CS5785 Homework 2

The homework is generally split into programming exercises and written exercises.

This homework is due on **March 8, just before class**. Upload your homework to CMS. Please upload code as a single `.zip` file and the writeup as a single `.pdf` file. A complete submission should include:

1. A write-up as a single `.pdf` file

2. Source code and data files for all of your experiments (AND figures) in `.py` files if you use Python or `.ipynb` files if you use the IPython Notebook. If you use some other language, include all build scripts necessary to build and run your project along with instructions on how to compile and run your code.

The write-up should be in professional lab report format. More specifically, it should contain a general summary of what you did, how well your solution works, any insights you found, etc. On the cover page, include the class name, homework number, and team member names. You are responsible for submitting clear, organized answers to the questions. You could use online LaTeX templates from Overleaf, under "Homework Assignment" and and "Project / Lab Report".

Please include all relevant information for a question, including text response, equations, figures, graphs, output, etc. If you include graphs, be sure to include the source code that generated them.

Please pay attention to the discussion board for relevant information regarding updates, tips, and policy changes. You are encouraged (but not required) to work in groups of 2.

IF YOU NEED HELP

There are several strategies available to you.

- If you ever get stuck, the best way is to ask your teammates on Piazza[1]. That way, your solutions will be available to the other students in the class.

- Your instructor and TAs will offer office hours[2], which are a great way to get some one-on-one help.

- You are allowed to use well known libraries such as `scikit-learn`, `scikit-image`, `numpy`, `scipy`, etc. in this assignment. Any reference or copy of public code repositories should be properly cited in your submission (examples include Github, Wikipedia, Blogs).

---

[1] http://piazza.com/cornell/fall2016/cs5785
[2] http://cs5785-cornell-tech-16fall.github.io/contact.html

PROGRAMMING EXERCISES

1. Eigenface for face recognition.

   In this assignment you will implement the Eigenface method for recognizing human faces. You will use face images from The Yale Face Database B, where there are 64 images under different lighting conditions per each of 10 distinct subjects, 640 face images in total. With your implementation, you will explore the power of the Singular Value Decomposition (SVD) in representing face images.

   Read more (optional):

   - Eigenface on Wikipedia: https://en.wikipedia.org/wiki/Eigenface
   - Eigenface on Scholarpedia: http://www.scholarpedia.org/article/Eigenfaces

   (a) Download The Face Dataset. After you unzip `faces.zip`, you will find a folder called *images* which contains all the training and test images; *train.txt* and *test.txt* specifies the training set and test (validation) set split respectively, each line gives an image path and the corresponding label.

   (b) Load the training set into a matrix **X**: there are 540 training images in total, each has 50×50 pixels that need to be concatenated into a 2500-dimensional vector. So the size of **X** should be 540×2500, where each row is a flattened face image. Pick a face image from **X** and display that image in grayscale. Do the same thing for the test set. The size of matrix $\mathbf{X_{test}}$ for the test set should be 100×2500.

   Below is the sample code for loading data from the training set. You can directly run it in IPython Notebook:

```python
import numpy as np
from scipy import misc
from matplotlib import pylab as plt
import matplotlib.cm as cm
%matplotlib inline

train_labels, train_data = [], []
for line in open('./faces/train.txt'):
    im = misc.imread(line.strip().split()[0])
    train_data.append(im.reshape(2500,))
    train_labels.append(line.strip().split()[1])
train_data, train_labels = np.array(train_data, dtype=float), np.array(train_labels, dtype=int)

print train_data.shape, train_labels.shape
plt.imshow(train_data[10, :].reshape(50,50), cmap = cm.Greys_r)
plt.show()
```

   (c) Average Face. Compute the *average face μ* from the whole training set by summing up every column in **X** then dividing by the number of faces. Display the *average face* as a grayscale image.

(d) Mean Subtraction. Subtract average face μ from every column in **X**. That is, $\mathbf{x_i} := \mathbf{x_i} - \mu$, where $\mathbf{x_i}$ is the i-th column of **X**. Pick a face image after mean subtraction from the new **X** and display that image in grayscale. Do the same thing for the test set $\mathbf{X_{test}}$ using the pre-computed average face μ in (c).

(e) Eigenface. Perform Singular Value Decomposition (SVD) on training set **X** ($\mathbf{X} = \mathbf{U\Sigma V}^T$) to get matrix $\mathbf{V}^T$, where each row of $\mathbf{V}^T$ has the same dimension as the face image. We refer to $\mathbf{v_i}$, the i-th row of $\mathbf{V}^T$, as i-th *eigenface*. Display the first 10 eigenfaces as 10 images in grayscale.

