AI HW #4

Ali Parvizi 9632433

This assignment is done in python 3.7. Install the requirements first

```
pip install -r requirements.py
```

Loader

The `loader.py` file, includes a loader function that reads a dataset file and converts its samples into Sample objects.

the loader function returns an array of normalized objects.

Part A: Linear Regression

Part A of the assignment is implemented in `partA.py` file. in this file, the two dataset files are loaded through the loader function from `loader.py`.

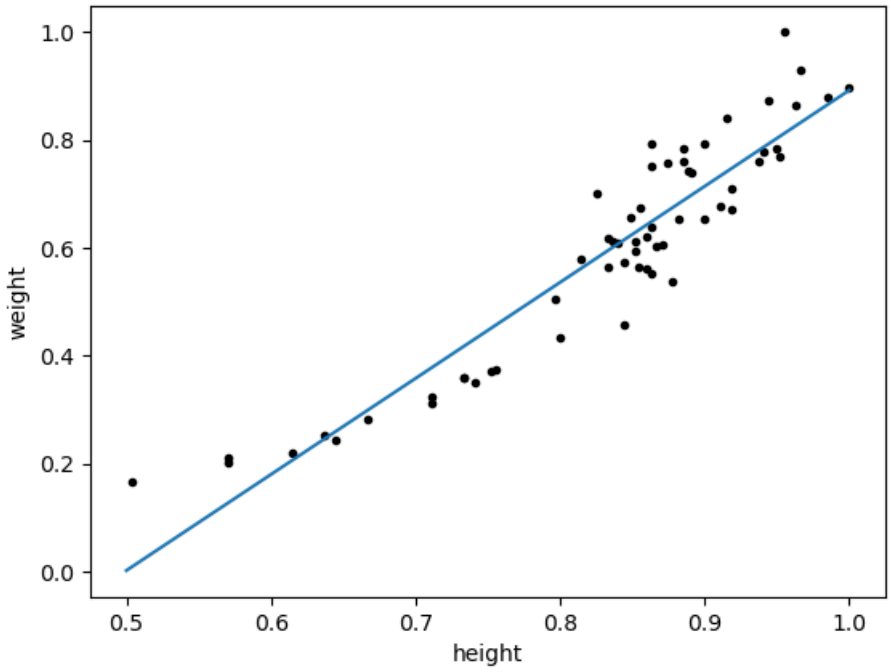
Two important functions in this file are: `gradient_descent` and `closed_form`. which correspond to gradient descent and closed form linear regression methods respectively.

