M.L. Week 1 Assignment

Con Óg Ó Laoghaire

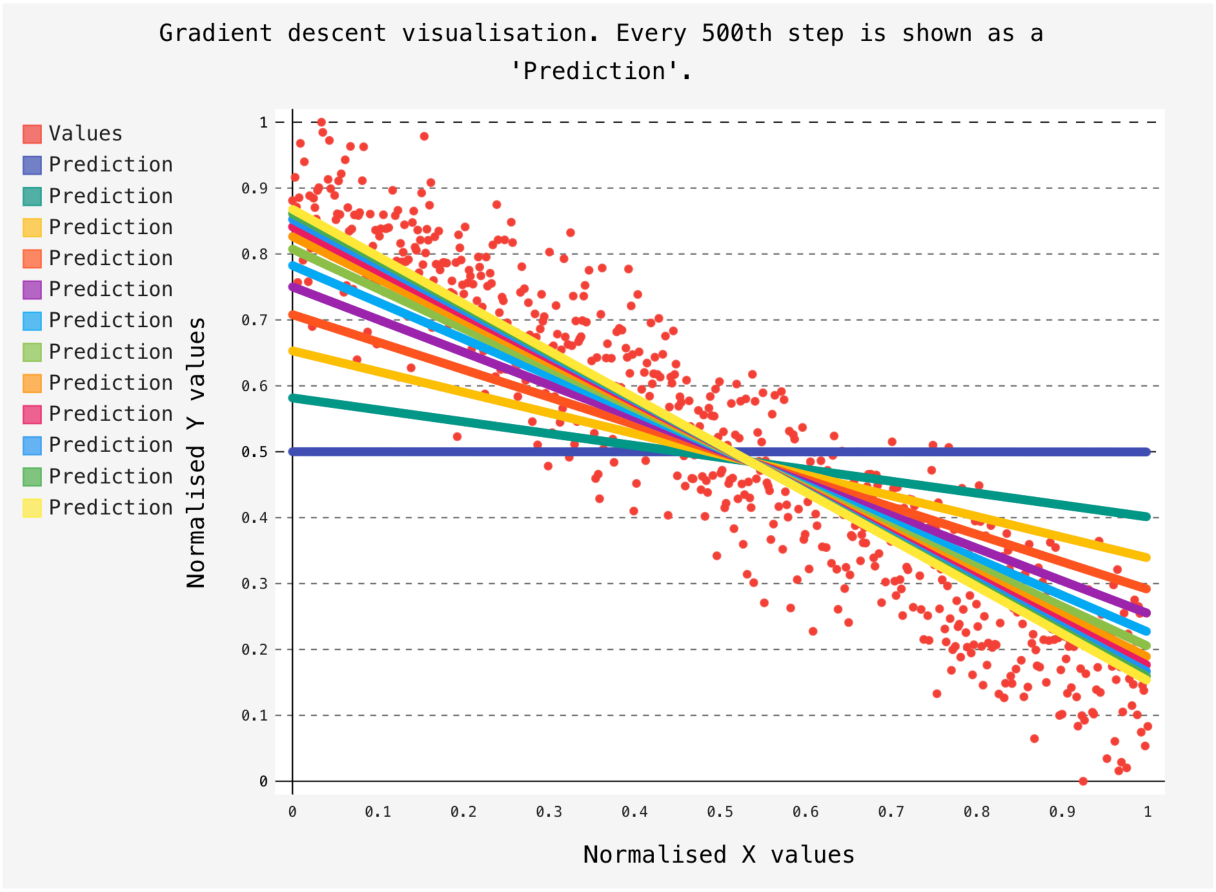
(a)

(i) I read in the datasheet with a reader from the csv library.

(ii) Normalising my data caused me great difficulty because I left this step until last. I designed the rest of my code to work on the non-normalised data, which led me to design my code based on various assumptions that were true only of the non-normalised data. The actual normalisation math was handy enough to make a function for. I also normalised the indices of the y-values, aka the x-values.

