COMP 3105 - Assignment 1

Alex Nedev(101195595), Jacob Lane(101220538)

October 1, 2023

Question 1

Part d

Table 1: Different training losses for different models

Model	L_2 loss	L_1 loss	L_∞ loss
L_2 Model	0.01614138	0.14072289	0.43503039
L_1 Model	0.01807485	0.1320479	0.49792531
L_∞ Model	0.02292726	0.17948013	0.32204459

Table 2: Different test losses for different models

Model	L_2 loss	L_1 loss	L_∞ loss
L_2 Model	0.02553363	0.17981798	0.76483583
L_1 Model	0.02805635	0.18837807	0.80326613
L_∞ Model	0.03441486	0.2074841	0.88088327

Part e

Looking at the tables above it's clear that not all of the loss types are created equal. In the training losses table, the L_1 loss is consistently the smallest one for every single model that was used. Furthermore, the training loss table seems to indicate that the L_∞ loss produces the highest loss out of all 3 types while the L_2 is not far behind the small loss rate of the L_1 loss.

Once we apply all our models to the test data, the differences between the 3 models are easier to spot. The L_1 loss is rather robust since it produces the smallest difference between the training and test data, seemingly providing a near identical loss rate. The L_∞ and L_2 loss on the other hand seems to be about twice as large for the test data as opposed to the training data.

However, the L_2 loss is still much smaller than the L_∞ loss despite this apparent doubling. One potential reason for the robustness of the L_1 loss is that, by design, it treats every error equally. As we've seen from the lectures, because it doesn't follow/chase outliers all that much, it can lead to quite a small loss for all of our models. The L_∞ loss on the other hand can be rather sensitive to outliers which tends to leave behind a larger loss when calculated for all of our models. Perhaps a reason why the L_2 loss seems to be quite robust while still having a larger loss value than the L_1 loss is because while it still does treat every error equally, it seems to be slightly mor sensitive to outlier than the L_1 loss. It is sort of like a middle ground between the robustness of the L_1 loss and the sensitivity of the L_∞ loss.

Part g

Table 1: Different training losses for different models

Model	L_2 loss	L_1 loss	L_∞ loss
L_2 Model	1.40974432	1.50926994	3.10899552
L_1 Model	1.74242007	1.43952693	4.18816745
L_∞ Model	1.50317552	1.57453811	2.45371274

Table 2: Different test losses for different models

Model	L_2 loss	L_1 loss	L_∞ loss
L_2 Model	1.5095517	1.55802469	3.30371182
L_1 Model	1.93067456	1.56145814	4.35304688
L_∞ Model	1.49021116	1.56429653	2.71521647

Question 2

Part d

Table 3: Training accuracies with different hyper-parameters

m	Train Accuracy	d	Train Accuracy	eta	Train Accuracy
10	0.937	1	0.8394	0.1	0.92355
50	0.9255	2	0.92685	1.0	0.9271
100	0.91695	4	0.9816	10.0	0.9225
200	0.921575	8	0.9995	100.0	0.90355

Table 4: Test accuracies with different hyper-parameters

m	Test Accuracy	d	Test Accuracy	eta	Test Accuracy
10	0.90283	1	0.84244	0.1	0.919955
50	0.91731	2	0.919615	1.0	0.919335
100	0.919285	4	0.974585	10.0	0.917825
200	0.919105	8	0.99726	100.0	0.897295

Part e

All the data from the training accuracy to the test accuracy is similar.

It is clear that changing the values of m causes some negative effects on the training accuracy but it seems pretty negligable to the overall accuracy

The d value however seems to be much more meaningful, everytime you double the d value the accuracy seems to go up drastically, I think this is because d controls the number of y elements in X and x elements in w, and having more elements in your training set will lead to significantly more accurate models for your test set.

The increased step size or eta also seems to negatively impact performance, this makes sense because an increased step size can actually cause you to miss the minimum so you could lose accuracy

Part g

eta	Training Accuracy	Validation Accuracy	Test Accuracy
0.1	0.84337349	0.72289157	0.83333333
1	0.90361446	0.69879518	0.73809524
10	0.75903614	0.77108434	0.78571429
100	0.90361446	0.60240964	0.66666667

This data shows, somewhat counterintuitively, that the best eta's are 0.1 and 10 while 1 and 100 fall behind rather significantly. I would recommend using 10 though, its train accuracy may be lower but it has a higher validation accuracy, I think in 0.1 the model overfit the training data because during the validation it was significantly lower. While the eta of 10 actually underfit the training data a little but during validation it performed significantly better. The final result may be lower in the eta of 10 but you cannot pick your model based on test accuracy.