William Bullock

Introduction to Data Science

Assignment Five – Final Assignment – Answer Sheet

(Please see companion .py file for the code used in this assignment)

Exercise 1:

for reference:
$t = f(x,w) = w0 + w1x1 + w2x2 + \ldots + wDxD = wTx,$

a)

Please see code

b)

Outputted W values for just the fixed acidity:

[5.2067 0.0504]

W0 is the offset parameter, and is the first parameter shown here.

This is analogous to the Y-intercept, if all other values of w and x were equal to 0, the line would intersect the y axis at ~5.21

W1 is the fixed acidity parameter, and is the second parameter shown here. Will act to raise the t value by the numerical value of fixed acidity multiplied by the weight (here; ~0.05).

c)

Outputted W values for all input variables:

[5.166+01 1.959e-02 -1.062+00 2.589-02
  5.023e-02 -2.755e+00 5.653e-03 -3.807e-03
 -4.721e+01 -4.266e-01 8.505e-01 2.379e-01]

Parameters contribute to the final t value, with a strength according to their distance from 0. Parameters with a negative value will contribute to lower the t value, whereas parameters with a positive value will contribute to raise their t value.

We can see that the offset parameter is ~ 51.66, this sets an initial value for t which is then is adjusted as aforementioned by each of the features according to their value multiplied by their weight.

From the spread of distributions seen here, density has a strong negative weight, that will significantly lower the t value as density increases, in contrast; sulphate content tends to contribute slightly towards raising the t value, whereas other parameters elicit even subtler changes.

We can also see that the values for the offset parameter and the fixed acid parameter have changed from what they were in **(b),** this is adjustment accounts for the weight and variance in the other parameters, and their ability to affect t value.

Exercise 2:

a)

Please see code

b)

0.7861

c)

0.6447

Root mean squared error is lower when the entirety of the parameters included are available, the algorithm has acquired a better fit.

In the latter case; When choosing a t value, the estimated t value (between 0 and 10) the estimation differs from the true value by ~0.645 specifically.

Both of these values are above an error rate of 5%, which is sometimes used as an arbitrary cut off rate.

Exercise 3:

A decision tree is a flowchart-like structure that is designed for data mining, with the goal of predicts the value of a target variable, based on several input variables. For example, being able to predict y given x1, x2, … xn.

Classification trees are one such variety of decision tree that is presented with a finite number of options for the target variable (y), known as classes. When one applies a classification tree to a dataset; the goal is to discover to which class the presented data belongs.

Typically this is achieved by partitioning the dataset based on input variables, whereby the values of input variables are each independently assessed and different thresholds of each variable are deemed to contribute towards certain classes (usually by using a 'greedy algorithm') until a suitable tree is constructed.

At this point new input variables can be introduced to the tree algorithm and classes will be assigned, based on how the model was fitted to the training data. The tree can assert if x1, x2, … xn are each, individually above some threshold, the truth or falsehood of this statement acting to suggest a particular class. The input variables are moved through as nodes, in some order, which need not be fixed.

A random forest classifier is a collection of these classification trees, in a form of bootstrap aggregating (bagging) so that the dataset has a great many iterations of the classification tree process applied to it, the partitioning step in each of these trees is completed by randomly splitting the data based on its input variables. When test data is introduced to the algorithm; each tree in the forest predicts a class, and the 'votes' are counted, the class is then determined as the most frequently chosen class.

Normalization (centering to a zero mean, and removing variance) has no effect on a random forest classifier. As the partitions are drawn for each classification tree are based by looking at input variables independently and their contribution towards a class, rather than their co-variance or correlation with other input variables, or any other associated features. As such a classification tree should give largely similar results regardless of how/if pre-processing of the dataset has occurred; and by extension so should a random forest classifier.

Exercise 4:

Prediction accuracy is variable but tends to be ~0.965 – 0.970, five iterations give:

- 0.9669
- 0.9721
- 0.9652
- 0.9652
- 0.9669

Exercise 5:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[\mathrm{e}^{-\frac{x}{2}}+10x^2\right]$$

$$=\frac{\mathrm{d}}{\mathrm{d}x}\left[\mathrm{e}^{-\frac{x}{2}}\right]+10\cdot\frac{\mathrm{d}}{\mathrm{d}x}\left[x^2\right]$$

$$=\mathrm{e}^{-\frac{x}{2}}\cdot\frac{\mathrm{d}}{\mathrm{d}x}\left[-\frac{x}{2}\right]+10\cdot 2x$$

$$=\left(-\frac{1}{2}\cdot\frac{\mathrm{d}}{\mathrm{d}x}\left[x\right]\right)\mathrm{e}^{-\frac{x}{2}}+20x$$

$$=20x-\frac{1\mathrm{e}^{-\frac{x}{2}}}{2}$$

$$=20x-\frac{\mathrm{e}^{-\frac{x}{2}}}{2}$$

Derivation process shown above.

Learning rate = 0.1)

At 10000 iterations;

Number of iterations taken to converge was 10001 (so convergence failed)

The final function value was ‘nan’

Learning rate = 0.01)

Number of iterations taken to converge was 96

The final function value was ‘0.024693232626518329’

Learning rate = 0.001)