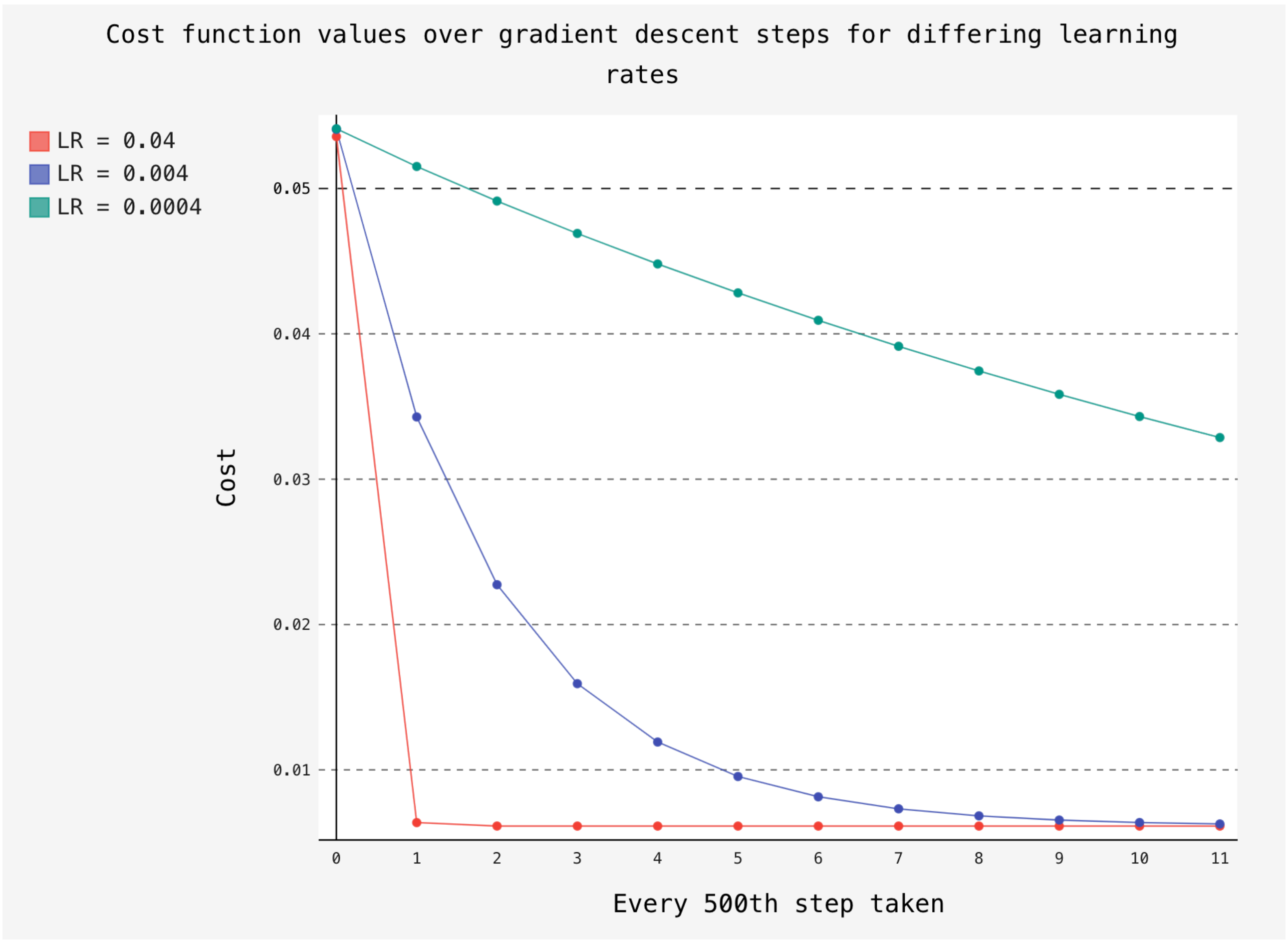
(iii) I probably made this step a lot harder for myself by using a high-level chart plotting library, rather then something more fundamental and straight-to-the-point like matplotlib (which I used for my very last graph). The library I used, pygal, doesn't have the functionality to plot multiple types of data representations on the same graph, e.g. dots as well as lines, so I ended up writing my own function, addLineToGraf, that draws a series of dots to form a line. However this line drawing technique is horrific in the cost it incurs upon execution time, which slowed down my trial and error cycle when working on the rest of the assignment.