How to Run

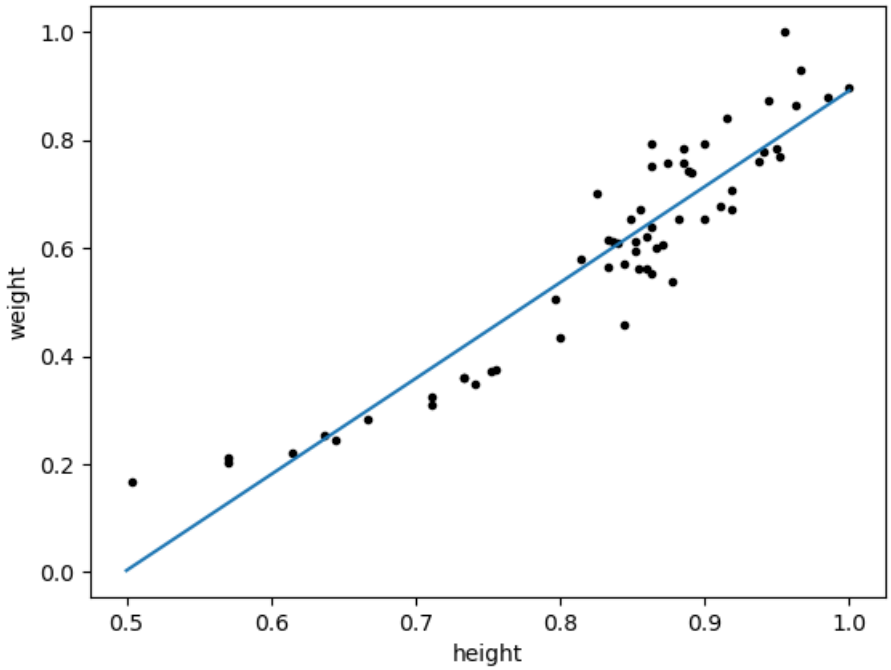
- run `python partA.py`

What it does

- by running the above command, partA.py will start executing. 4 plots will be generated one after another. The first plot shows all the normalized data from dataset 1 and a line that has been learned through gradient decent. After you close this plot, the second plot will show up. which is a plot of all the normalized data from dataset 1 and a line that has been learned through closed form equations.
- This is how normalization is done: value / (max_value - min_value) with the above formulation every sample data is normalized. If we don't normalize the data, different features with different scales and units may cause abnormalities in the final result. for example if one feature is measured in meters and another feature is measured in kilometers and the value of the first one is 0.01 and the the value of the second one is 800, a 25 percent increase in each of them leads to the first value being 0.0125 and the second value being 1000, this means that the change in the second value dominates the behaviour of the whole system.