Number of iterations taken to converge was 935

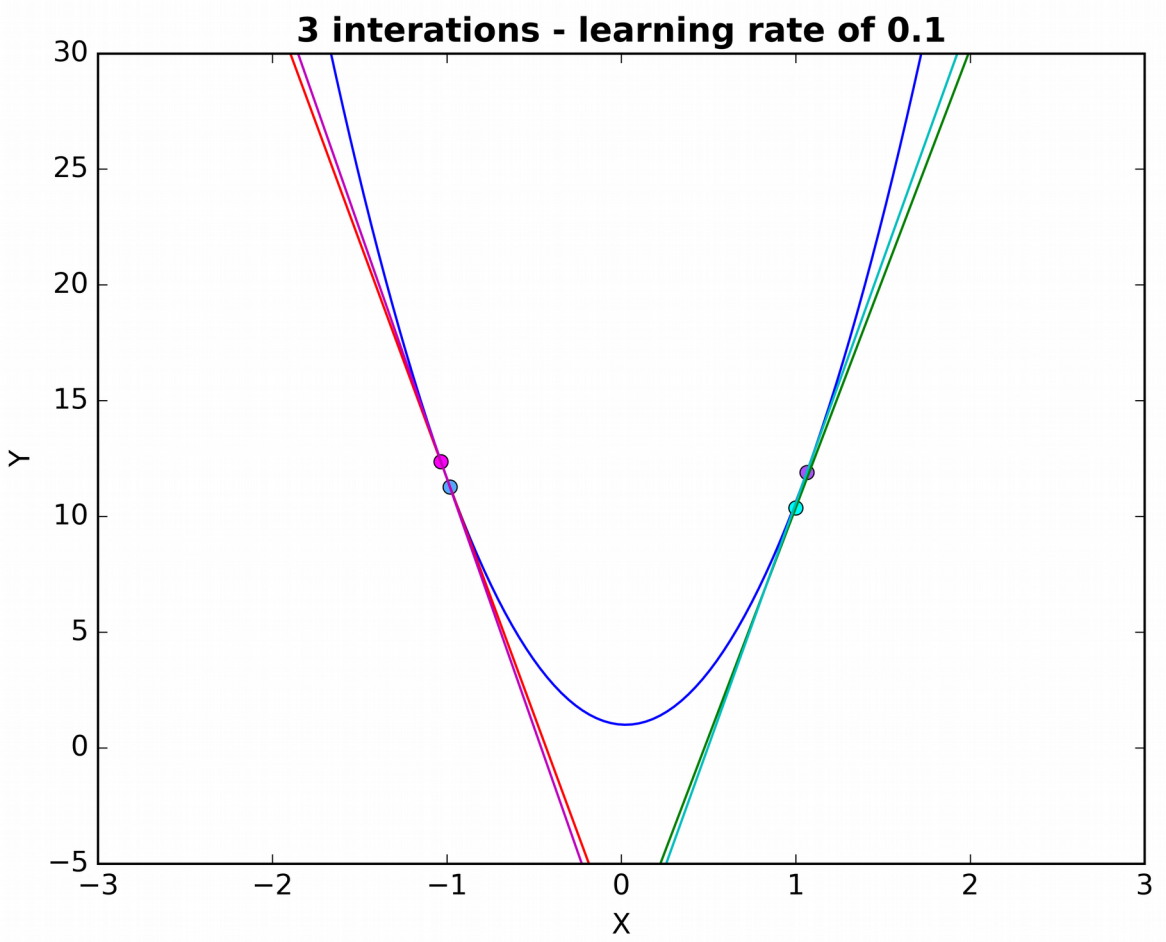
The final function value was ‘0.024693237099916006’

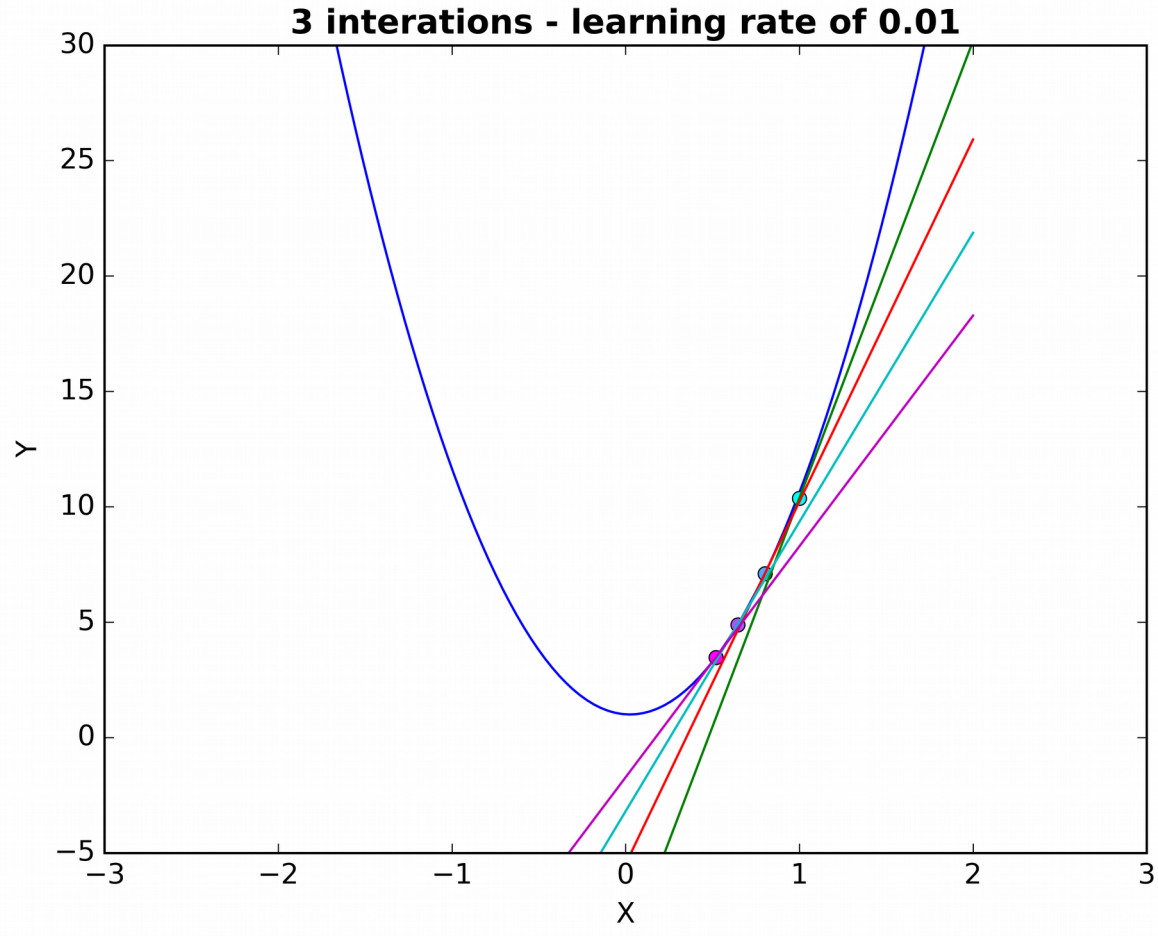
Learning rate = 0.0001)