Here is my very colourful, high-level library visualising my gradient descent implementation:

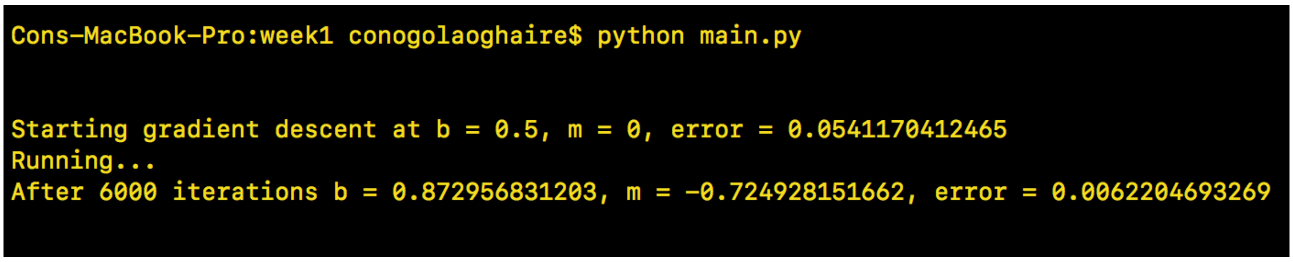


(b)

(i) This was handy enough to implement on top of my initial gradient desicent implementation. I changed my learning rate from being a global constant to a local variable that could be used as a parameter of my gradient descent runner.

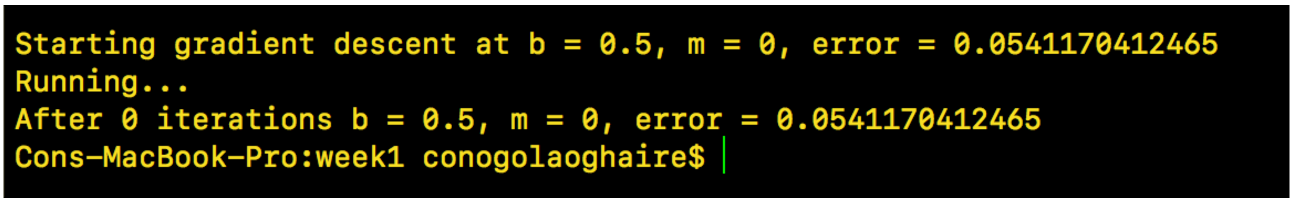


(ii)



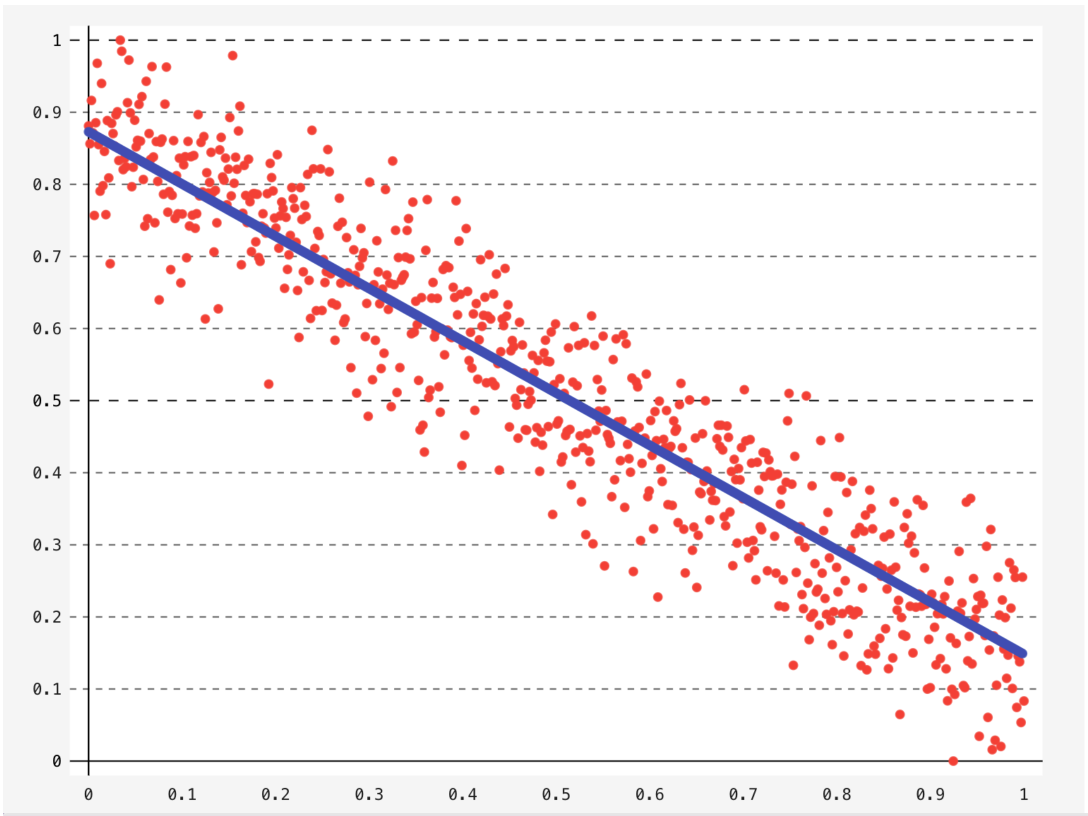
```
Cons-MacBook-Pro:week1 conogolaoghaire$ python main.py


Starting gradient descent at b = 0.5, m = 0, error = 0.0541170412465
Running...
After 6000 iterations b = 0.872956831203, m = -0.724928151662, error = 0.0062204693269
```

