Artificial Intelligence Spring 2018
Homework 3: Machine Learning
Due Date: Sunday, April 15 at 11:59pm

ACADEMIC HONESTY

As usual, the standard honor code and academic honesty policy applies. We run **automated checks for plagiarism** to ensure only original work is given credit. Submissions isomorphic to (1) those that exist anywhere online, (2) those submitted by classmates, or (3) those submitted by students in prior semesters, will be considered plagiarism.

SUBMISSION

You will submit one zip file, named **hw3_myUNI.zip**, which contains files:

- written.pdf,
- problem1.py or problem1_3.py
- problem2.py or problem2_3.py
- problem3.py or problem3_3.py
- README.txt or .pdf

You can submit as many times as you like before the deadline. Only your most recent submission will be graded. The written submission can be handwritten and scanned (PDF scanning mobile apps typically work well), or typed, but it must be legible in order to receive credit.

WRITTEN

I. Learning Methods

1. For the learning situation below, name the best learning algorithm to use, and briefly justify:
 "Predict the average rainfall in NYC using the past six months of data on the currents and tides in the Atlantic ocean."

2. The idea of boosting is to train weak learners on weighted training examples. Pick all that apply.
   a. Give large weights to easy examples to get rid of them
   b. Use any classifier as far as its accuracy is slightly worse than random
   c. The classification output is a weighted majority voting of all weak classifiers output

3. K-means Clustering. Pick all that apply.
   a. The basic K-means algorithm requires setting up the parameter K (number of clusters) a priori. [ ] In K-means, we assume that each cluster fits a Gaussian distribution (normal distribution).
   b. We can set K to optimally cluster the data by starting with a small number of clusters, and then iteratively splitting them until all clusters fit a normal distribution.
   c. A clustering is good if it has a high intra-cluster similarity and a low inter-cluster similarity.
   d. A clustering is good if it has a low intra-cluster similarity and a high inter-cluster similarity.
   e. A clustering is good if it has a low intra-cluster similarity and a low inter-cluster similarity.

II. Naive Bayes

We would like to predict boy/girl gender based on **the kids name and physical features**, with an emphasis on unisex names**.**

Name	Tall	Eye	Hair	Sex
Tyler	no	blue	short	boy
Salma	yes	brown	long	girl
Tyler	no	blue	long	girl
Tyler	no	blue	long	girl
John	yes	brown	short	boy
Leila	no	blue	long	girl
Alexandra	yes	brown	short	girl
Ali	yes	blue	long	boy

1. Build a naive Bayes classifier using simple probabilities (e.g. no m-estimate or smoothing).

2. What is the prediction for a " Tall kid named Tyler with long hair and brown eyes"? Justify briefly.

3. Based on our knowledge of frequencies in the larger population, we know the priors P (boy) = P (girl) ≈ 0.5. What are the priors based on the frequency in this training set? What can you say about this dataset? Explain.

III. Neural Networks

Construct a **one-hidden layer** neural network for the Boolean function below. Show all your work.

(X or not Y) XOR (not Z or not T)

[BONUS +3]
Consider a simple neural network model with one linear output neuron and no hidden layer. We want to take the average of two such networks. In other words, given input $\mathbf{x}$ and two networks with weights $\mathbf{w_1}, \mathbf{w_2}$ respectively, the output is:

$$y = \frac{1}{2}\mathbf{w_1}^T\mathbf{x} + \frac{1}{2}\mathbf{w_2}^T\mathbf{x}$$

Will this improve the performance of this model? Prove your claim.

PROGRAMMING

We ask you to implement three small and separate machine learning models. We provide sample input/output file pairs for your reference. The file `visualize.py` contains two functions for plotting data which you may edit at will.

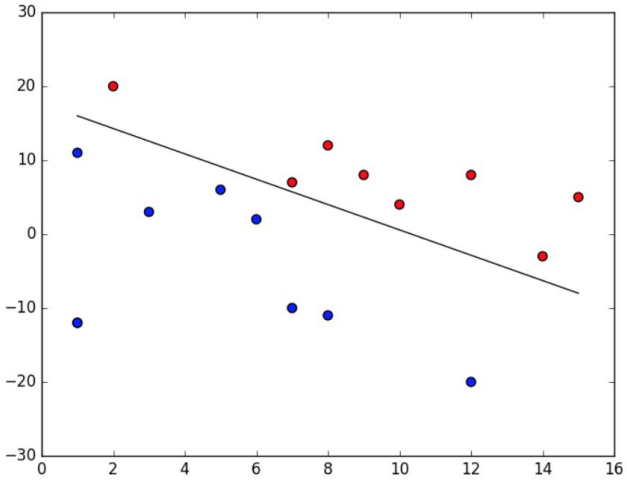
I. Perceptron
II. Linear Regression
III. Clustering

Note: The Python Pandas library helps simplify a lot of the intermediate steps we ask from you below. Key functions include reading and writing to csv's, matrix operations, and visualizing data.