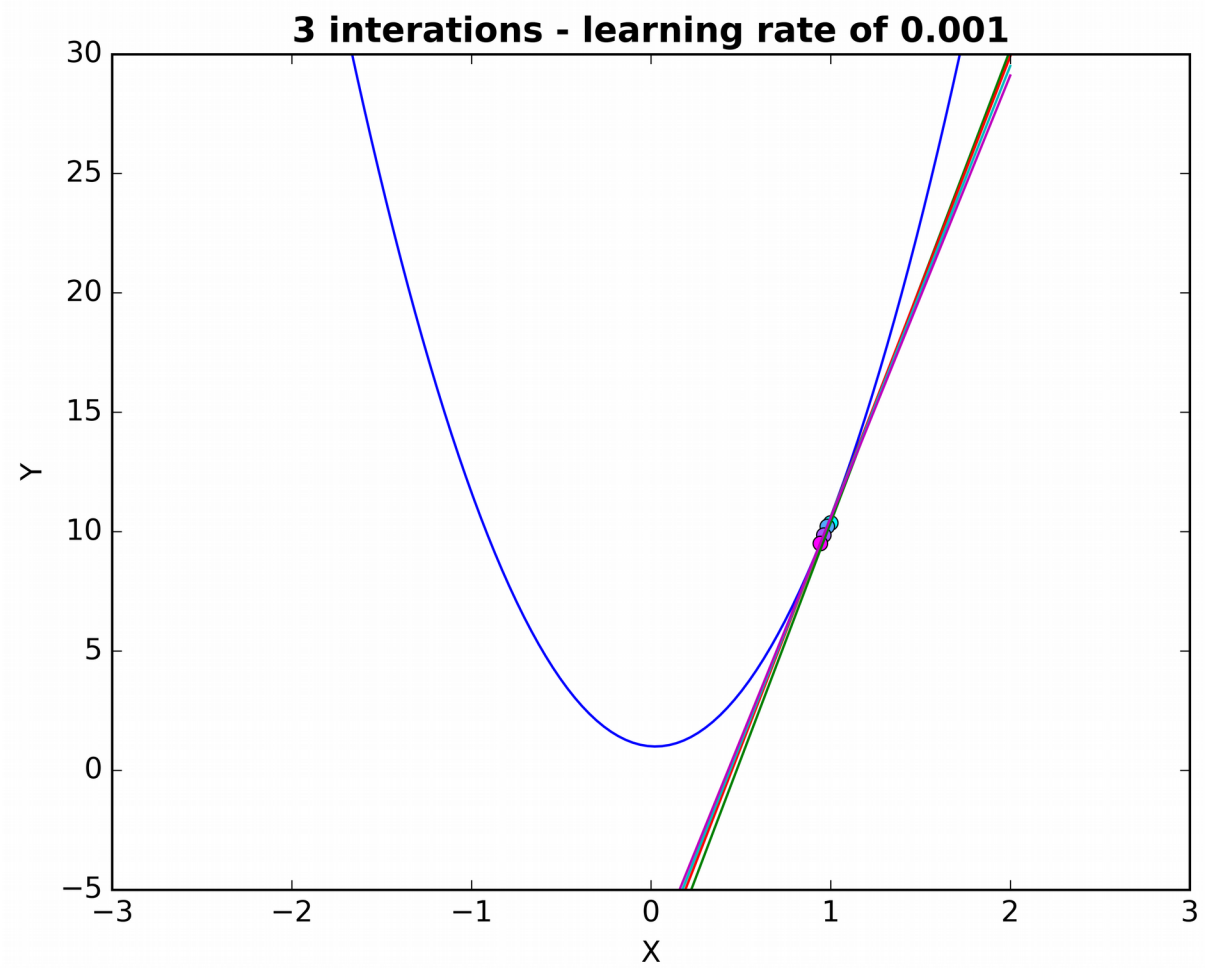
Number of iterations taken to converge was 8291

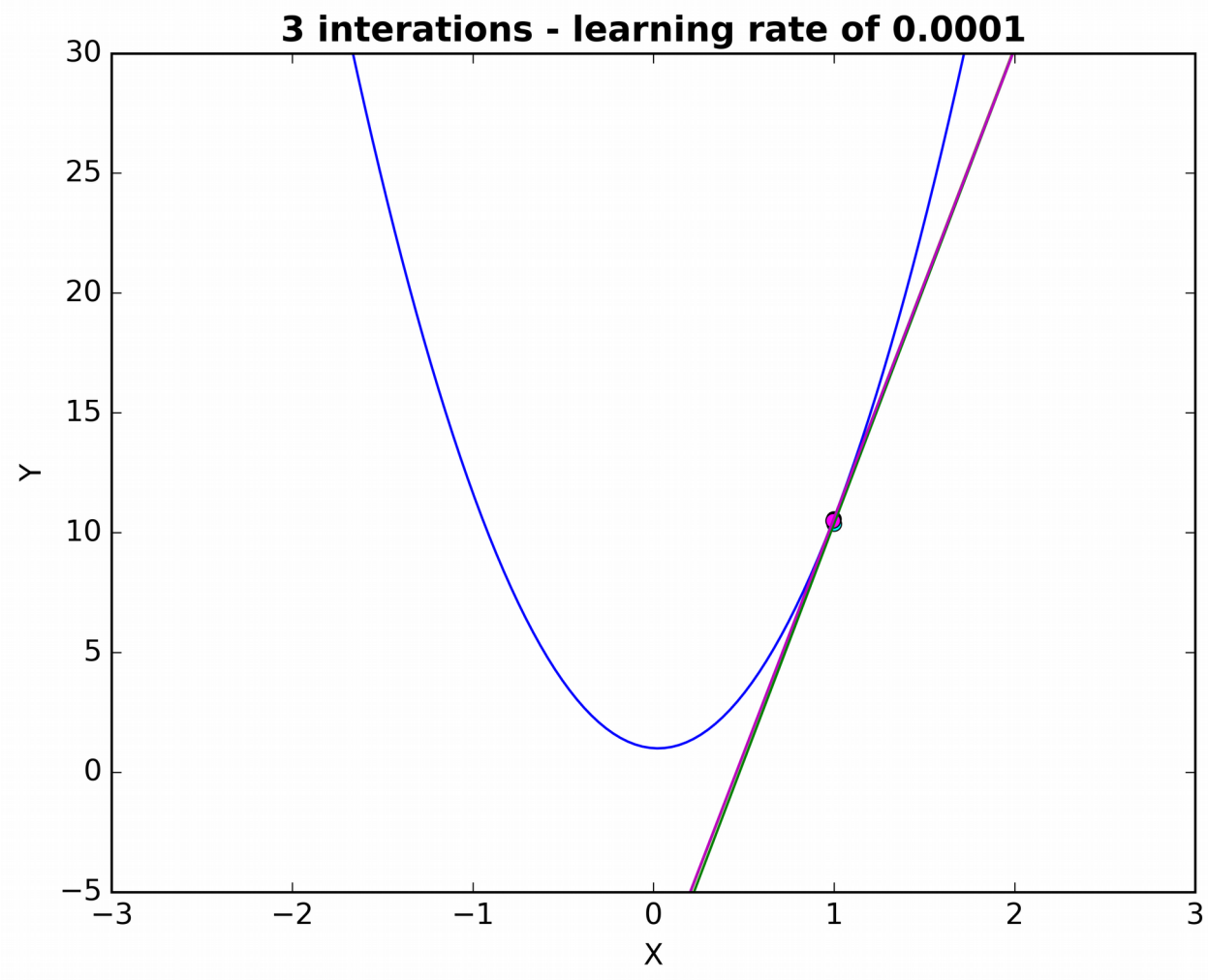
The final function value was ‘0.024693281532673802’

