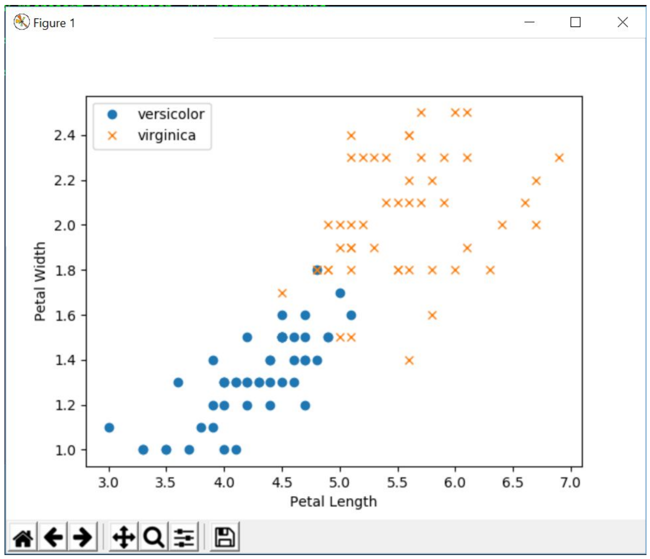
Dominique Owens
EECS 390

# Project 2 Write Up

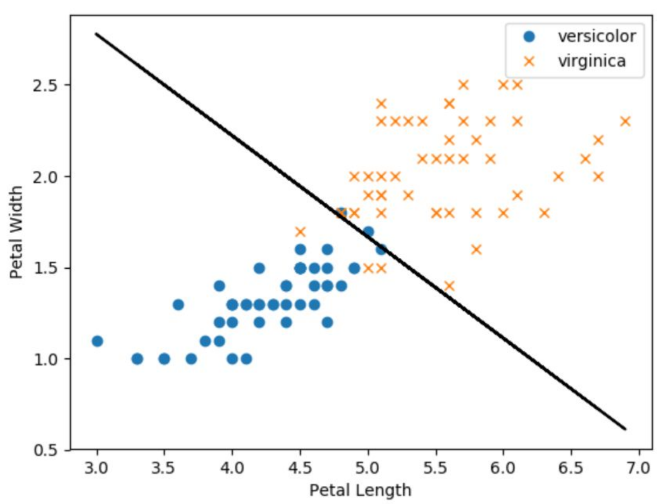
For this project I am using python 3.6

1. Exercise 1
    a. Run this line in the terminal : python ex1.py partA
    This will generate a graph and console output.

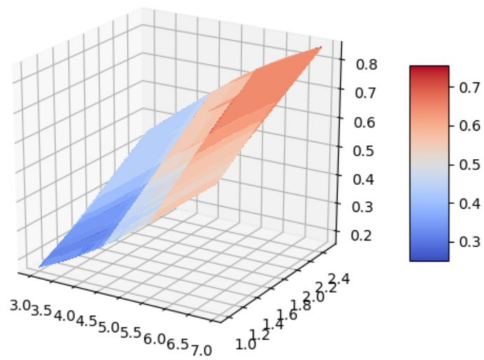



    b. Run this line in the terminal : python ex1.py partB
    This will generate console output.

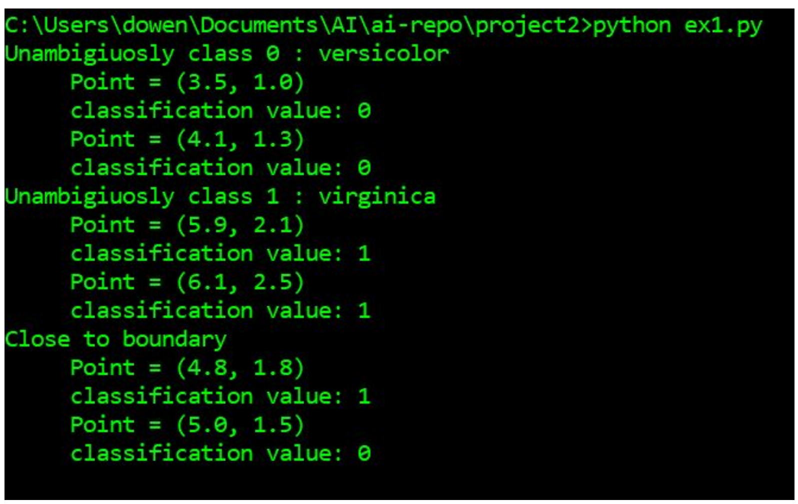
    c. Run this line in the terminal : python ex1.py partC
    This will generate a graph and console output.



d. Run this line in the terminal : python ex1.py partD
   This method will display a graph.




