ASSIGNMENT 2

(100 POINTS) DUE BY WEDNESDAY, MARCH 25, 2020, 2 PM

Assignment Policy:

- No late submission is allowed.
- Collaboration is okay, however, students must submit assignment in their own words. Plagiarism in any form will not be accepted.
- For the coding aspect of the assignment, students have to submit their individual .ipynb files.

Question 1 (SMOOTH FUNCTIONS - 10 POINTS)**.** A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz. Show that smoothness implies that for all $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$, we have:

$$f(\boldsymbol{v}) \leq f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle + \frac{\beta}{2}\|\boldsymbol{v} - \boldsymbol{w}\|^2$$

Question 2 (GRADIENT DESCENT FOR SMOOTH FUNCTIONS - 30 POINTS)**.** Show that if f is a β-smooth function, then the gradient descent with step size $\eta = \frac{1}{2\beta}$ when run for T time steps has the following guarantee:

$$f(\boldsymbol{w}_{avg}) - f(\boldsymbol{w}^*) \leq \frac{C\beta\|\boldsymbol{w}^{(1)} - \boldsymbol{w}^*\|^2}{T},$$

where $\boldsymbol{w}_{avg} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{w}^{(t)}$, C is some constant, and $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} f(\boldsymbol{w})$.

Question 3 (GRADIENT DESCENT IN LINEAR REGRESSION ON SYNTHETIC DATA - 30 POINTS)**.** Generate data as $Y = \theta_0 + \theta_1 * X + \epsilon$, where, $\theta_0 = 4$, $\theta_1 = 3$, $X \sim \mathcal{N}(0, 4)$ and $\epsilon \sim \mathcal{N}(0, 1)$.

(1) Code up the Ordinary least squares solution and show the data and the prediction on the same graph.

```
def  cal_cost(theta,X,y):
    '''

    Calculates the cost for given X and Y. The following shows and example of a single dimensional
X
    theta = Vector of thetas
    X     = Row of X's np.zeros((2,j))
    y     = Actual y's np.zeros((2,1))

    where:
        j is the no of features
    '''

    m = len(y)

    predictions = X.dot(theta)
    cost = (1/2*m) * np.sum(np.square(predictions-y))
    return cost
```

FIGURE 1

```
def gradient_descent(X,y,theta,learning_rate=0.01,iterations=100):
    '''
    X    = Matrix of X with added bias units
    y    = Vector of Y
    theta=Vector of thetas np.random.randn(j,1)
    learning_rate
    iterations = no of iterations

    Returns the final theta vector and array of cost history over no of iterations
    '''
    m = len(y)
    cost_history = np.zeros(iterations)
    theta_history = np.zeros((iterations,2))
    for it in range(iterations):

        prediction = np.dot(X,theta)

        theta = theta -(1/m)*learning_rate*( X.T.dot((prediction - y)))
        theta_history[it,:] =theta.T
        cost_history[it]  = cal_cost(theta,X,y)

    return theta, cost_history, theta_history
```

FIGURE 2

```
import pandas as pd
import numpy as np
df = pd.read_csv('https://archive.ics.uci.edu/ml/'
        'machine-learning-databases/iris/iris.data', header=None)

# select setosa and versicolor
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', -1, 1)

# extract sepal length and petal length
X = df.iloc[0:100, [0, 2]].values

# Split x and y (feature and target)
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    test_size=0.2)

from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(10), solver='sgd', learning_rate_init=0.01, max_iter=500,random_state=113)

# Train the model
mlp.fit(X_train, y_train)
```

FIGURE 3

(2) Use the definitions of functions provided in Figure 1 and Figure 2 to plot results for two learning rates (one low and one high) for different number of iterations. You should show the final learner in the backdrop of the data. Also show how the error reduces with the iterations.
(3) Use the provided functions to code up and show similar plots for Stochastic Gradient Descent and Mini-batch Gradient Descent.

Explain your observations.

Question 4 (GRADIENT DESCENT STRATEGIES FOR CLASSIFYING IRIS DATA - 30 POINTS)**.** In this example we use the iris data. Use the pseudocode given in Figure 3 to train a Multilayer Perceptron Classifier (with 10 hidden layers).

- Find the training and the test loss of the classifier.
- Plot the convergence in training with the number of iterations.

Use different values of learning rates, iterations and gradient solvers (as far as you can explore). You can use the properties of the MLPClassifier class of scikit-learn. Explain your observations.