Assignment (4)

Artificial Intelligence and Machine Learning (24-787)

Due date: 11/15/2022 (Tue) @ 11:59 pm EST

In case a problem requires programming, it should be programmed in Python. In Programming, you should use plain Python language, unless otherwise stated. For example, if the intention of a Problem is familiarity with numpy library, it will be clearly noted in that problem to use numpy. Please submit your homework through Gradescope.

Submissions: There are two steps to submitting your assignment on Gradescope:

1. HW04 Writeup: Submit a combined pdf file containing the **answers to theoretical questions** as well as the **pdf form of the FILE.ipynb notebooks**.

- Convert the jupyter notebook to a pdf file. Ensure that the submitted notebooks have been run and the cell outputs are visible - **Hint:** Restart and Run All option in the Kernel menu. **Make sure all plots are visible in the pdf.**
- If an assignment has theoretical and mathematical derivation, scan your handwritten solution and make a PDF file.
- Then concatenate them all together in your favorite PDF viewer/editor. The file name (FILE)should be saved as **HW-assignmentnumber-andrew-ID.pdf**. For example for assignment 1, my FILE = HW-1-andrewid.pdf
- Submit this final PDF on Gradescope. During submission, it will prompt you to select the pages of the PDF that correspond to each question, **make sure to tag the questions correctly!**

2. HW04 Code: Submit a ZIP folder containing the FILE.ipynb notebooks for each of the programming questions. The ZIP folder containing your iPython notebook solutions should be named as HW-assignmentnumber-andrew-ID.zip

You can refer to Numpy documentation while working on this assignment. **ANY** deviations from the submission structure shown above would attract penalty to the assignment score. Please use Piazza for any questions on the assignment.

PROBLEM 1

Topic: PCA without skLearn **[30 points]**

a) Load and Standardize- 5 points: Write a function to load the dataset from q1-data/features.npy and normalize it to have mean of 0 and standard deviation of 1 for each feature.

b) Eigen-decomposition- 5 points: Write a function that takes standardized dataset as the input. The function should compute the Covariance Matrix and return the sorted eigenvalues in descending order and the corresponding eigenvectors. You can use **np.linalg.eig** to compute eigenvectors.Include the eigenvalues in your report.

c) Evaluation- 10 points: Write a function that iterates through dimensionality k from 1 to the number of dimensions in the input features and computes the variance explained on reducing the dimensionality to k. Use the function created in part b. Print the variance explained as a function of k. Include the eigenvalue for each k in your report. Which value of k would you pick and Why?

d) Visualization- 8 points: Write a function that projects the original data to a 2-dimensional feature subspace. Load the labels from q1-data/labels.npy and plot the 2-D representation as a scatter plot with labels as legends of the plot. If everything goes well, you'll get a figure like in Figure 1 below.

e) Theory- 2 points:

Assume you have a dataset with the original dimensionality as 2 and you have to reduce it to 1. Provide a sample scatter plot of the original data (less than 10 data) where PCA might produce misleading results.
Hint: PCA fails to find non-linear structure in data.

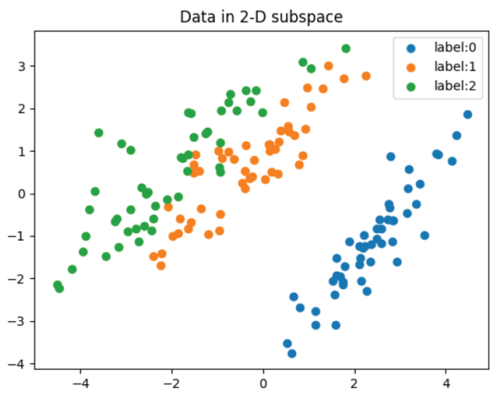

Fig. 1: Data in 2D Subspace

PROBLEM 2

Topic: Scikit-learn **[30 points]**

Amino acids are organic compounds that constitute protein in our bodies. In this problem, you are given a dataset which contains data for 20 different types of amino acids. For each type of amino acid, two features are extracted: ionic current in picoamps, and residence time in picoseconds.
Your job is to run various machine learning algorithms on the data using scikit-learn. However, do not use the whole dataset for training, split it into training dataset and testing dataset, accounting for 70% and 30% respectively. You may do this only once and use them for all the models in this problem. Some useful methods for this problem are: **pandas.read_csv**, **sklearn.model_selection**, **train_test_split**, **numpy.meshgrid**, **pyplot.pcolormesh**
For each question that asks to apply a machine learning algorithm, you need to print accuracy on both training and testing datasets in percentage; you also need to plot all the data (training + testing), using only one color, as well as the decision boundary on the same figure. You can use any color as long as it distinguishes the data .

a) Data Preprocessing- 5 points: The given data is not very well organized in that it doesn't follow the form of nExamples x (nFeatures + 1). The first row is a list of labels, and the two columns below each label are the data of the two features for that acid type (Hint: Please open the spreadsheet and take a look at the structure of data). Load the data and rearrange them so that x of shape (2000, 2) represents the examples and y of shape (2000,) represents the labels. Note: you need to map the string labels to numeric values so y consists of 2000 integers in the range [0, 19]. After you get x and y, split the data into training set and testing set.

b) K-Means- 10 points: Although you have been given the labels, let's pretend we are doing unsupervised learning for the purpose of using KMeans. Train the training set, setting number of clusters to 20. In the first figure, plot the decision boundary and data . Also plot centroids using red stars in the same figure. In another figure, scatter plot all the data using different colors for different classes

c) Random Forest- 10 points: Train the training dataset with random forest, leaving the number of estimators as default. In the first figure, plot decision boundary and data . Report accuracy on the testing set. In the second figure, plot the accuracy on testing set in percentage against the number of estimators which is in range [1, 25].

d) Analysis- 5 points: After applying different machine learning algorithms to this dataset, what have you noticed? Concisely write your findings and provide your reasoning. You can think of it from any perspective, such as accuracy, performance, decision boundary shapes and so on.

