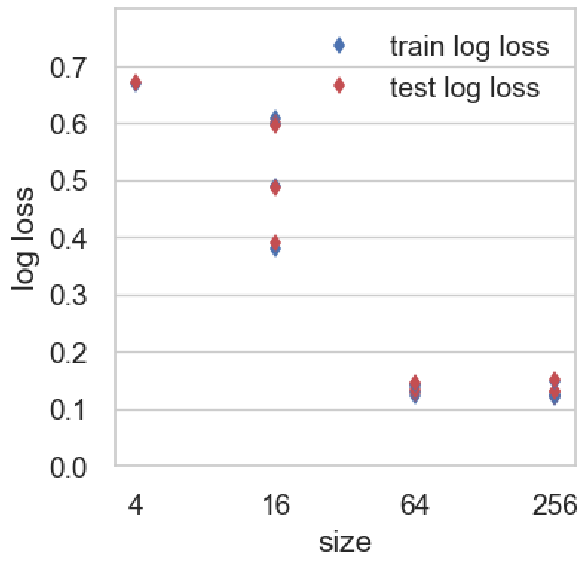
Homework 3

Matthew Carey

Problem 1:



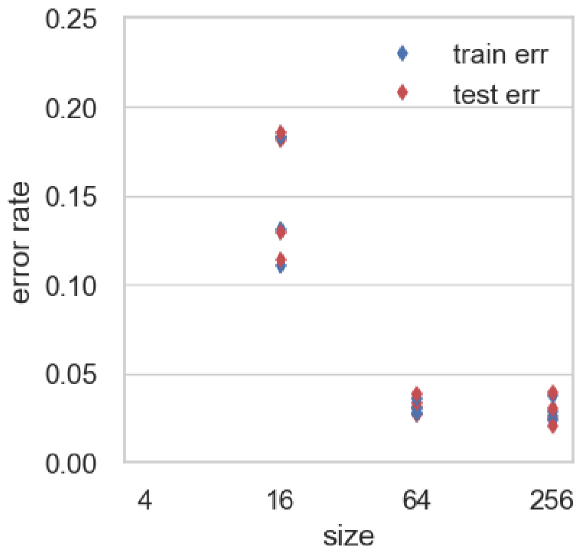


Figure 1: log loss vs layer size (left) and error rate vs layer size (right)

a)

```
Standard Deviation of Test Log Loss:
Size
4       0.133109
16      0.099598
64      0.008928
256     0.012594

Standard Deviation of Test Error Rate:
Size
4       0.072602
16      0.036245
64      0.004608
256     0.007846
```

Upon analyzing the standard deviations across multiple random runs for various sizes of the internal layer, it becomes apparent that the test data shows reduced susceptibility to random initializations with larger layer sizes, up until a size of 64. However, beyond a size of 64, there's a slight increase in susceptibility to random initializations. Interestingly, this trend is mirrored in the training data as well. Specifically, the variance in log loss in the test data is lowest with a size of 64, while it becomes slightly larger with a size of 256.

b) Based on figure 1, to achieve the best log loss performance on heldout data I would recommend using a hidden layer size of 64. This is because while the lowest log loss achieved was with a size of 256, some runs with 256 had a log loss greater than any of the log losses of 64. The size of 64 had a significantly smaller standard deviation, meaning that the results were more consistent, likely due to the fact that a size of 256 was overfitting.

c) Similarly to part b, to minimize error rate on heldout data, I would recommend using a hidden layer size of 64. This is because while 64 and 256 have very similar test error rates, the standard deviation for 64 is significantly less than it is for 256. This means that the results we see are much more consistent with a size of 64, and even though some of the results for 256 were lower than for 64, some results from 256 were also higher than all results from 64. Thus, to reliably achieve the lowest error rate on heldout data, a hidden layer size of 64 is optimal.

Problem 2:

