Mini Project #1

4 Problem

In this problem we are going to implement the LDA for 20 Newsgroups data set. We are going to:

1. We begin by getting familiar with the data set and split the data set into two groups, i.e. training data and test data.
2. Then we use the one dimension reduction technique (i.e. PCA) to reduce the feature dimension since working with original data set will probably result in memory crash.
3. After reducing the feature dimension, we start building LDA model using the train data. In LDA we basically compute the model parameters by using formulas in (4:36), (4:37) and (4:38) in K. P. Murphy book with some minor changes to make the computations more efficient.
4. Finally, we predict the LDA with the computed parameters on the test data and compute its accuracy rate and compare its performance to other linear models built in SciKit-Learn package.

Please open the jupyter notebook called LDA_student.ipynb and start working on LDA problem by following the instruction.

5 Problem

5.1 KNN and 4 News group Classification

You will implement k-Nearest Neighbor (KNN) classifiers on the NEWS_DATASET . You are going to use the 4 classes ('alt.atheism','talk.religion.misc','comp.graphics','sci.space') among the original 20 classes.

5.1.1 Run: hw1_student → python features.py

5.1.2 Implement KNN algorithm and choose the parameter K.

1. Your working files are :
   - hw1_student → knn.ipynb
   - hw1_student → cs536_1 → models → k_nearest_neighbor.py
2. Fill out the working files with your code as appropriate. For the metric, please use the L2 norm.
3. When you finish your implementation, you will find validation accuracy rates for the different Ks at the **train_ratio:1.0** (controlling number of training dataset).
4. Now, you will change the **train_ratio** from 0.1 to 1.0 (interval step : 0.1) and guess the appropriate Ks for each value of the **train_ratio**s.
5. How did you choose the K? Explain your rule for the selection of the K.

5.1.3 Prediction on Test Set

1. Your working files :
   - hw1_student → knn_test.ipynb and
   - hw1_student → cs536_1 → models → k_nearest_neighbor.py
2. Fill out the working files with your code as appropriate.
3. When you finish your implementation, you will find test accuracy rates for the different Ks at the **train_ratio:1.0**.
4. Now, you will change the **train_ratio** from 0.1 to 1.0 (interval step : 0.1) and find the test_ratios.
5. Based on the your test accuracies, are your previous selections (validation set) still valid? If not, how would you change the selection of the parameter K for each **train_ratio**.

Table 1: Validation Accuracy

Train_Ratio	K								
	1	**3**	**5**	**7**	**9**	**11**	**15**	**19**	K **Selection**
0.1									
0.2									
0.3									
0.4									
0.5									
0.6									
0.7									
0.8									
0.9									
1.0									

Table 2: Test Accuracy

Train_Ratio	K							
	1	**3**	**5**	**7**	**9**	**11**	**15**	**19**
0.1								
0.2								
0.3								
0.4								
0.5								
0.6								
0.7								
0.8								
0.9								
1.0								

5.1.4 Analysis

Based on overall experiments results, please characterize the KNN along with the number of training data. Do you have any suggestion for better K selection in section 5.1.2 experiment.

4 Problem

Let $X \sim \mathcal{N}(0,1)$ and $Y = W \cdot X$, where $P(W = +1) = P(W = -1) = 0.5$. Clearly, X and Y are not independent. Show that:

1. $Y \sim \mathcal{N}(0,1)$.
2. $cov[X,Y] = 0$. Thus X and Y are uncorrelated but dependent, even though they are both Gaussian. Hint: you can use the rule of iterated expectation $\mathbb{E}[XY] = \mathbb{E}[\mathbb{E}[XY|W]]$.
3. Sample four sets of 100 random points X and construct the corresponding 100 points $Y|X$. Draw the scatter plots of $\mathcal{D}_i = \{(X_n, Y_n)\}_{n=1}^{N=100}$ for those four sets, $i = 1, \ldots, 4$.

   Note: Use `numpy.random.seed(0)` before sampling points for the four sets. (But do this only once, not for each set.) To sample Gaussian points use `numpy.random.randn` and use `numpy.random.rand` to create samples of $Y|X$.
4. Now find the optimal least sqaures linear regressors $Y \approx w \cdot X + w_0$ for the four datasets. Overlay those regressors on the scatter plots above. Comment on differences between the four cases and explain why there are differences, if any.