(f) Low-rank Approximation. Since Σ is a diagonal matrix with non-negative real numbers on the diagonal in non-ascending order, we can use the first r elements in Σ together with first r columns in **U** and first r rows in $\mathbf{V}^T$ to approximate **X**. That is, we can approximate **X** by $\hat{\mathbf{X}}_{\mathbf{r}} = \mathbf{U}[:,: r] \ \Sigma[: r,: r] \ \mathbf{V}^T[: r,:]$. The matrix $\hat{\mathbf{X}}_{\mathbf{r}}$ is called rank-r approximation of **X**. Plot the rank-r approximation error $\|\mathbf{X} - \hat{\mathbf{X}}_{\mathbf{r}}\|_F$ [3] as a function of r when $r = 1, 2, \ldots, 200$.

(g) Eigenface Feature. The top r eigenfaces $\mathbf{V}^T[: r,:] = \{v_1, v_2, \ldots, v_r\}^T$ span an r-dimensional linear subspace of the original image space called *face space*, whose origin is the average face μ, and whose axes are the eigenfaces $\{v_1, v_2, \ldots, v_r\}$. Therefore, using the top r eigenfaces $\{v_1, v_2, \ldots, v_r\}$, we can represent a 2500-dimensional face image **z** as an r-dimensional feature vector **f**: $\mathbf{f} = \mathbf{V}^T[: r,:] \ \mathbf{z} = [v_1, v_2, \ldots, v_r]^T \mathbf{z}$. Write a function to generate r-dimensional feature matrix **F** and $\mathbf{F_{test}}$ for training images **X** and test images $\mathbf{X_{test}}$, respectively (to get **F**, multiply **X** to the transpose of first r rows of $\mathbf{V}^T$, **F** should have same number of rows as **X** and r columns; similarly for $\mathbf{X_{test}}$).

(h) Face Recognition. Extract training and test features for $r = 10$. Train a Logistic Regression model using **F** and test on $\mathbf{F_{test}}$. Report the classification accuracy on the test set. Plot the classification accuracy on the test set as a function of r when $r = 1, 2, \ldots, 200$. Use "one-vs-rest" logistic regression, where a classifier is trained for each possible output label. Each classifier is trained on faces with that label as positive data and all faces with other labels as negative data. `sklearn` calls this "ovr" mode.

(i) Repeat part (1h), but instead of using logistic regression, train a Naive Bayes classifier on the training set and report accuracy on the test set.

2. **Regularized linear regression.**

(a) Join the House Prices: Advanced Regression Techniques competition on Kaggle. Download the training and test data. The competition page describes how these files are formatted.

(b) Tell us about the data. How many samples are there in the training set? How many features? Which features are categorical?

(c) What variables seem to be important? Which seem to correlate with the sale price? Plot the relationship between sale price and year of sale, garage area, lot area, and other variables of your choice.

(d) Try to run a variety of regression methods using sklearn methods:

- k-Nearest Neighbors with leave-one-out cross validation
- Ordinary least squares with 10-fold cross validation
- Ridge regression with 10-fold cross validation

---

[3] $\|.\|_F$ is the Frobenius Norm of a matrix: $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{i=1}^{n} |a_{ij}|^2}$, which can be directly computed in numpy.

- LASSO with 10-fold cross validation
- A regression model of your choice using backward stepwise feature selection with 10-fold cross validation
- A regression model of your choice using forward stepwise feature selection with 10-fold cross validation

  For each model, give a brief description of how it works and report validation accuracy.

(e) Pick three models to investigate further. Which variables are being retained by these models, and which are regularized away? Do these variables match your intuitions about which variables are important and which are not?

(f) Train your best-performing classifier with all of the training data, and generate test labels on test set. Submit your results to Kaggle and report the accuracy. Is it higher or lower than the cross validation accuracy? Why might that happen?

WRITTEN EXERCISES

You can find links to the textbooks for our class on the course website.
Submit the answers to these questions along with your writeup as a single `.pdf` file. We do recommend you to type your solutions using LaTeX or other text editors, since hand-written solutions are often hard to read. If you handwrite them, please be legible!

1. **Naive Bayes classifiers.** In a medical study, 100 patients all fell into one of three classes: Pneumonia, Flu, or Healthy. The following database indicates how many patients in each class had fever and headache.

   Consider a patient with fever and no headache.

   (a) What *probability* would a Bayes optimal classifier assign to the proposition that the patient has Pneumonia, Flu, or neither? Show your work. (For this question, the three values should sum to 1.)

   (b) What *probability* would a naive Bayes classifier assign to the proposition that the patient has Pneumonia, Flu, or neither? Show your work. (For this question, the three values should sum to 1.)