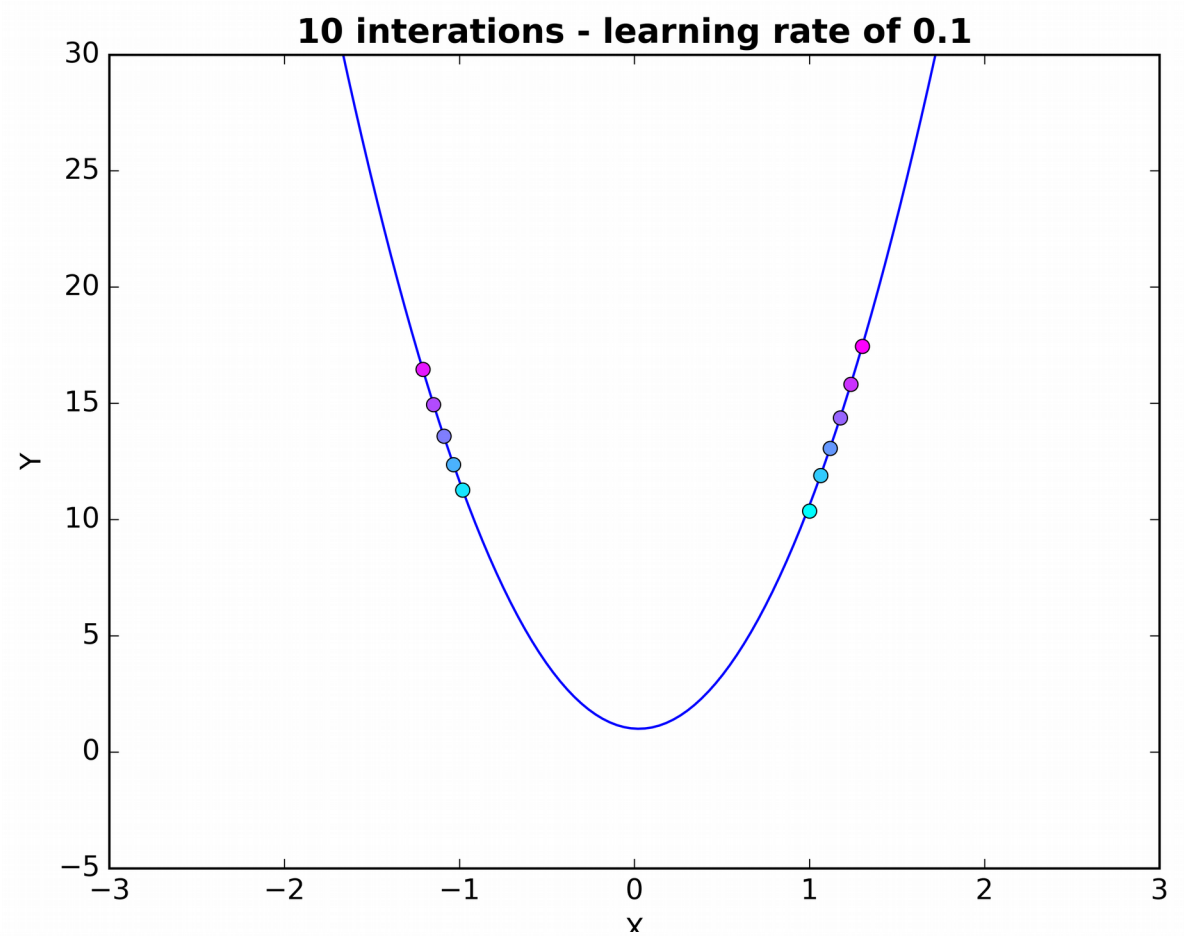
Plots:

