COMPENG 4SL4

Assignment 4

Instructor: Dr. Dumitrescu

Hritheekka Chinnakonda – chinnakh – 400292782 – C01 – L03

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [Hritheekka Chinnakonda, chinnakh, 400292782]

Splitting ratio:

- Training set: 60% of initial dataset
- Validation set: 20% of initial dataset
- Test set: 20% of initial dataset

Activation Function used at the hidden units and output: Logistic Sigmoid (Derivative Logistic Sigmoid for Backpropagation)

Variant of GD: Batch GD

Reasoning: Batch GD has a more deterministic convergence to the global minimum compared to other variants, as this method considers the entire dataset for each update, resulting in a more accurate estimate of the gradient.

Early Stopping Method:

- Fixed a number of epochs to run the GD algorithm
- after each epoch iteration:
    - the training cross-entropy loss and the validation cross-entropy loss were computed
- lastly, the weights that achieve the lowest validation cross-entropy loss were chosen

How the weights were initialized:
The weights are initialized using methods that considers the input and output dimensions of each layer. The method that was employed to initialize the weights in my code is called Xavier or Glorot Initialization in which the initial weights are set to avoid exploding or vanishing gradients during training. They are randomly generated using a uniform distribution within set limits.

Number of epochs or iterations: 1000

Learning Rate: 0.005

Specify the training and the validation cross-entropy loss for each model that you trained:

Run	n1	n2	Validation Loss	Training Loss
1	1	1	0.689596783	0.689270974
2	1	1	0.689593028	0.689267446
3	1	1	0.689591795	0.689265406
1	1	2	0.689591364	0.689264754
2	1	2	0.689591242	0.689264123
3	1	2	0.68959483	0.689272098
1	1	3	0.689591244	0.689264318
2	1	3	0.689605032	0.689259223
3	1	3	0.689597099	0.689271095
1	1	4	0.689609262	0.689287425
2	1	4	0.689592725	0.689265705
3	1	4	0.689591389	0.689267057
1	1	5	0.689591342	0.68926592
2	1	5	0.689596523	0.689258556
3	1	5	0.689596705	0.68925816
1	1	6	0.689592452	0.689267615
2	1	6	0.689601703	0.689271389

3	1	6	0.689591257	0.68926615
1	1	7	0.689593782	0.68927111
2	1	7	0.689591621	0.68926804
3	1	7	0.68959147	0.689262449
1	2	1	0.689598625	0.689270761
2	2	1	0.689796659	0.689393118
3	2	1	0.689594521	0.689269237
1	2	2	0.689596251	0.689271036
2	2	2	0.689591234	0.689264699
3	2	2	0.689599277	0.689276142
1	2	3	0.689668055	0.689294109
2	2	3	0.689591418	0.689264233
3	2	3	0.689591691	0.689264662
1	2	4	0.689598582	0.689257394
2	2	4	0.689592149	0.689261159
3	2	4	0.689598281	0.689257348
1	2	5	0.68959345	0.689274717
2	2	5	0.689591456	0.689265183
3	2	5	0.689594455	0.689270355
1	2	6	0.689594379	0.689263176
2	2	6	0.689591716	0.689263247
3	2	6	0.68959306	0.689265629
1	3	1	0.689593433	0.689267271
2	3	1	0.689592605	0.689266552
3	3	1	0.68959345	0.689268487
1	3	2	0.68959295	0.689270659
2	3	2	0.689592576	0.689263551
3	3	2	0.689593909	0.689268213
1	3	3	0.689591828	0.689264767
2	3	3	0.689609664	0.689290179
3	3	3	0.689591321	0.689264383
1	3	4	0.689612561	0.689260975
2	3	4	0.689591841	0.68926732
3	3	4	0.689592331	0.689264714
1	3	5	0.689593035	0.689263973
2	3	5	0.689614174	0.689262427
3	3	5	0.68959188	0.689267892
1	4	1	0.689591464	0.689265895
2	4	1	0.689595808	0.689269435
3	4	1	0.6895929	0.689267606
1	4	2	0.689593926	0.689267655
2	4	2	0.689615195	0.689310958
3	4	2	0.689593064	0.689267704
1	4	3	0.689620027	0.689265022
2	4	3	0.68959296	0.689268352
3	4	3	0.689592567	0.689265489

1	4	4	0.689592667	0.689263186
2	4	4	0.689591901	0.689265289
3	4	4	0.689591457	0.6892648
1	5	1	0.689591531	0.689264542
2	5	1	0.689596077	0.689270296
3	5	1	0.689594604	0.689269727
1	5	2	0.689592076	0.689266958
2	5	2	0.68959188	0.689266017
3	5	2	0.689592909	0.689264359
1	5	3	0.689594153	0.6892718
2	5	3	0.689593143	0.689267072
3	5	3	0.689599357	0.689275959
1	6	1	0.689592241	0.689265762
2	6	1	0.689595651	0.689269898
3	6	1	0.689592339	0.68926581
1	6	2	0.689594705	0.689269501
2	6	2	0.689591594	0.689266035
3	6	2	0.689594875	0.689269191
1	7	1	0.689598277	0.689272373
2	7	1	0.689595973	0.689271637
3	7	1	0.689591613	0.689265755

Specify the pair (n1, n2) that you finally chose for your model and the weights of the trained model:

Chosen pair: n1 = 1, n2 = 2

Weights:

Hidden Layer 1	array([[-0.05668723, -0.07832953], [-0.15455762, 0.19406176], [-0.1536965 , -0.15776204], [0.04848469, 0.36638581]])
Hidden Layer 2	array([[0.39144226, -0.11701077], [-0.41561928, -0.04674589]])
Output Layer	array([[-0.01214964], [-0.33991867]])

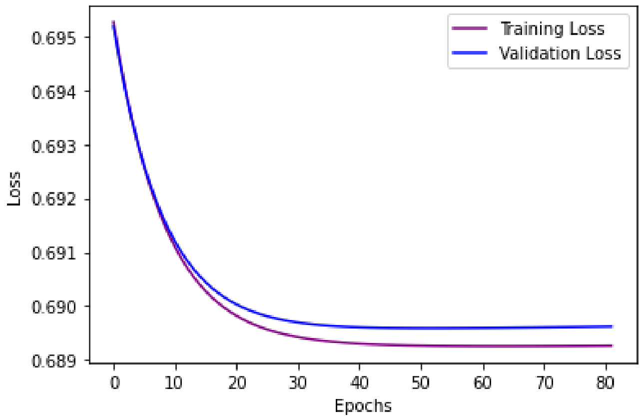
Best Loss of the final model:

- Training set = 0.689265754705322
- Validation set = 0.6895912342086119
- Test set = 0.39636363636363636

Misclassification errors of the final model:

- Training set = 0.45592705167173253
- Validation set = 0.45785876993166286
- Test set = 0.39636363636363636

Plot of the learning curve:



When looking at the learning curve plot above, it is seen that as the number of epochs increases, both the validation and training loss form a U-shape and decreases until finally plateauing.

The error values can be reduced further by identifying where the classifier falls short or struggles in specific functions. Further experimenting can be done with different model architectures, or hyperparameter tuning (learning rate, batch sizes etc.)