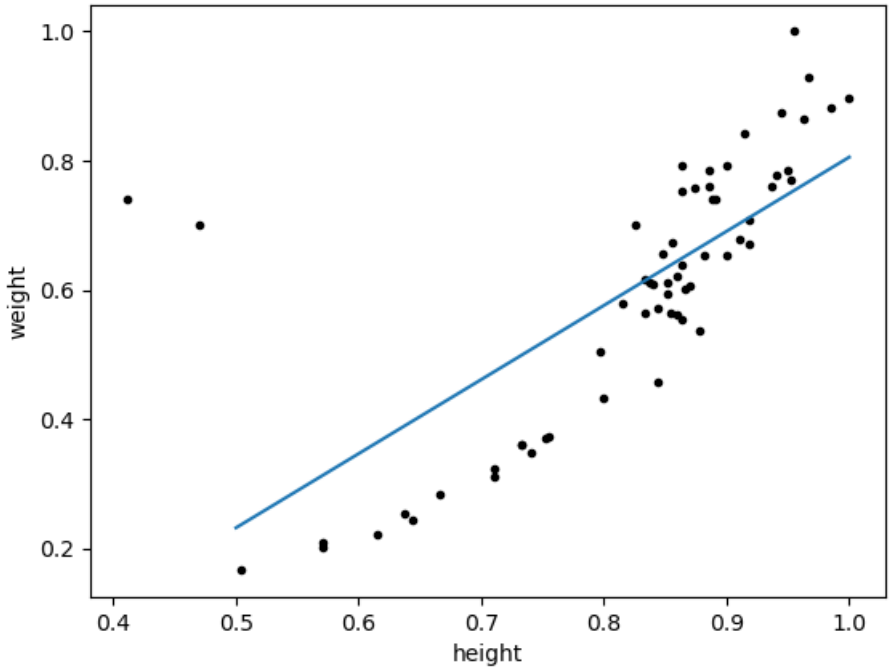


Data set 1 - Closed Form

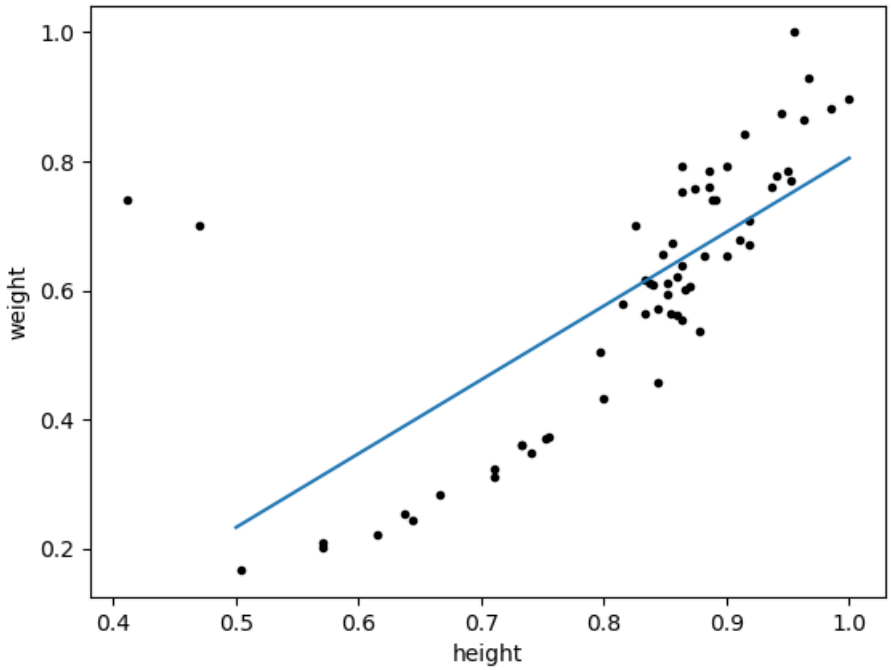


Data set 1 - Gradient Descent

- The 3rd and 4th plots are the same as the first two plots but with the data from dataset 2.