PROBLEM 3

Logistic Regression with Non-linear Decision Boundaries and Regularization [40 points]

In homework 2, you were asked to train a binary classifier using Logistic Regression (LR). This time, let's see how LR can be used to deal with datasets that are not linearly separable.

a) (Programming problem) Load and plot the data points in **(q3_data.csv)**. There are 3 columns in the given file. The first 2 columns are data points $\{x^{(i)} = (x_1^{(i)}, x_2^{(i)}), i = 0, 1, \cdots, m-1\}$ (m is the number of data points). And the 3rd column is the label $\{y^{(i)}, i = 0, 1, \cdots, m-1\}$. Points with different labels should be in different color. Output your plot.

b) (Programming problem) You should have noticed that there is no way a linear decision boundary would separate those data points. This case is called not linearly separable in the original space. Therefore, we'll transform the original 2D data points into a higher dimension space, where the transformed data points might become linearly separable again.

Now implement a function named map_feature(...) to transform the original 2D data into 28D data using the following transformation:

$$\text{map\_feature}(x) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & \cdots & x_1x_2^5 & x_2^6 \end{bmatrix}^T$$

The implemented function should take the original data matrix ($2 \times m$, not augmented with 1's) as input and return the transformed data matrix ($28 \times m$). This process is also called feature mapping.

c) (Programming problem) Formulating the LR problem in the transformed space is exactly the same as before (in homework 2). The cost function is given by

$$J(w) = \frac{1}{m}\sum_{i=0}^{m-1}\left[-y^{(i)}\log\left(h(w^Tx^{(i)})\right) - (1-y^{(i)})\log\left(1 - h(w^Tx^{(i)})\right)\right]$$

and the gradient is given by

$$G(w) = \frac{\partial J(w)}{\partial w} = \frac{1}{m}\sum_{i=0}^{m-1}\left(h(w^Tx^{(i)}) - y^{(i)}\right)x^{(i)}$$

where $h(\cdot)$ is the sigmoid function. Differently, the parameter w is of size 28×1 instead of 3×1 (including bias w_0).
It seems we are ready to implement a non-linear classifier. But hold on, we are not there yet!

While the feature mapping allows us to build a more expressive classifier, it is also more susceptible to overfitting. We'll use the regularization techniques taught in the lecture to combat the overfitting problem. The regularized cost function looks like this:

$$J(w) = \frac{1}{m}\sum_{i=0}^{m-1}\left[-y^{(i)}\log\left(h(w^Tx^{(i)})\right) - (1-y^{(i)})\log\left(1 - h(w^Tx^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n-1}w_j^2$$

where λ is the hyper-parameter to regularize w and $n = 28$. In the regularized logistic regression, the bias parameter w_0 should be excluded from regularization. So in the above cost function, the summation starts at $j = 1$ (corresponding to w_1). The gradients are as follows:

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{m}\sum_{i=0}^{m-1}\left(h(w^Tx^{(i)}) - y^{(i)}\right)x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{m}\sum_{i=0}^{m-1}\left[\left(h(w^T x^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j \qquad \text{for } j \geq 1$$

Now implement a function named logistic_regression_regularized(...) to solve the regularized logistic regression problem. Initialize w to be zeros and $\lambda = 1$. Choose your favorite gradient descent method (GD, SGD or mini-batch) and use proper learning rate and number of iterations. You may recycle the code from homework 2. Report the optimal values of w and plot the decision boundary in the original 2D space (Hint: numpy.meshgrid and matplotlib.pyplot.contour will be your friend). Your decision boundary should be similar to Fig. 2.

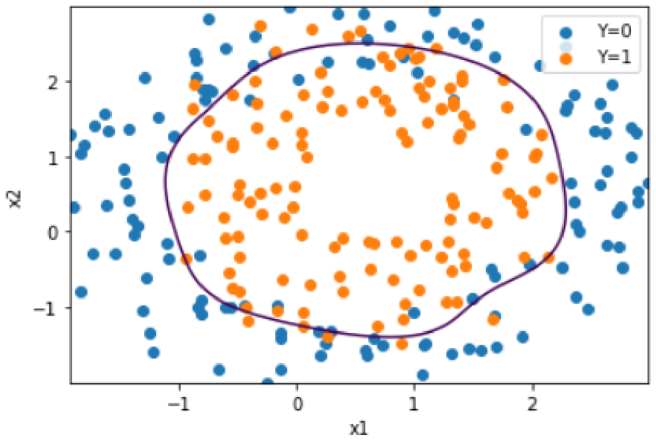

Fig. 2: Decision boundary with $\lambda = 1.0$.

d) (Programming problem) Compare the results of $\lambda = 0, 1, 100$, and 10000. How does changing λ affect the decision boundary? Why is the decision boundary affected in that way? Write your answer in the markdown cell provided.