for the third plot. You should see the faces converge towards the original image.

3D (*) 10 pts

Train a support vector machine on these data (you may wish to run your PCA again, this time whitening the data). Once again, this is similar to what we did for MNIST. Print the classification accuracy and confusion matrix.

4 K-mean/Gaussian Mixture Models/Expectation Maximization

4A 15 pts

One use for clustering algorithms is in data compression. In the case of images, imagine that we want to reduce an image from 16 to 3 bits (255 to 8 colors) in the optimal way. This is called image quantization. A simple way to do this is to run a clustering algorithm on the image data's color bands. Run the K-means algorithm on an image of your choice (reshape it such that it has 3 features, the R,G, and B bands) specifying that there are 8 clusters. Replace each pixel value with the mean value of its assigned cluster. Print the quantized image and the original image.

4B (*) 10 pts

Implement the Expectation-Maximization algorithm for a Gaussian Mixture Model (see Bishop P. 438/439). Apply it to the old faithful data set (faithful.dat) and plot the resulting Gaussians (see the code plot_ellipse.py for code that plots these Gaussians)