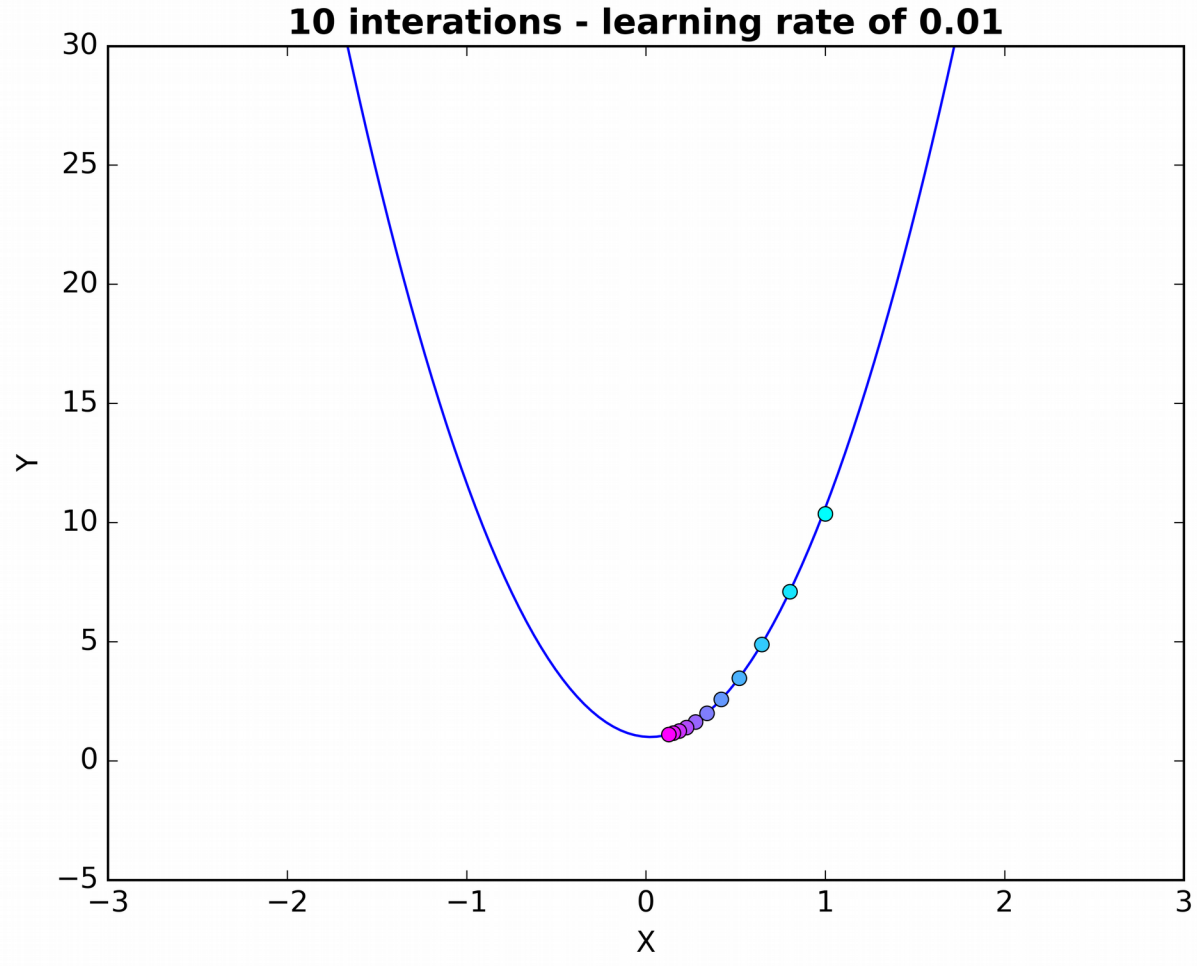

3 interations - learning rate of 0.1
30
25
20
15
10
5
0
−5
−3
−2
−1
0
1
2
3
Y
X

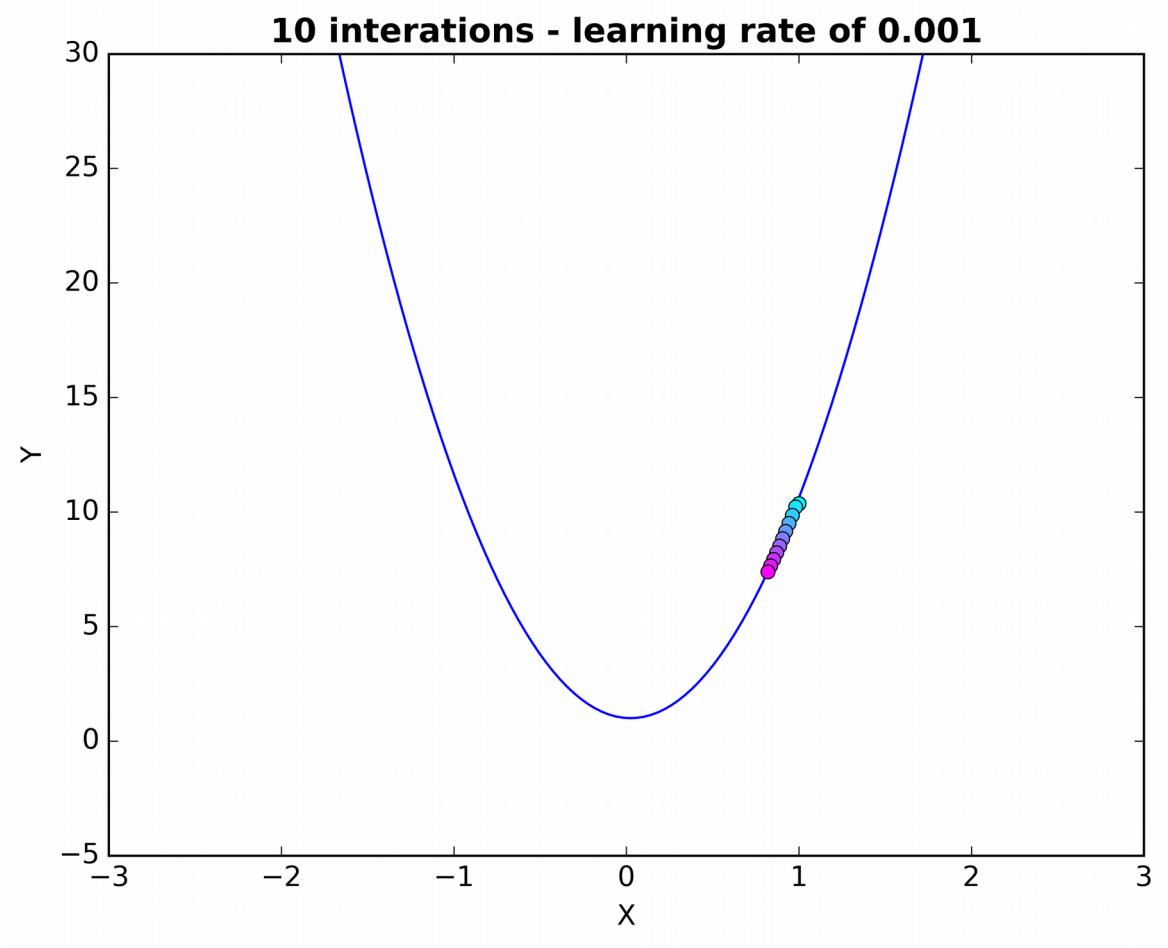

3 interations - learning rate of 0.01
30
25
20
15
10
5
0
−5
−3
−2
−1
0
1
2
3
Y
X

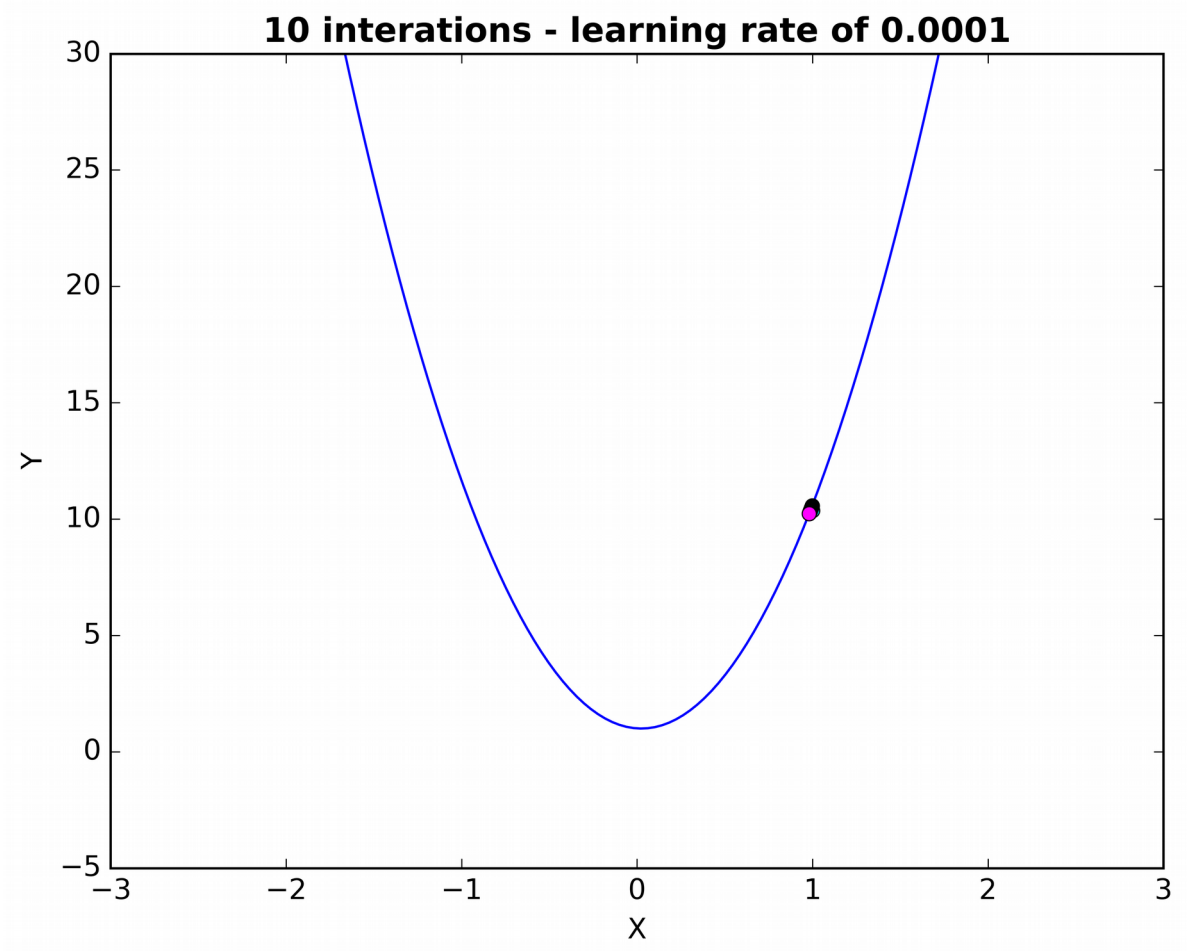

3 interations - learning rate of 0.001
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X


3 interations - learning rate of 0.0001
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X


10 interations - learning rate of 0.1
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X


10 interations - learning rate of 0.01
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X


10 interations - learning rate of 0.001
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X


10 interations - learning rate of 0.0001
30
25
20
15
10
5
0
−5
Y
−3
−2
−1
0
1
2
3
X

Discussion:

All learning rates except 0.1 reach convergence eventually, given enough iterations. 0.1 appears to be such a large step size that it constantly steps over the minimum value of the function, never approaching it.

0.01 reaches convergence first, with the lowest number of iterations, the other learning rates take much longer, but give a more accurate result due to smaller step sizes.

0.01 is ultimately the best learning rate to choose as it gives a very accurate minimum value using 10/100 times fewer iterations, and thus much less time / processing power. The other working learning rates do not seem to give greater accuracy appropriate to their operational cost.

Exercise 6:

Iris2D1

Trained on training data, tested on train data.
Parameters:
[-28.96078791 -4.20033394 11.24331337]
0 – 1 loss error:
0.0428571428571

Trained on training data, tested on test data.
Parameters:
[-28.95974596 -4.20020127 11.24315384]
0 – 1 loss error:
0.1

Iris2D2

Trained on training data, tested on train data.
Parameters:
[-113.52306173   42.4338969   -36.46457313]
0 – 1 loss error:
0.0

Trained on training data, tested on test data.

