MATH96012 Project 2

Matthew Cowley 01059624

November 15, 2019

Part 1.1

Code for CLR appears to be working well.

Part 1.2

Code to calculate the $\mathrm{clr}_t est_e rror is wrong, I couldn't not find the mistake.$

Part 1.3

$$C = -\sum_{k:Y^{(k)}=1} log\left(a_1^{(k)}\right) - \sum_{k:Y^{(k)}=0} log\left(1-a_1^{(k)}\right) + \lambda\sum_{j=1}^{N}\left(w_j^2\right) \;\; M1$$

$$C = -\sum_{k=1}^{D}\left[Y^{(k)} log\left(a_1^{(k)}\right) + \left(1-Y^{(k)}\right) log\left(1-a_1^{(k)}\right)\right] + \lambda\sum_{j=1}^{N}\left(w_j^2\right) \;\; M2$$

There are 2 versions of the cost function we can use, the difference between the 2 is that one index's over Y for $Y^k = 0$ and $Y^k = 1$, calculating 2 sums independently with this index's. But the other just makes use of the fact that Y is either 1 or 0. So multiply by Y is the same as indexing over $Y^k = 1$ (similarly multiplying by 1-Y is the same as indexing over $Y^k = 0$). So the 2nd method just iterates over all the data with no need to calculate the index.

I used the 2nd method (M2) as it is far easier to code requiring no index's to be calculated.

Part 2.1

Again appears to working, but was unable to confirm.

Part 2.2

Again $\mathrm{mlr}_t est_e rror code is outputting incorrect values, this might be due to a mistake in mlr model code.$

Part 2.3

An Advantage of Python is it is easy to program the problem in. This is because you need less code to do the same task e.g you don't have to declare dimensions and type of variables at start of program. There are many packages already available as well to do the task. Also it is very hard to see result of FORTRAN as you develop the code due to it being a compile language. As a result makes it hard to discover your errors. There is also not a good unit testing framework for FORTRAN.

The main benefit of using FORTRAN to do mlr model is the speed up in calculations, allowing you to use higher d,n and m. Due to the size of the problem FORTRAN's high speed and compile nature mean, it is faster then python, enabling it process more data so give better results. Essentially FORTRAN has better performance for this task.

Part 2.4

See next page.

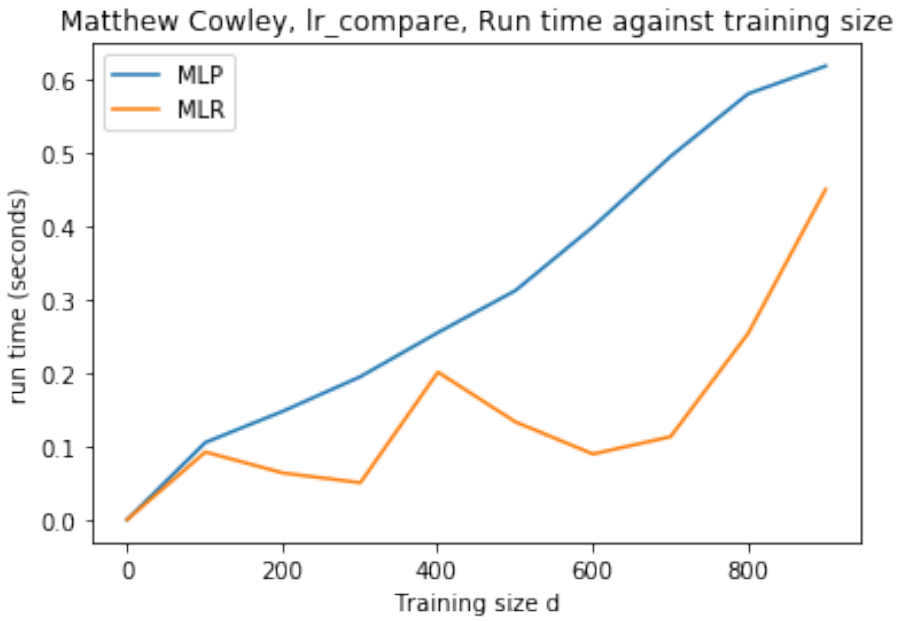

Figure 1: How training size effects run time for different methods

Part 2.4

As we can see as training size increases run time increases for both MLR and MLP. But what is interesting is that MLR is quicker than MLP for $d > 100$ and $d < 800$. For $d < 100$ the run time is quite similar for both methods, and this would warrant further investigation for this region of training data size. Moreover, we should note that this result is to be expected, since MLP is itself written in a compiled language and is highly optimized, so we would expect it to outperform our fortran code in terms of run time. In addition, we should also be careful with our conclusions due to the difference in the actual methods we are implementing - so even with perfect fortran implementation of our mlr model, it may be that the neural network with the parameters we set is inherently faster to compute. Some conclusions may also be wrong due to the MLR code being incorrect, due to it giving implausible outputs.