I. Perceptron

Implement the perceptron learning algorithm (PLA) for a linearly separable dataset. Your starter code includes `input1.csv`, where each line contains `feature_1,feature_2,label`. All values are numeric with labels 1 or -1. Take a look at the data input file. We suggest using matplotlib or pandas.DataFrame.plot (in `visualize.py`) to view data, and pandas.DataFrame.describe to see stats.

Write your PLA in `problem1.py` (or `problem1_3.py` for Python 3). Your program takes a csv of input data and a location to write the output csv with the weights from each iteration of your PLA, written as `weight_1,weight_2,b`. The weights in the last line of your output csv defines the **decision boundary** computed for the given dataset. There's an example `output1.csv` in the starter. Feel free to visualize and include a screenshot of your final decision boundary in your README, like the one below:



We will execute your code as follows:

```
$ python problem1.py input1.csv output1.csv
```

II. Linear Regression

Use gradient descent to build a linear regression model for predicting height (m) using age (yr) and weight (kg), using data derived from CDC growth charts data.

Data Preparation and Normalization. Load and understand the data from `input2.csv`, remembering to add a vector column for the intercept at the front of your matrix. You'll notice the features are not on the same scale. What is the mean and standard deviation of each feature? Scale each feature (i.e. age and weight) by its standard deviation, and set its mean to zero. You do not need to scale the intercept. For each feature column, x, use the following formula:

$$x_{\text{scaled}} = \frac{x - \mu(x)}{stdev(x)}$$

Gradient Descent. Implement gradient descent to find a regression model in `problem2.py` (or `problem2_3.py` for Python 3). Initialize your β's to zero. Recall the empirical risk and gradient descent rule as follows:

$$R(\beta) = \frac{1}{2n}\sum_{i=0}^{n}(f(x_i) - y_i)^2$$

$$\forall j \qquad \beta_j := \beta_j - \alpha\frac{1}{n}\sum_{i=0}^{n}(f(x_i) - y_i)x_i$$

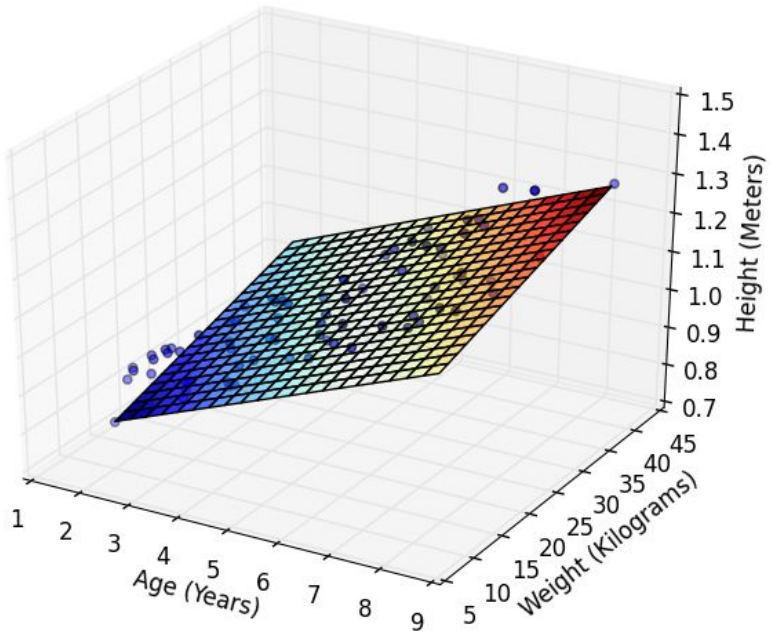
You will run gradient descent with these nine learning rates: α ∈ {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10}, **exactly 100 iterations per α-value**, plus a tenth rate and number of iterations of your choice. To pick the tenth rate and loop count, observe how α affects the rate of convergence for the nine rates listed, then pick a rate and loop count you believe will perform well. Briefly explain your choice in your README.

The program should generate an output file containing ten lines, one for each (α, num_iters) hyperparameter pair. Each line contains: `α,num_iters,bias,b_age,b_weight`, expressed to your choice of decimal places (see example `output2.csv` in your starter code). Each line of this file defines the regression model your gradient descent method computed on the given data and hyperparameters.

We will execute your code as follows:

```
$ python problem2.py input2.csv output2.csv
```

Optional. Visualize the result of each linear regression model in three-dimensional space. You can plot each feature on the xy-plane, and plot the regression equation as a plane in xyz-space. Ex:



III. Clustering

Try the following two clustering algorithms on the given image `trees.png`. The second method is an optional bonus portion. We will segment and group the pixels according to their RGB values. You can read about image segmentation here: https://en.wikipedia.org/wiki/Image_segmentation.

1. K-means: For this part you will experiment with the k-means algorithm. Choose 3 representative k values and use sklearn.cluster. KMeans to process the image file. Thoroughly record (screenshots, short annotations) and explain your results in your README.
2. **[BONUS +12]** Spectral Clustering
   Choose a dataset on which spectral clustering works better than K-means. Thoroughly record the results of spectral clustering. Compare the results of both methods and offer an explanation in your README.

IV. Before You Submit

- **Make sure** your code executes. In particular, make sure you name your file correctly according to the instructions specified above, especially regarding different Python versions.
- **Make sure** your program does not print anything to the screen.