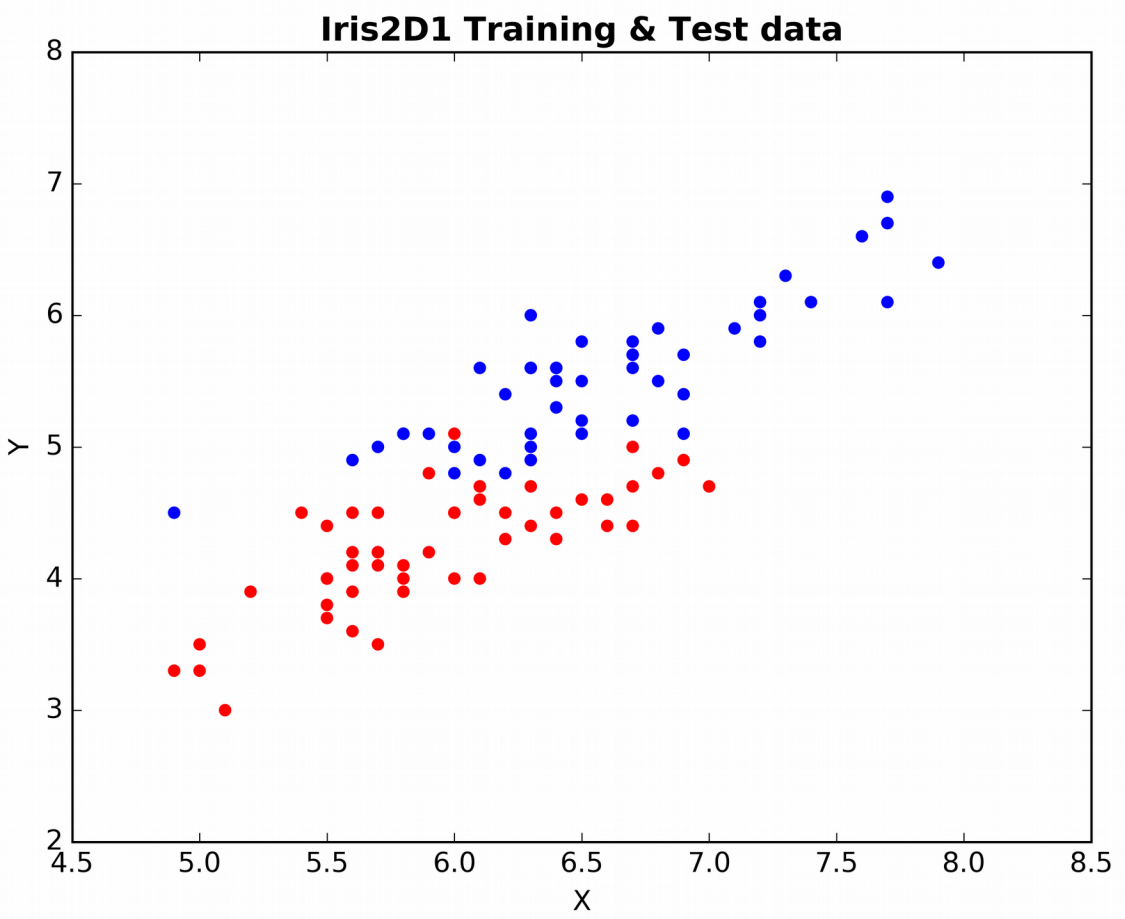
Parameters:
[-113.58052248   42.45451221 -36.48153456]
0 – 1 loss error:
0.0

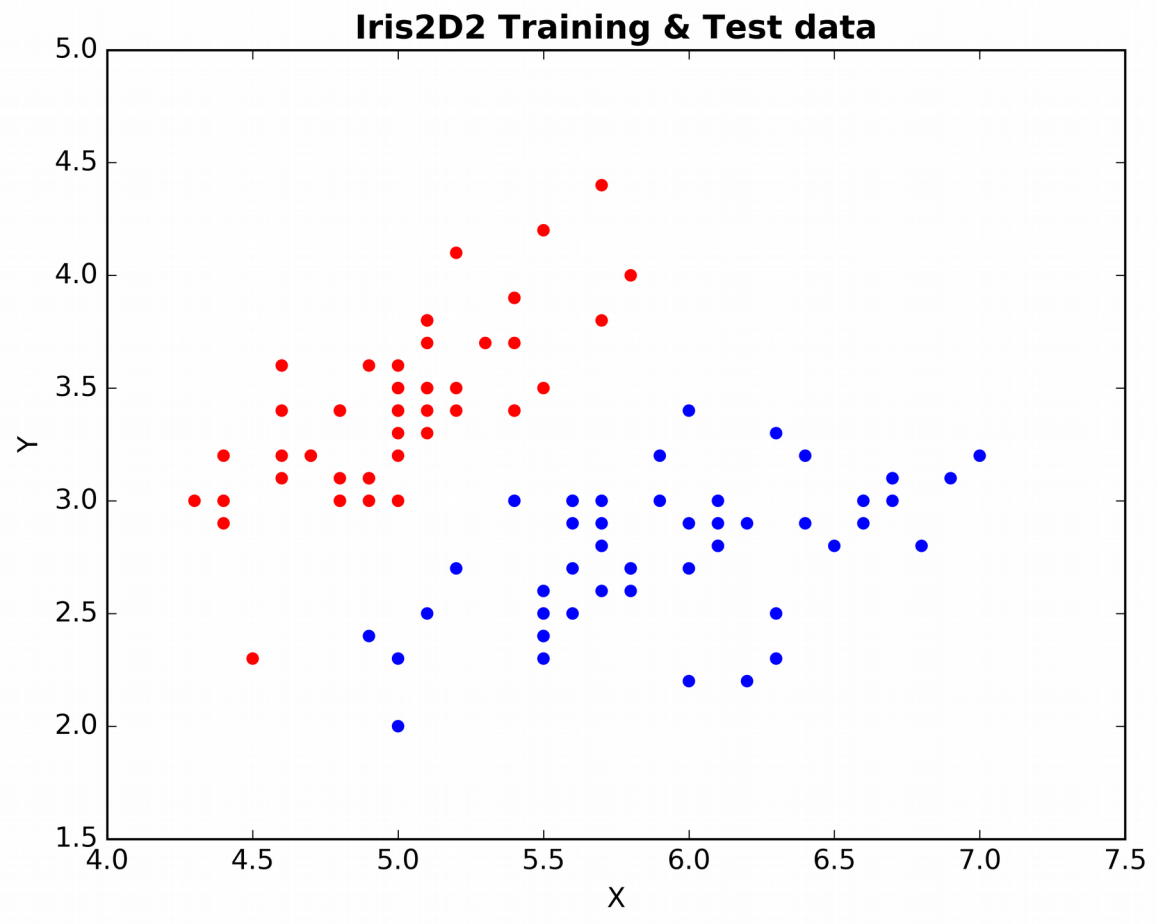
Discussion

Prediction is perfect for both instances of Iris2D2, this is because the two classes are well and prominently defined within this dataset, this is also illustrated by the plots.

