HW3 Report

Section 1

Q1:

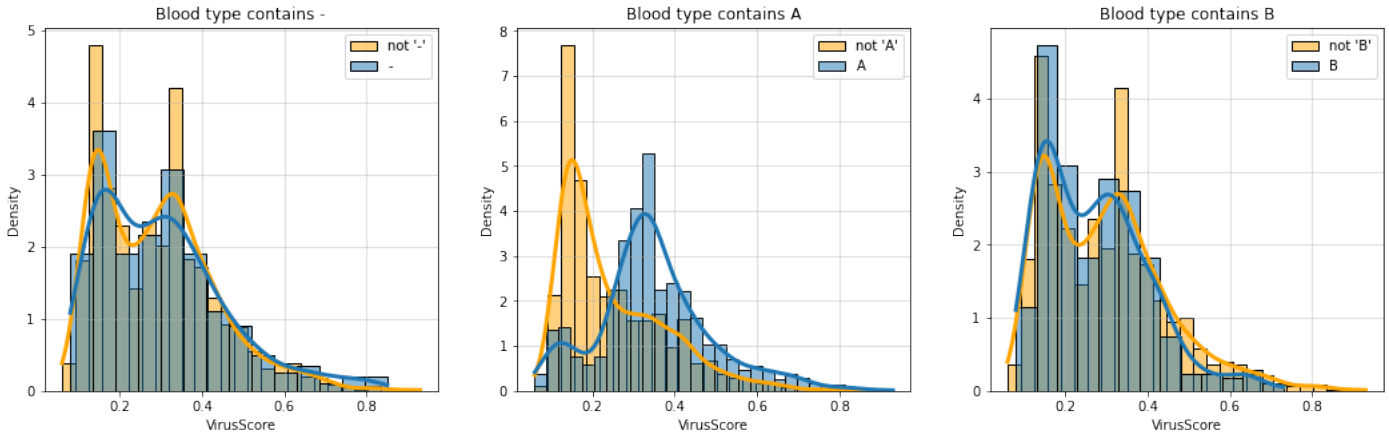

Figure 1: KDE plots of `VirusScore` conditioned different conditions of `blood_type`

Q2:

In figure 1 in the plot of A versus not A, we observe that the groups of patients with and and without "A" in their blood types are mostly seperable along a boundary that is approximately the `VirusScore` of 0.225.

Therefore, the condition of contains/does not contain A would be most informative for learning `VirusScore`. As it turns out, we decided already in hw1 to create this feature.

Q3:

$$\frac{\partial}{\partial b}\mathcal{L}\left(\underline{\mathbf{w}},b\right)\overset{a}{=}\frac{\partial}{\partial b}\frac{1}{m}\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i+b-y_i\right)^2\overset{b}{=}\frac{1}{m}\frac{\partial}{\partial b}\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i+b-y_i\right)^2\overset{c}{=}\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b}\left(\underline{w}^{\top}\underline{x}_i+b-y_i\right)^2\overset{d}{=}$$

$$\overset{d}{=}\frac{1}{m}\sum_{i=1}^{m}2\left(\underline{w}^{\top}\underline{x}_i+b-y_i\right)\cdot(1)\overset{e}{=}\frac{2}{m}\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i+b-y_i\right)\overset{f}{=}\frac{2}{m}\left[mb+\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i-y_i\right)\right]\overset{g}{=}$$

$$\overset{g}{=}2b+\frac{2}{m}\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i-y_i\right)\Rightarrow\frac{\partial}{\partial b}\mathcal{L}\left(\underline{\mathbf{w}},b\right)=2b+\frac{2}{m}\sum_{i=1}^{m}\left(\underline{w}^{\top}\underline{x}_i-y_i\right)$$

a : Definition of $\mathcal{L}\left(\underline{\mathbf{w}},b\right)$
b : $\frac{1}{m}$ is scalar
c : Derivative of a sum is the sum of derivatives
d : Derivative of $\left(\underline{w}^{\top}\underline{x}_i-b-y_i\right)^2$ w.r.t b
e : 2 is scalar
f : Sum of b
g : Removing b from the sum

Q4:

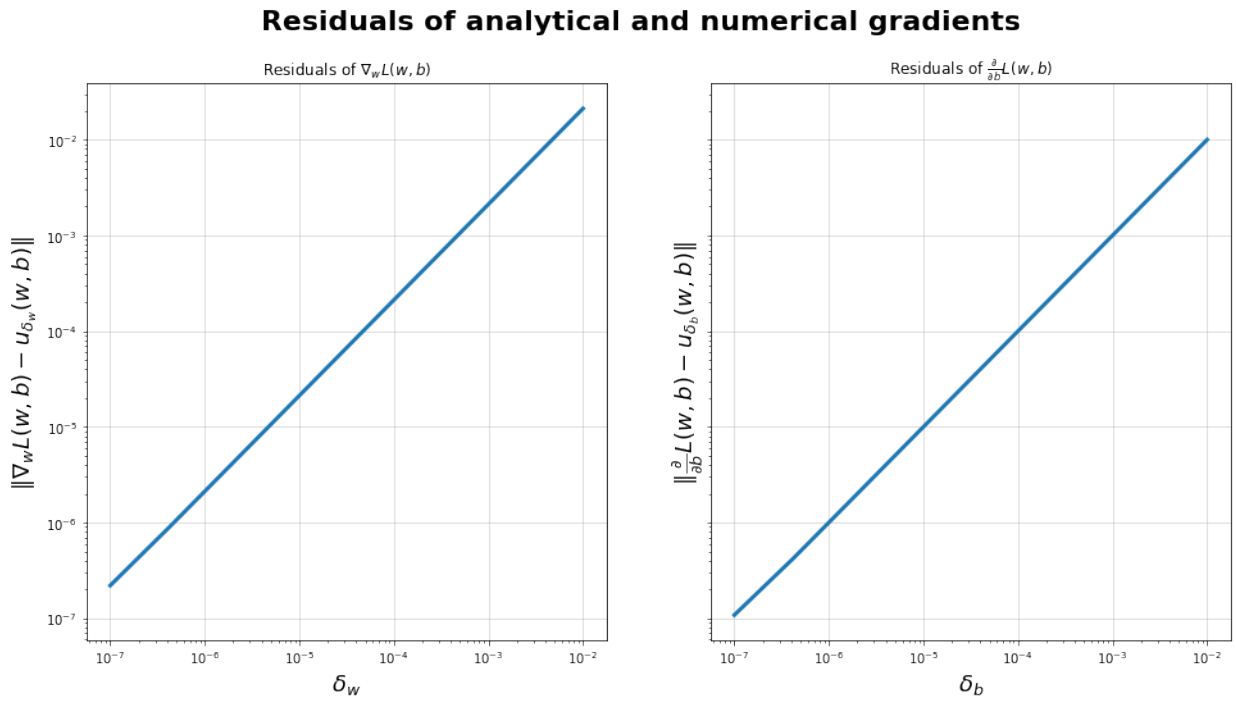

Figure 2: Plot of Residuals of analytical and numerical gradients

As we can see in figure 2, the difference between the analytic and numerical gradients increases in a monotonic fashion as the value of δ increases. This is logical, as δ is the differential size used in the definition of the numerical gradient, and therefore a smaller δ equates to a more precise estimation of the analytic gradient by the numerical gradient.

Q5:

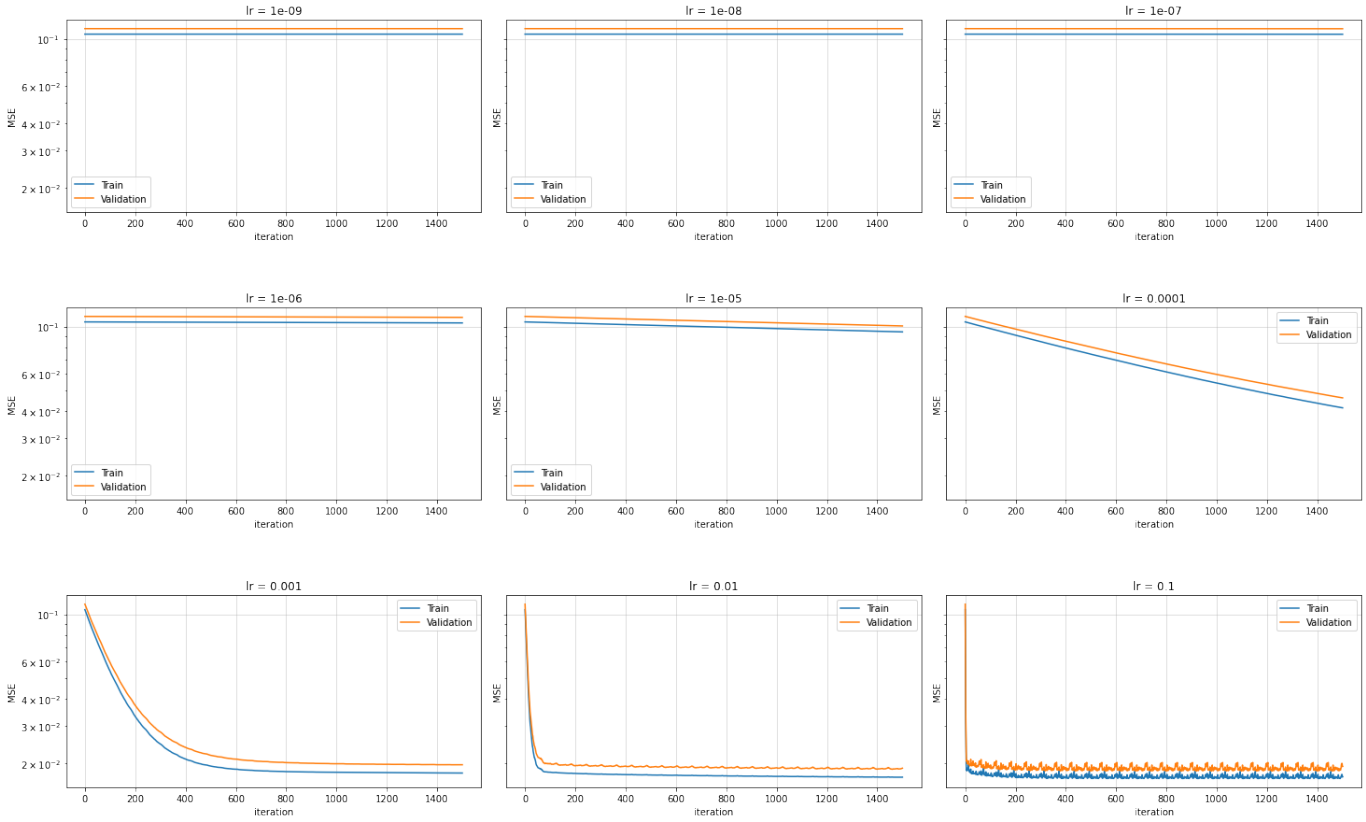

Figure 3: Graphs of Training and Validation Losses as Functions of Iteration Number for Different Learning Rates

We can see in figure 3 that for smaller lr, for those that converge, the convergence is at a higher loss for both the training and validation. This matches the theory, since if the lr is too small, the SGD algorithm is likely to converge to a sub-optimal solution. In addition we observe that for the lr equal to 0.1 the graphs do not converge for both training and validation losses and for lr equal to 0.01 the validation loss does not converge, which fits the theory that says that learning rates that are too high are likely to cause the SGD algorithm to take steps that are too large and thus repeatedly skip-over the optimal solution. This points to 0.001 as being the optimal lr, as both the validation and training losses converge to values that are substantially lower than the next smaller lr.

Q6:

Model	Section	Train MSE	Valid MSE
		Cross Validated	
Dummy	3	0.0204	0.0205

Section 4

Q8:

Model	Section	Train MSE	Valid MSE
		Cross Validated	
Dummy	3	0.0204	0.0205
Ridge linear	4	0.0204	0.0205

Section 6

Q16:

When using a polynomial feature mapping, we can expect the training error to decrease and the validation error to increase. This is because the polynomial mapping will give more flexibility to the regression model to more closely try to fit the training data during training in a polynomial way, thereby leading to lower training error, but also causing overfitting and therefore leading to a higher validation error.