(iii)



```
Starting gradient descent at b = 0.5, m = 0, error = 0.0541170412465
Running...
After 0 iterations b = 0.5, m = 0, error = 0.0541170412465
Cons-MacBook-Pro:week1 conogolaoghaire$
```

(iv)
The first graph here is the final prediction of my gradient descent implemenation visualised with pygal.
The second graph is the bog-standard product of the go-to libraries(pandas, sklearn, and matplotlib) applied to week-1's assignment.
They values these graphs represent look bang on the same to my human eye.
I am quite glad I went through implementing afresh with all the conventional libraries(rather then just doing what I had been doing with the addition of sklearn), because now I see exactly how useful they are.



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

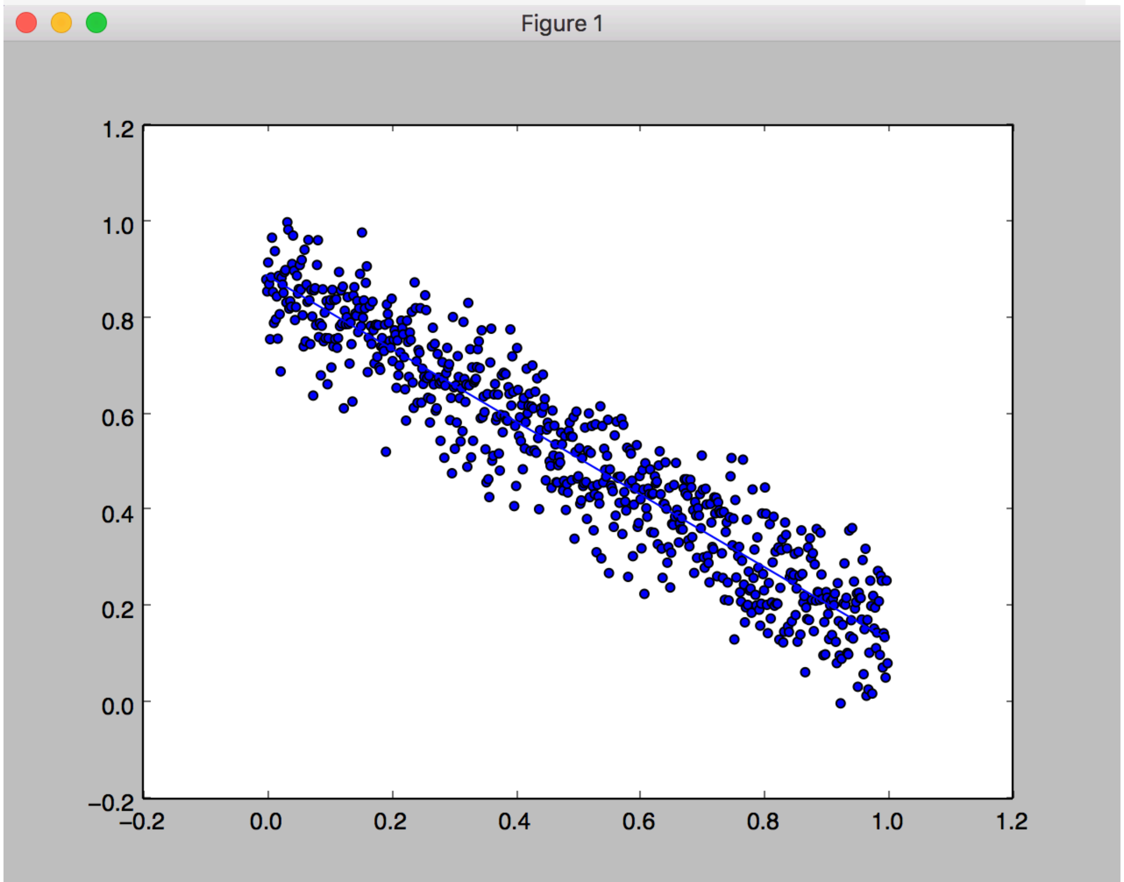


Figure 1