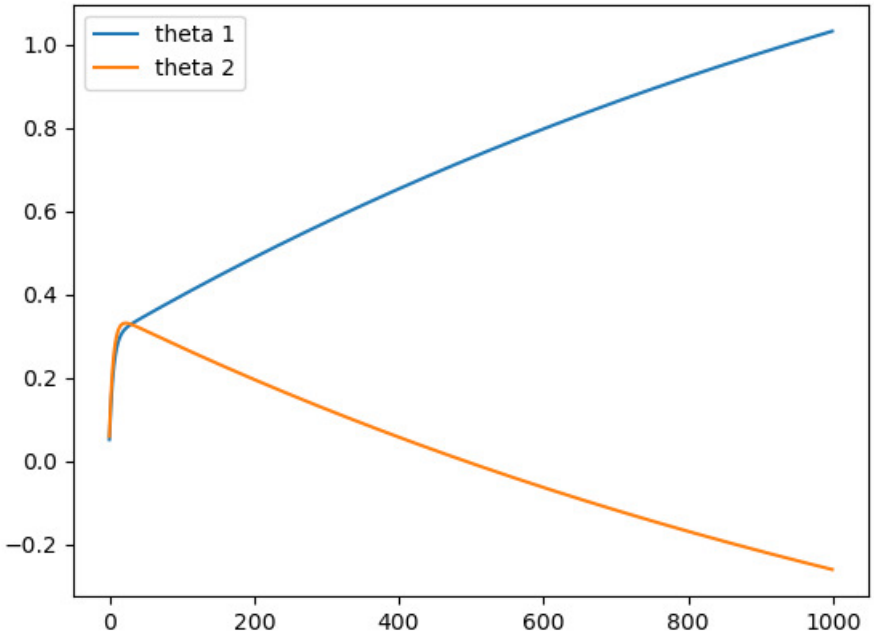


Data set 2 - Closed Form



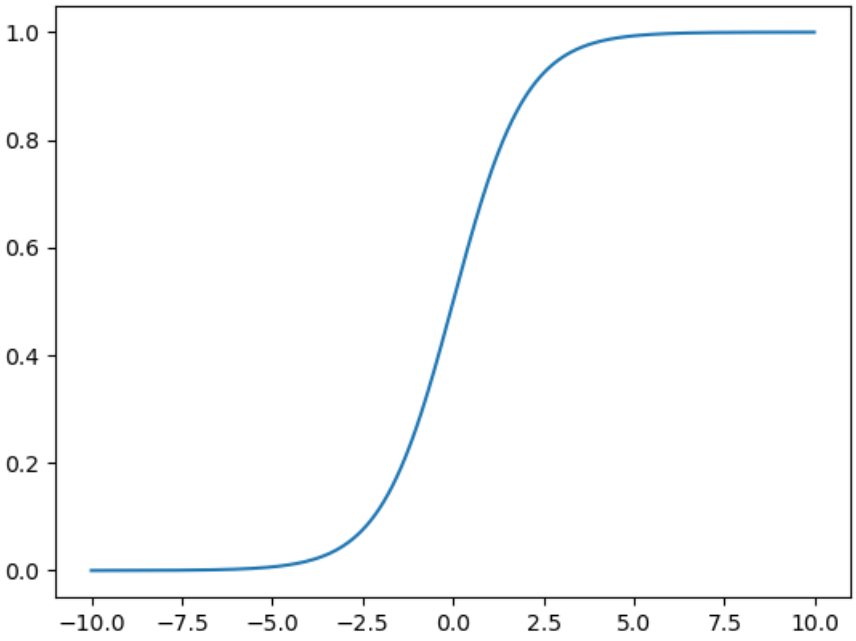
Data set 2 - Gradient Descent

- dataset 2 has some outliers. These outliers cause the learned model to be a bit off and the accuracy of the model decreases because it's trying to account for all of the data including the outliers. In other words the Least Square Objective function tries to minimize the error with respect to all of the samples. now if some of the samples are far from the other samples, the learned method would be less accurate for the majority of the samples.
- the 5th plot shows theta 1 and theta 2 as they were learned by gradient decent method



Theta learn

- the last plot shows a binary sigmoid function. A binary sigmoid function is used in classification problems. if the output of this function is more than 0.5, the input is considered to be of a class represented by 1 and if it's less than 0.5, it's considered to be of class 0



Theta learn

Part B: Weighted Linear Regression

In locally weighted linear regression, instead of all data having the same importance, the closer a sample is to a new unknown sample the more important it is. we could use a gaussian like weight function.

- $w^i = \exp(-(X^i - X^{new})^2 / 2)$

according to the above weight function, the closer X^i is to X^{new} the more weight it has thus it has more effect. Now we will modify the normal linear regression cost function to include the weights.

- $J(\theta) = \Sigma\, w^{(i)} (y^{(i)} - (\theta_0 + \theta_1 x^{(i)}))^2$

Now we need to find argmin(θ).

- $\partial J / \partial \theta_0 = -2 \Sigma\, w^{(i)} (y^{(i)} - (\theta_0 + \theta_1 x^{(i)}))$
- $\partial J / \partial \theta_1 = -2 \Sigma\, w^{(i)} (y^{(i)} - (\theta_0 + \theta_1 x^{(i)}))\, x^{(i)}$

Equating the above formulas to zero, we will have:

- $\Sigma\, w^{(i)}\theta_0 + \Sigma w^{(i)}\, \theta_1 x^{(i)} = \Sigma w^{(i)} y^{(i)}$
- $\Sigma\, w^{(i)}\theta_0 + \Sigma w^{(i)}\, \theta_1 x^{(i)} x^{(i)} = \Sigma w^{(i)} y^{(i)} x^{(i)}$

it can be rewritten as:

$$\begin{bmatrix} \sum w^{(i)} & \sum w^{(i)} x^{(i)} \\ \sum w^{(i)} x^{(i)} & \sum w^{(i)} x^{(i)} x^{(i)} \end{bmatrix} \begin{bmatrix} \Theta 0 \\ \Theta 1 \end{bmatrix} = \begin{bmatrix} \sum w^{(i)} y^{(i)} \\ \sum w^{(i)} y^{(i)} x^{(i)} \end{bmatrix}$$

