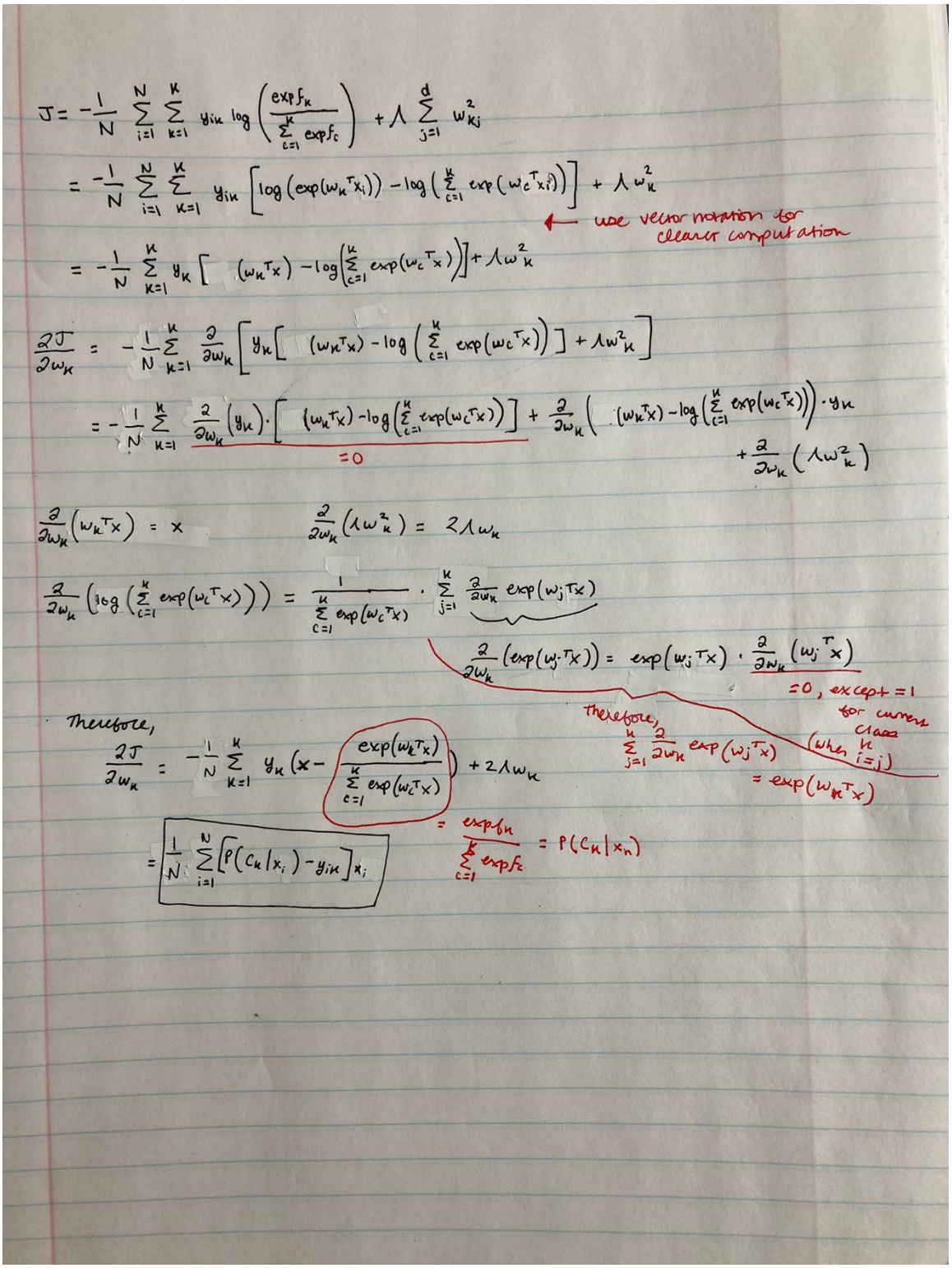
Allison Aprile
CS-584 A
March 1, 2021
HW1 - Report

Gradient Derivation (Model Evaluation on next page)