e. Run this line in the terminal : python ex1.py partE
   This method prints data to the console. The console data should look as follows :



```
C:\Users\dowen\Documents\AI\ai-repo\project2>python ex1.py
Unambigiuosly class 0 : versicolor
     Point = (3.5, 1.0)
     classification value: 0
     Point = (4.1, 1.3)
     classification value: 0
Unambigiuosly class 1 : virginica
     Point = (5.9, 2.1)
     classification value: 1
     Point = (6.1, 2.5)
     classification value: 1
Close to boundary
     Point = (4.8, 1.8)
     classification value: 1
     Point = (5.0, 1.5)
     classification value: 0
```

2. Exercise 2
   a. Run this line in the terminal : python ex2.py partA
      This generates console output.

   b. The derivation the gradient of the objective function above with respect to the neural network output is as follows:

Our objective function is the mean square error function, which is

$$E = \frac{1}{n}(\sigma(w \cdot x) - y)^2,$$

where $n$ is the number of data points, $y$ is the expected class $x_i$, and the sigma function is the is the sigmoid function.

To take this derivative of this function, we need to use the chain rule and take the derivative of the function itself and the sigmoid function inside it. The derivative of the objective function is as follows:

$$\frac{\partial E}{\partial w} = \frac{1}{n} * 2 * (\sigma(w \cdot x) - y) * \frac{\partial}{\partial w}\sigma(w \cdot x) * x$$

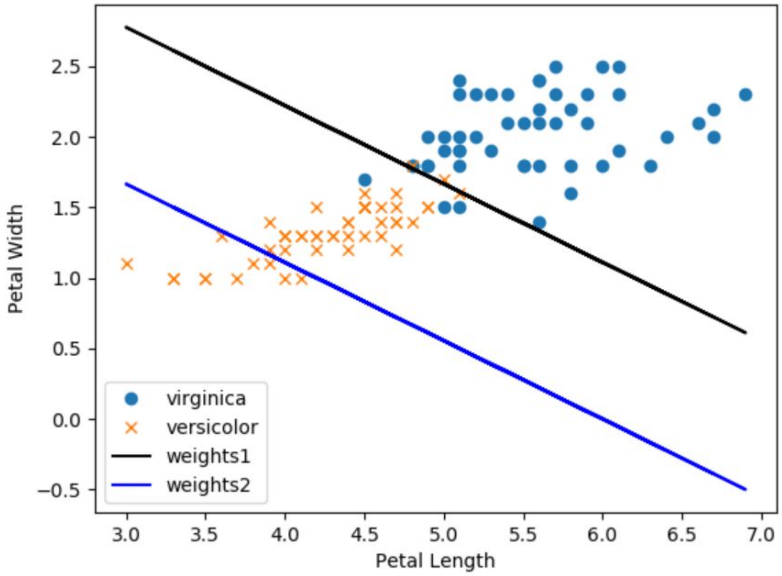
$$\frac{\partial E}{\partial w} = \frac{1}{n} * 2 * (\sigma(w \cdot x) - y) * \sigma(w \cdot x) * (1 - \sigma(w \cdot x)) * x$$

c. C
Vector Form:
$$\frac{\partial E}{\partial w} = \frac{1}{n} * 2 * (\sigma(w \cdot x) - y) * \sigma(w \cdot x) * (1 - \sigma(w \cdot x)) * x$$