in other words: $A\theta = b$

now we can solve for θ and arrive at a closed form formula: $\theta = A^{-1}b$

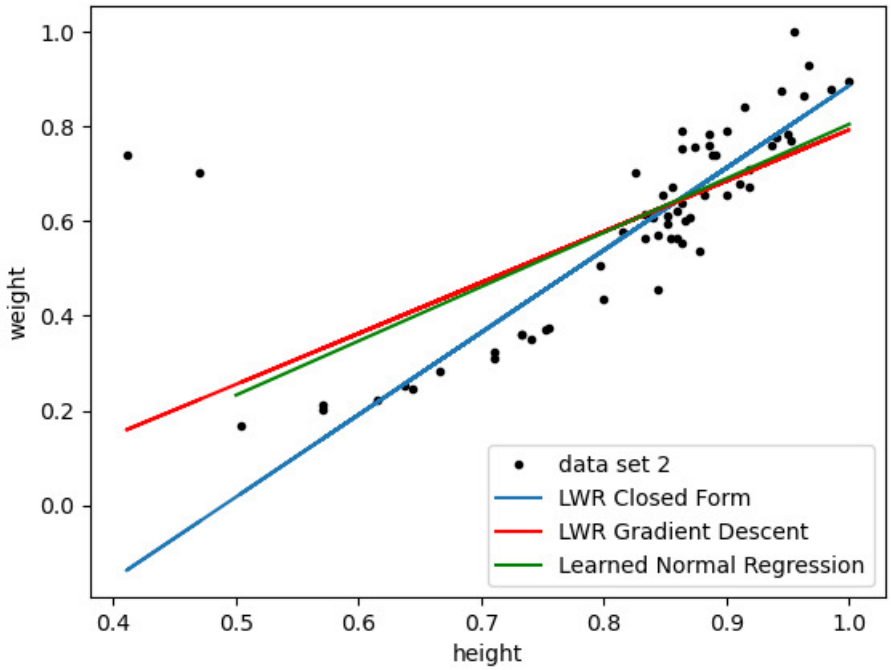
How to Run

run `python partB.py`

What it does

Computes a linear model for the second data set which includes some outliers using LWR. the resulting model and the data set samples along with a model from normal regression are generated in a figure.

In the figure below you can see the effect of locally weighted regression. in LWR the model is more accurate even when there are some outliers in the sample data, because their effect on farther samples are less.



Part C: Binary Classification

This is the cost function that is used through out the training phase:

- $J(\theta) = - 1/m \Sigma [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))]$

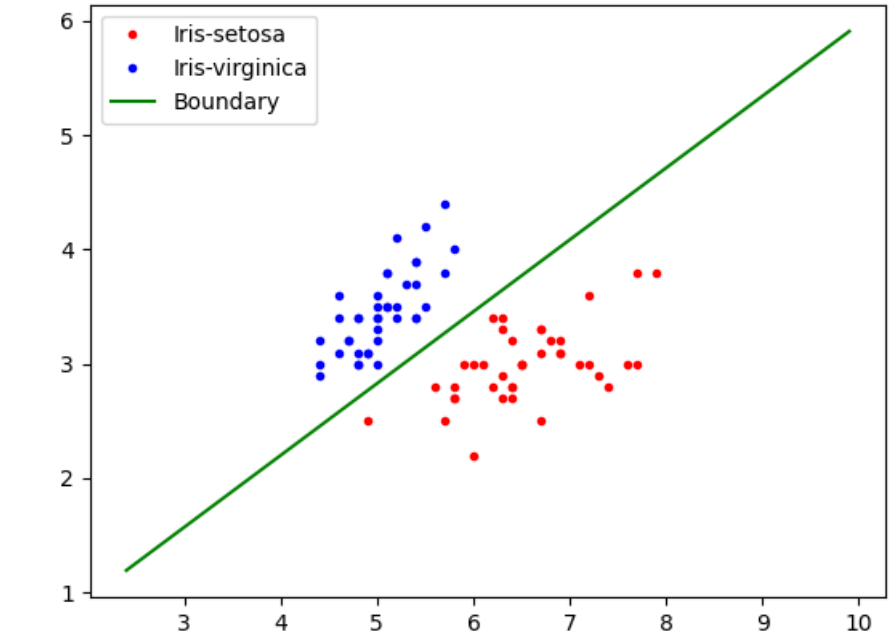
80 percent of the iris.data data set is randomly chosen for training, and the remaining 20 percent are selected for testing.

after training a model on the train data, we measure the accuracy of the model by the following formula:

- accuracy = hit / N

where hit is the total number of test cases that the model predicted correctly, and N is the number of total test cases.

The accuracy of the model is measured both with the test data and the training data.



How to run

run `python partC.py`

What it does

first a random set of training and testing data will be automatically generated.

Logistic Regression method with a sigmoid function is used to classify the samples and the threshold for classification is set to 0.5.