5 Problem

5.1 Linear Regression for News group Classification

You will implement Linear Regression for classification problem on the NEWS_DATASET. ('alt.atheism','talk.religion.misc','comp.graphics','sci.space') . You are going to use the 4 classes among the original 20 classes. You will encode group labels based on the two schemes. One is discrete scalar label and the other one is one-hot vector. For the group "3" example, the encoding can be one-hot vector representation like "[0,0,1,0]" or it can be represented by "2" (discrete labels). For each of the two cases, we will learn the linear regression model $y = w^T \cdot \phi(x)$ and define an classifier function C(y) mapping the y to the categorical encodings(discrete scalar label or one-hot vector). Also, you will compare their performance based on the two design cases of $\phi(x)$.

5.1.1 DATA FILE : train_PCA.pkl and test_PCA.pkl

The files, Train_PCA and Test_PCA, are arrays that each shape is (2034,1500) and (1353,1500). PCA(Principal Components Analysis) is applied to the previous Tf-idf features samples (2034, 33810) and we used only 95% engery principal components and the reduced dimension is 1500.

5.1.2 Discrete Label Experiment

1. We will encode k^{th} data sample's label, c_k, into an element of the set $\{0, 1, 2, 3\}$. This implies that you can use the data samples' target labels as they are.
2. Your working files are :
   - hw2_student → Linear_Regression_Scalar.ipynb
   - hw2_student → cs536_2 → models → Linear_Regression_Classifier.py
3. In your model of $y = w^T \cdot \phi(x)$, specify the dimension(shape) of w , x , and y.
4. In this experiment, we will use the $\phi(x) = x$
5. What is your C(y) function choice? Draw your C(y) on the domain of y. Why do you choose the function?
6. Fill out the working files with your code as appropriate. For the cost, we will use the mean square error(MSE) with quadratic regularization, where n is the number of samples,

$$E(w) = \sum_{k=1}^{n} \frac{1}{2}\{c_k - C(w^T \cdot x_k)\}^2 + L||w||^2 \tag{1}$$

7. In the experiment, please compare the train and validation accuracies along with the different L values. Draw a plot $\log(L)$ vs train and $\log(L)$ vs validation accuracies together. What value L would you choose based on the plot? Why?

8. Based on the your choice of L, re-train the model and find the test accuracy.

5.1.3 One-Hot Vector Label Experiment

1. We will encode k^{th} sample's group label c_k into an element of the set $\{[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]\}$.

2. Your working files are :

   - hw2_student → Linear_Regression_hotvector.ipynb
   - hw2_student → cs536_2 → models → Linear_Regression_Classifier.py

3. In your model of $y = w^T \cdot \phi(x)$, specify the dimension (shape) of w , x , and y.

4. In this experiment, we will use the $\phi(x) = x$

5. If you do multi-class classification (choose only one), then what is your C(y) function choice?

6. Fill out the working files with your code as appropriate. we will use the mean square error(MSE) with quadratic regularization, where n is the number of samples,

$$E(w) = \sum_{k=1}^{n} \frac{1}{2} ||c_k - C(w^T \cdot x_k)||^2 + L||w||^2 \tag{2}$$

   In the experiment, please compare the train and validation accuracies along with the different L values. Draw a plot $\log(L)$ vs train and $\log(L)$ vs validation accuracies together. What value L would you choose based on the plot? Why?

7. Based on the your choice of L, re-train the model and find the test accuracy.

8. Plot the ROC curves for each group.

5.1.4 RBF feature $\phi_m(x) = exp\frac{-(x-\mu_m)^2}{2\cdot\sigma_m^2}$ (m^{th} component feature)

1. We will transform 1500 dimensional PCA features to 1000 dimensional RBF feature set.

2. You will be given the K-means clustering results with 1000 clusters. You could construct 1000 RBF functions by using the clustering results. File : Kmean_cluster.pkl

3. You will implement your own name RBF_featre.py which creates train_RBF.pkl and test_RBF.pkl based on the RBF functions.

4. By using the new features (train_RBF.pkl and test_RBF.pkl) and the files(Linear_Regression_hotvector.ipynb and Linear_Regression_Scalar.ipynb), find your L and the test result for the RBF features.

5. Fill out the Table 1.

6. For the one-hot vector, plot the ROC curves for the RBF.

5.2 Analysis

1. We learned w based on the mean square error (MSE) formulation. What does the MSE assume about $P(C = c|w, \phi(x))$ as we learn w based on the maximum a posteriori estimation? Do you think it is the right approach? If you can design the $P(C = c|w, \phi(x))$ again, what probabilistic density would you use and why?

2. Please compare the test performance of the above two implementations 5.1.2 and 5.1.3. Why one is better than the other?

3. In Table 1, does the RBF function help to improve the inferior encoding scheme? If it does, why it can improve the one? If it does not, what remedy would you suggest?