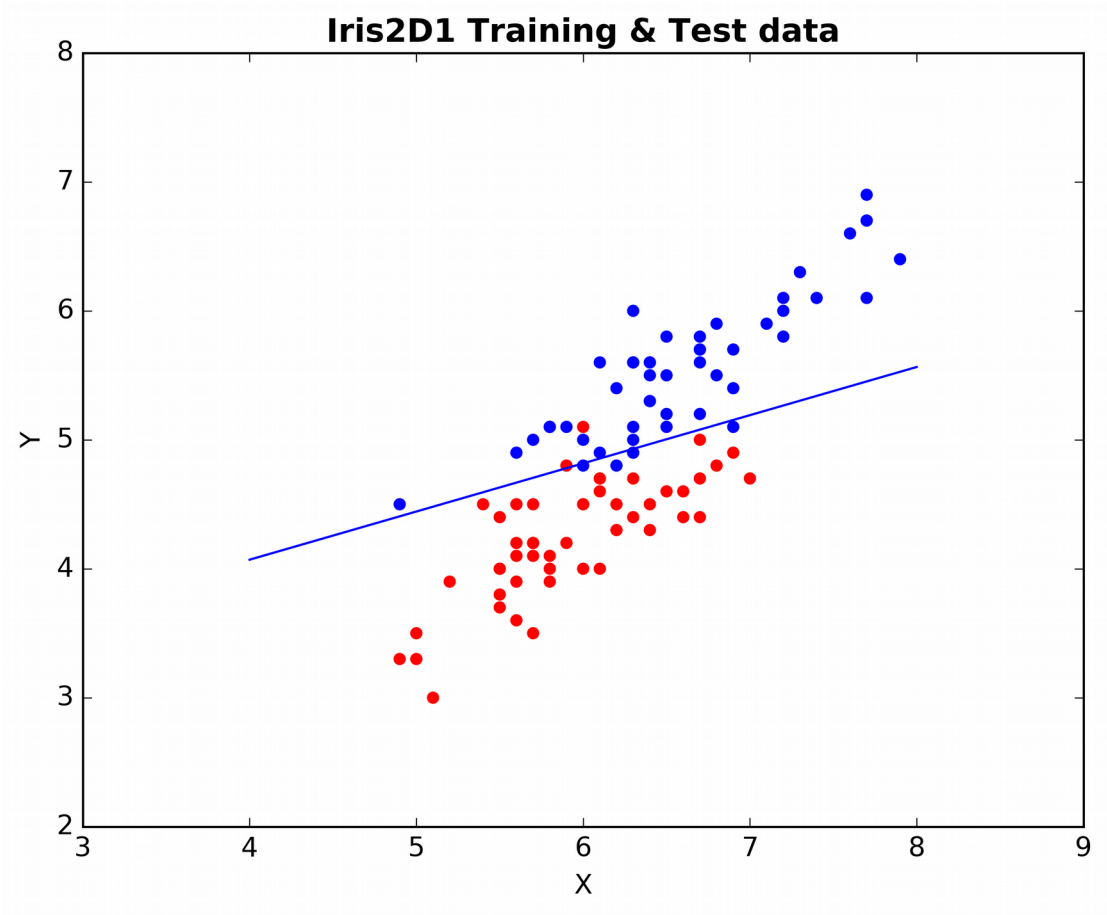
Prediction is imperfect for Iris2D1 even when testing on the same data that the algorithm is trained from, this is because the classes have a certain degree of overlap, this too is visible in the following plots.

Preliminary Plots:





Bonus Plots:


Iris2D1 Training & Test data
Y
X
8
7
6
5
4
3
2
3
4
5
6
7
8
9

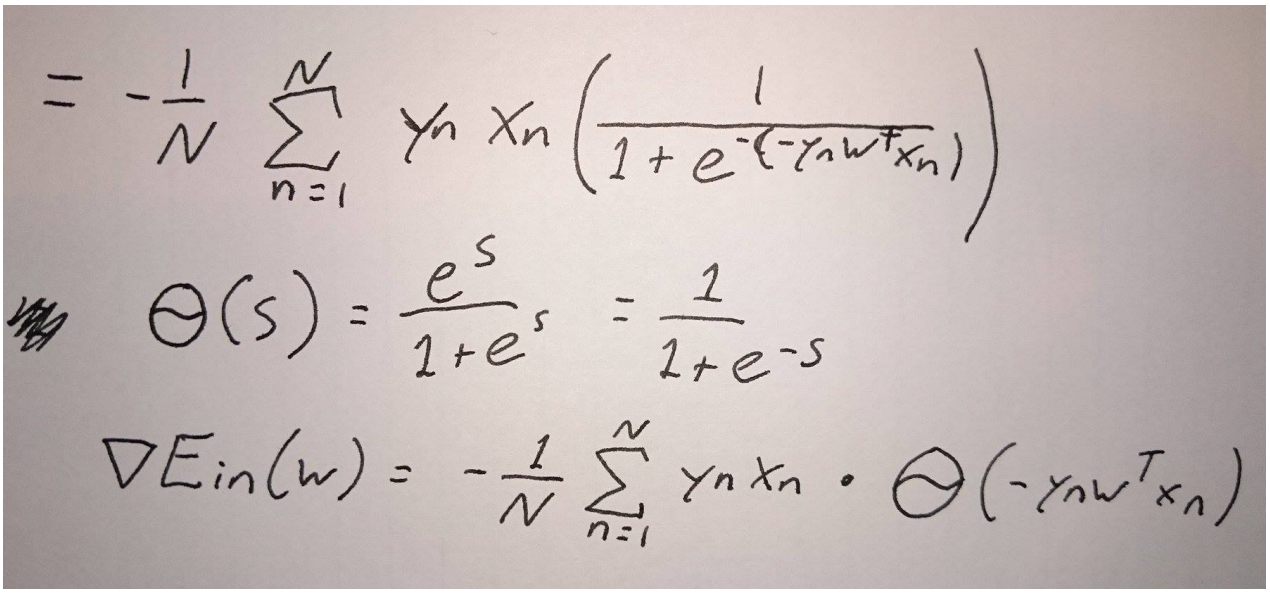

Iris2D2 Training & Test data
Y
X
7
6
5
4
3
2
1
3
4
5
6
7
8
9

Exercise 7:

This is exercise 3.7 in the book's Logistc regression chapter.

1.)

$$\nabla E_{in}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n \mathbf{x}_n}{1+e^{y_n \mathbf{w}^\top \mathbf{x}_n}}$$



$$= \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n \theta\left(-y_n \mathbf{w}^\top \mathbf{x}_n\right).$$

2.)

A 'misclassified' example contributes more to the gradient than a correctly classified one because misclassified samples act as outliers and affect log likelihood calculations more heavily than other 'true – positive' values