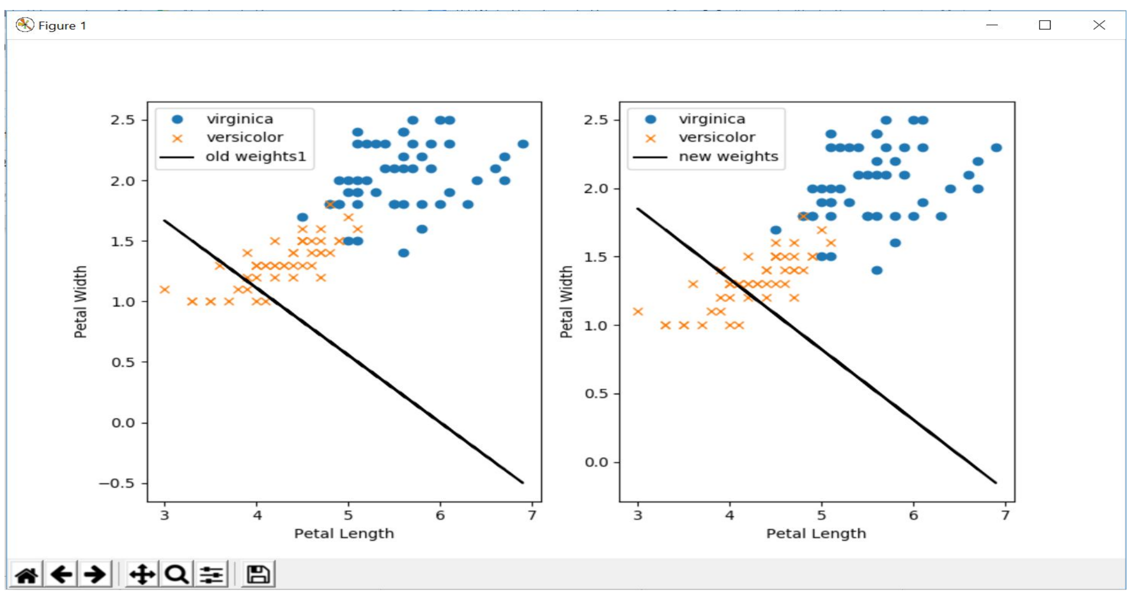
Scalar Form:
$$\frac{\partial E}{\partial w_j} = \frac{1}{n} * 2 * (\sigma(w_j \cdot x_j) - y) * \sigma(w_j \cdot x_j) * (1 - \sigma(w_j \cdot x_j)) * x_j$$

d. Run this line in the terminal : python ex2.py partD
This method prints out console data and constructs a graph. The console data and graph are as follows:

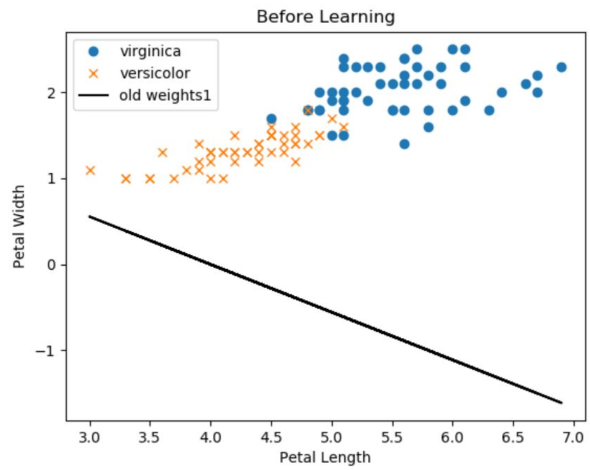
```
C:\Users\dowen\Documents\AI\ai-repo\project2>python ex2.py
Mean Squared Error for weight1: [-4, 0.5, 0.9]
0.3738014278558939
Mean Squared Error for weight2: [-3, 0.5, 0.9]
0.44354569677995287
```

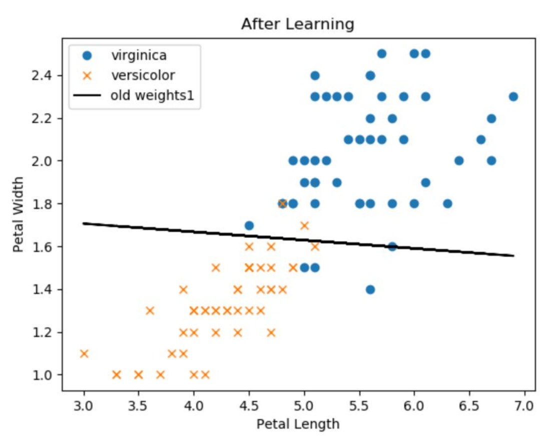


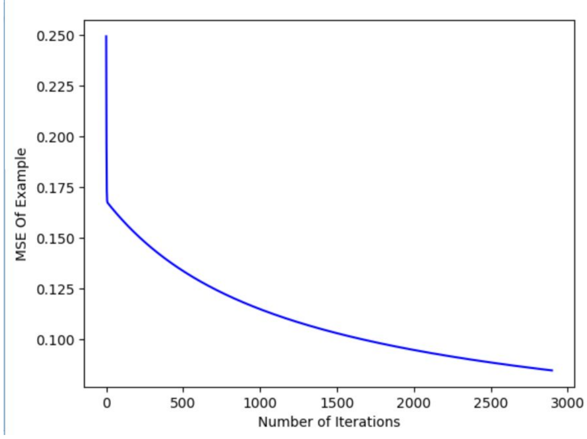

e. Run this line in the terminal : python ex2.py partE
   This method generates a graph and console output.