Table 1: Test Accuracy and L

Basis function	*Encoding*	
	Discrete Scalar	**One-Hot vector**
$\phi(x) = x$	test accuracy and L	
$\phi_m(x) = exp\frac{-(x-\mu_m)^2}{2 \cdot \sigma_m^2}$		

6 Problem: Optimal Bayesian Estimation

In this problem you are going to implement optimal Bayesian estimation model for classification of NEWS_DATASET base on different scenarios. Please open OptBayesEstim.ipynb and follow the instruction form there. All implementations and reports must be reported in that file. At the end of your work, just submit the Opt-BayesEstim.ipynbfile.

4. [80 pts] You will build a classifier $G_L(x)$ which can classify a new music file into a genre. You will use a music dataset MARSYAS, `http://marsyas.info/index.html`. Among the 10 genres, we will use only four genres (classical, jazz, pop,and rock) in this problem. Each genre has the 100 music files and each music file will be preprocessed into 1200 frame samples. (folder **data** → 320:train, 40:val, 40:test)

   (a) Music file into 1200 frames

      i. A digital file of a music can be represented by a time sequence vector X.

      ii. Then, the first frame is $F_X^1 = X[0:2048]$, the second frame is $F_X^2 = X[512:2560]$, the third frame is $F_X^3 = X[1024:3072]$, and so on.

      iii. Each music file will have the 1200 frames, so the G_L will classify each frame into a genre, and based on the individual frame's classification result, we will do majority voting to decide the final genre for the whole music file.

      iv. Each frame will be a basic unit for the feature extraction, and the feature dimension will be 32. Hence, basically, we will have 32 by 1200 matrix after feature extraction process.

   (b) Extract features

   For the feature extraction, we will use a libROSA, `https://librosa.github.io/librosa`, package and extract spectral features for each frame.

   - spectral_centroid → compute mean and std (2)
   - spectral_rolloff → compute mean and std (2)
   - number_of_zero crossing → compute mean and std (2)
   - flux difference → compute mean and std (2)
   - chroma_stft extract for 12 bins and compute mean and std for each (2 x 12 = 24)
   - total feature dimension 32

   (c) Average Frames

   Please read the code **feature_extraction.py** for the better understandings, and take a close look at the function of the variable **average_win**. Depending on the **average_win**, 1200 frames will be averaged over the number of the frames(window size). For example, if the average_win = 40 then, a music file will become 30 ($\frac{1200}{40}$) samples. Then, the 320 training music files will become 9600 (320 x 30) samples, so the final training set will be 32 by 9600 matrix.

   (d) G_L Algorithm

   The algorithm of the G_L does as the following.

      i. When we initialize G_L, we store train data set as dictionary D (dimension_of_feature, number_of_train_samples), the D's genre labels, and the L (model parameter).

      ii. Accept a new test data point x (dimension_of_feature, 1) as an input.

      iii. Compute w^*

$$w^* = \arg\min_w ||x - D \cdot w||^2 + L \cdot ||w||^1$$

      iv. Find g^*, and g^* will be output of the G_L.

$$g^* = \arg\min_g ||x - D \cdot (w \star \Pi_g)||^2, \text{where g} \in \{0, 1, 2, 3\}$$

      v. The definition of the operation $\star$ and the i^{th} component of the vector Π_g are followings.

$$[x_1, x_2] \star [y_1, y_2] = [x_1 \cdot y_1, x_2 \cdot y_2]$$

$$\Pi_g(i) = \begin{cases} 1 \text{ if the genre of } i^{th} \text{ column of D equals g} \\ 0 \text{ else otherwise} \end{cases}$$

      vi. Repeat 4(d)ii - 4(d)iv for all the test points, then do majority voting to decide 40 test/val file's genres.

(e) [40 pts] Implementation G_L and experiments

i. Your working files are:
   - Lasso_Regression_Classifier.ipynb
   - cs536_3 → models → Lasso_Regression_Classifier.py
   - feature_extraction.py
   - logistic.py

ii. Please run the feature_extraction.py, then it creates feature set in the feature_data folder.

iii. Run logistic.py then it will give you the logistic classifier accuracies based on the created feature.

iv. Fill out your code as appropriate(Lasso_Regression_Classifier.ipynb and Lasso_Regression_Classifier.py).

v. Find the validation accuracy according to the different L values. Based on the validation accuracy, choose the My L and find a test accuracy based on the My L.

vi. Fill out the table 1. The default value for the average_win was 40. Please modify average_win in the feature_extraction.py and Lasso_Regression_Classifier.ipynb and check validation/test results again in the new feature set.

Table 1: Experiment Results

Average_win	**Logistic Accuracy**	*LandValidationResults*									*TestResults*	
		0.1	**1**	**10**	**13**	**15**	**17**	**21**	**30**	**100**	**My L**	**Test accuracy**
20												
40												
80												
120												
240												