Allison Aprile
CS-584 A
March 1, 2021
HW1 - Report

Model Evaluation

Overall Comparison

*(Precision and recalls are macro-averaged)

Classifier	Epochs	Parameters	Training Loss	Validation Loss	Validation Precision	Validation Recall	Testing Precision	Testing Recall
LR (Mini-Batch)	100	Batch_size = 100, lambda = 0.01, Learning_rate = 0.01	18.314	18.367	0.26	0.25	0.25	0.25
LR (SGD)	100	Lambda = 0.01, learning_rate = 0.01	~	~	0.80	0.74	0.80	0.73
MLP	10	Batch_size = 100, hidden_neurons = 5, learning_rate = 0.01	-0.606 (Incorrect calculation)	-0.619 (Incorrect calculation)	0.44	0.25	0.21	0.26

Unfortunately, my classifiers did not perform as expected, with the exception of the Logistic Regression SGD. However, because the training loss and validation loss could not be recorded, I cannot analyze these results too extensively against the other models.

In general, the MLP should be the best model because, despite having a longer training time than LR (measured by runtime, not epochs), it converges faster (based on epochs). This can be seen in the table above.

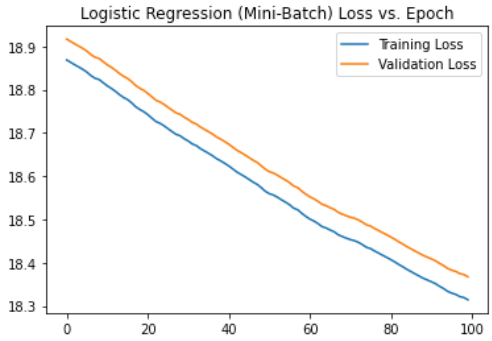
The only useful metric in the entire table is the precision and recall. When training, the goal is to get both precision and recall to 1.0. However, that is not reasonable – improving one typically reduces the other. For both LR (Mini-Batch) and MLP, the precision and recall is no better than 'random guess', with the exception of the MLP validation precision. This indicates that for one class (as seen in the additional charts below), the model had fewer false positives than it would given a 'random guess'. For my MLP, this was Class 4. Interestingly, this did not translate to the testing, where the precision for Class 4 was 0.

Overall, I would say that LR (SGD) performed the best of my models. Theoretically, SGD is more efficient than standard batch GD or mini-batch GD because it does not have to iterate through multiple samples when calculating the gradient. As for why it outperformed LR (Mini-Batch), given they were both trained with the same number of epochs, SGD reduced the loss faster than Mini-Batch. Plotting the Mini-Batch GD losses alongside the SGD losses vs. number of epochs would have reinforced this.

(See next page for detailed model performance analytics)

Allison Aprile
CS-584 A
March 1, 2021
HW1 - Report

Logistic Regression (Mini-Batch)



Training Loss: 18.31403
Validation Loss: 18.3672

Validation Class-Level Analytics:

	precision	recall	f1-score	support
1	0.26	0.32	0.29	5944
2	0.25	0.10	0.14	6057
3	0.28	0.10	0.14	5979
4	0.23	0.47	0.31	6020
accuracy			0.25	24000
macro avg	0.26	0.25	0.22	24000
weighted avg	0.26	0.25	0.22	24000

```
[[1918  503  439 3084]
 [1558  594  483 3422]
 [1916  564  575 2924]
 [1913  711  563 2833]]
```

Testing Class-Level Analytics:

	precision	recall	f1-score	support
1	0.26	0.32	0.29	1900
2	0.27	0.10	0.15	1900
3	0.24	0.08	0.12	1900
4	0.24	0.48	0.32	1900
accuracy			0.25	7600
macro avg	0.25	0.25	0.22	7600
weighted avg	0.25	0.25	0.22	7600

```
[[ 608  146  158  988]
 [ 505  193  145 1057]
 [ 632  189  155  924]
 [ 599  199  183  919]]
```

Based on the above results, this model performs only slightly better than ‘random guess’. To adjust this model, I would a) increase the batch size, b) increase the learning rate, and/or c) train for more epochs.