Pneumonia		
Fever	Headache	Count
T	T	5
T	F	0
F	T	4
F	F	1
Total:		10

Flu		
Fever	Headache	Count
T	T	9
T	F	6
F	T	3
F	F	2
Total:		20

Healthy		
Fever	Headache	Count
T	T	2
T	F	3
F	T	7
F	F	58
Total:		70

2. **Naive Bayes for data with nominal attributes.** Given the training data in the table below (Buy Computer data), predict the class of the following new example using Naive Bayes classification: age≤30, income=medium, student=yes, credit-rating=fair. Please show your work.

ID	age	income	student	credit-rating	Class: buys-computer
1	≤30	high	no	fair	no
2	≤30	high	no	excellent	no
3	31…40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31…40	low	yes	excellent	yes
8	≤30	medium	no	fair	no
9	≤30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	≤30	medium	yes	excellent	yes
12	31…40	medium	no	excellent	yes
13	31…40	high	yes	fair	yes
14	>40	medium	no	excellent	no

3. **SVD of Rank Deficient Matrix.** Consider matrix M. It has rank 2, as you can see by observing that there times the first column minus the other two columns is 0.

$$M = \begin{bmatrix} 1 & 0 & 3 \\ 3 & 7 & 2 \\ 2 & -2 & 8 \\ 0 & -1 & 1 \\ 5 & 8 & 7 \end{bmatrix}. \tag{1}$$

(a) Compute the matrices $M^T M$ and $M M^T$.

(b) Find the eigenvalues for your matrices of part (a).

(c) Find the eigenvectors for the matrices of part (a).

(d) Find the SVD for the original matrix M from parts (b) and (c). Note that there are only two nonzero eigenvalues, so your matrix Σ should have only two singular values, while U and V have only two columns.

(e) Set your smaller singular value to 0 and compute the one-dimensional approximation to the matrix M.

4. **Graphical PCA and LDA**. Consider the dataset shown in Fig. , with two dimensions and two classes. Suppose we train an LDA classifier trained on this dataset. Recall that LDA considers the log-odds ratio between two distributions when comparing the two classes, $C = 1$ and $C = 2$:

$$\log \frac{P(C = 1|x)}{P(C = 2|x)} \tag{2}$$

The functions $P(x|C = 1)$ and $P(x|C = 2)$ are assumed to be Gaussian, $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$. For more information about LDA, see HTF Sec. 4.3. The class textbook is freely available on the course website.
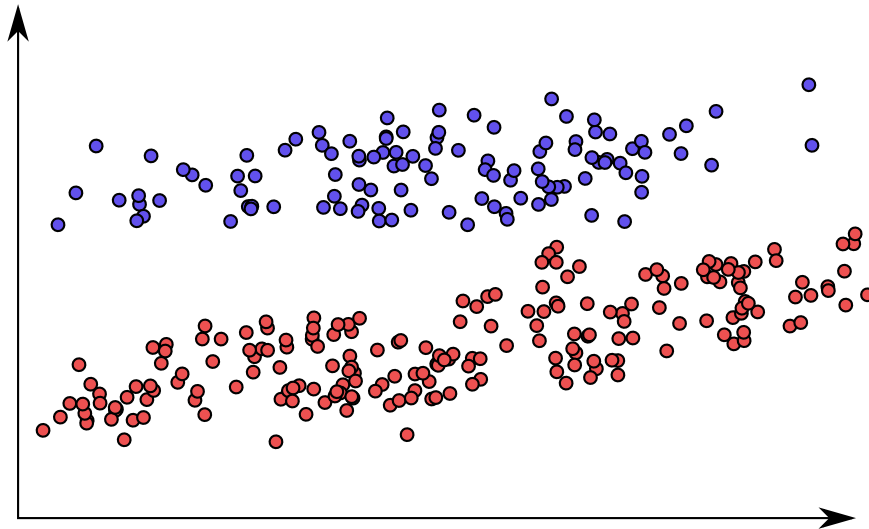
Figure 1: A 2-dimensional dataset with two classes, red and blue.

(a) What is the assumption made by LDA about Σ_1 and Σ_2?

(b) Please draw the *level set* of LDA's $P(x|C = 1)$ and $P(x|C = 2)$ on the chart. For example, you might draw an outline around the set of points one standard deviation away from the mean for each class.

(c) Please draw the direction of the dataset's first principal component on the figure.

(d) Please draw the direction of the first LDA component on the figure. Be sure to clearly indicate which is which. Why are the PCA and LDA components different?

(e) How do class labels influence the components chosen by PCA?

Feel free to reproduce the figure in your homework if you need to.