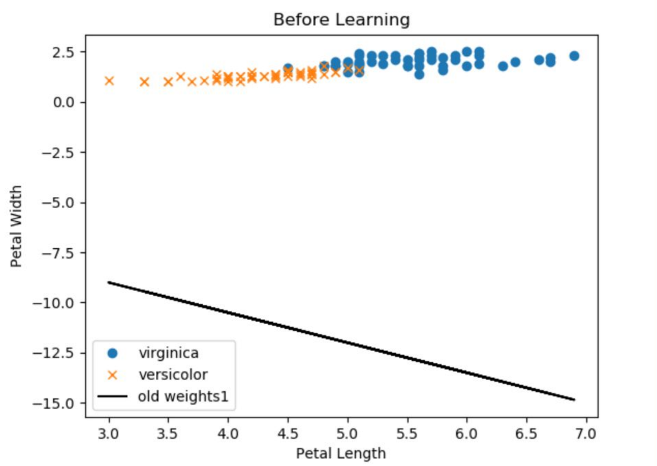

3. Exercise 3
   a. Run this line in the terminal : python ex3.py partA
      This generates several graphs and console output.

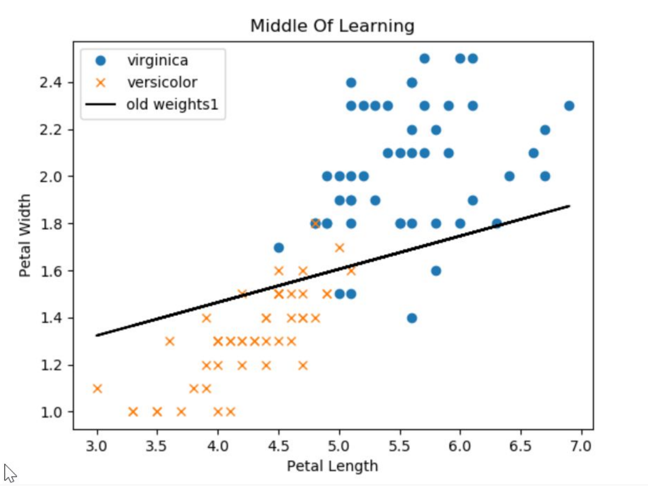


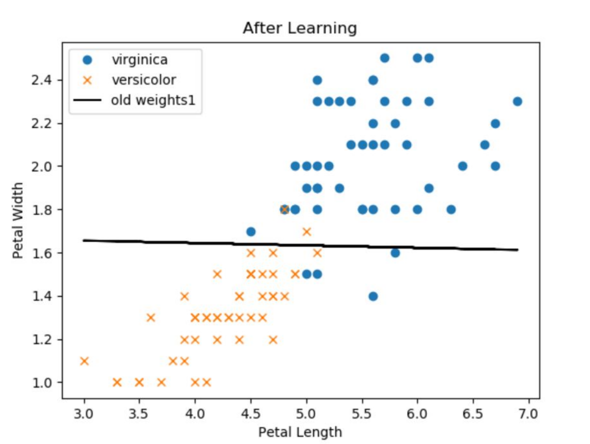



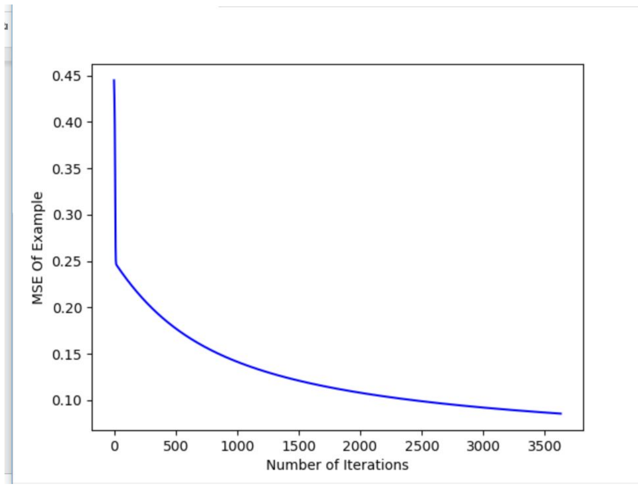




b. Included in partA
c. Run this line in the terminal : python ex3.py partC
   This generates several graphs and console output.


Before Learning
virginica
versicolor
old weights1
Petal Width
Petal Length


Middle Of Learning
virginica
versicolor
old weights1
Petal Width
Petal Length


After Learning
virginica
versicolor
old weights1
Petal Width
Petal Length




d. I chose an epsilon value of .1 because it allows for more iterations which provides more examples for learning.
e. I chose to stop the gradient decent when the difference of the mean squared error of the weights is very small. This means that the difference between the weights themselves is negligible and the function has converged to a local minimum.

4. Extra Credit
    a. Run this line in the terminal : python extra_credit.py partA
    b. Run this line in the terminal : python extra_credit.py partB