Allison Aprile
CS-584 A
March 1, 2021
HW1 - Report

Logistic Regression (SGD)

*Complexity issue with cross-entropy loss calculation prevented me from getting loss data

Validation Class-Level Analytics:

	precision	recall	f1-score	support
1	0.57	0.94	0.71	6071
2	0.97	0.60	0.74	5920
3	0.91	0.58	0.71	5980
4	0.77	0.82	0.80	6029
accuracy			0.74	24000
macro avg	0.80	0.74	0.74	24000
weighted avg	0.80	0.74	0.74	24000

```
[[5720   92  120  139]
 [2055 3554   21  290]
 [1469    5 3458 1048]
 [ 833   20  217 4959]]
```

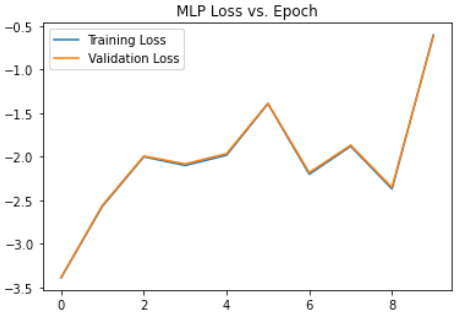
Testing Class-Level Analytics:

	precision	recall	f1-score	support
1	0.56	0.93	0.70	1900
2	0.97	0.60	0.74	1900
3	0.88	0.58	0.70	1900
4	0.78	0.81	0.79	1900
accuracy			0.73	7600
macro avg	0.80	0.73	0.73	7600
weighted avg	0.80	0.73	0.73	7600

```
[[1776   33   45   46]
 [ 670 1137    9   84]
 [ 481    3 1103  313]
 [ 261    4   92 1543]]
```

This model performed the best out of the three, especially in terms of class performance. Overall, it performs above the standard of 'random guess'. The precision and recall values are distributed fairly equally amongst the class, meaning that it is an unbiased model – unlike the MLP especially, whose precision and recall values indicate bias in the weights (despite training on a balanced dataset).

Allison Aprile
CS-584 A
March 1, 2021
HW1 - Report



MLP
(Trained with Mini-Batch GD)

Training Loss: -0.6065 (Incorrectly calculated)
Validation Loss: -0.6192 (Incorrectly calculated)

Validation Class-Level Analytics:

	precision	recall	f1-score	support
1	0.26	0.80	0.40	6071
2	0.26	0.04	0.07	5920
3	0.23	0.18	0.20	5980
4	1.00	0.00	0.00	6029
accuracy			0.26	24000
macro avg	0.44	0.25	0.17	24000
weighted avg	0.44	0.26	0.17	24000

```
[[4835  158 1078    0]
 [4306  238 1376    0]
 [4655  231 1094    0]
 [4531  281 1216    1]]
```

Testing Class-Level Analytics:

	precision	recall	f1-score	support
1	0.27	0.81	0.40	1900
2	0.31	0.05	0.08	1900
3	0.24	0.19	0.21	1900
4	0.00	0.00	0.00	1900
accuracy			0.26	7600
macro avg	0.21	0.26	0.17	7600
weighted avg	0.21	0.26	0.17	7600

```
[[1548   48  304    0]
 [1350   90  460    0]
 [1471   63  366    0]
 [1435   86  379    0]]
```

The MLP performance was definitely the most surprising; I expected it to perform the best of the three models, in terms of both accuracy and time complexity, based on my studies. However, something definitiely went wrong with my algorithm, particularly when calculating the gradients, because the resulting estimator is clearly biased. The high recall and low precision indicates this, as this trend is typically seen with models trained on unbalanced datasets.

The mystery with my model is that dataset perfectly balanced, at least before the training/validation split. Perhaps the training/validation split caused this issue, although that is unlikely. More likely, there is some calculation that causes the gradient for one class to be updated more than the others.

Despite the high recall, when averaged, the model still performs in the range of the ‘random guess’ standard. For that reason, I would not consider this to be a usable classifier.