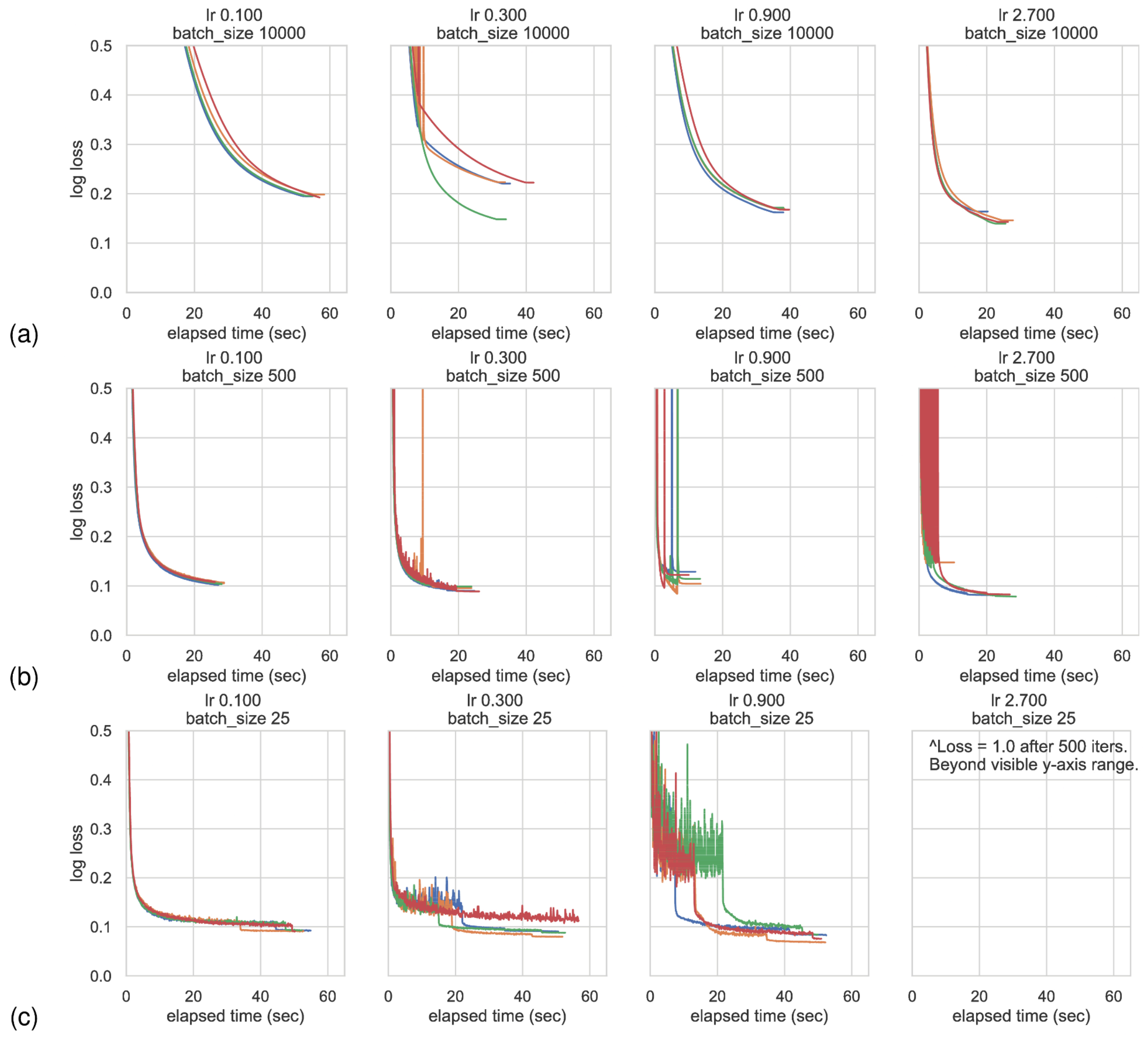


Figure 2

a) In reference to figure 2: (a) for a batch size of 10000, the largest learning rate that produces good results quickly and consistently across at least 3 runs is a learning rate of 2.7. In a shorter time than any other learning rate, all runs reach the lowest log loss of any other learning rate. (b) For a batch size of 500, the largest learning rate that still produces good results quickly and consistently among at least 3 runs is also 2.7. This is because although the orange run converges at a higher value, the other 3 all converge to a log loss of < 0.1. Despite the red run oscillating, it still converges to a stable value that is consistent with the other runs in a short time. (c) For a batch size of 25, the largest learning rate that still produces good results quickly and consistently among at least 3 runs is 0.3. Although the red run continues oscillating and does not converge to a consistent value, the other three runs all converge to a stable value of log loss < 0.1. It should be noted however, that for a learning rate of 0.1, all 4 runs converge much more consistently to a similar log loss and in similar time.

b) For batch size (a), the method only reached a loss of 0.15 consistently after 30 seconds. For batch size (b), the method reached a loss of 0.1 after approximately 10 seconds. For batch size (c), the method reached a loss of 0.1 after approximately 20 seconds.

c) The batch size of 500 achieved a loss of 0.1 after approximately 10 seconds, which is faster than both the batch size of 10000 and 25. Larger batch sizes like 10000 take longer to process each iteration because they require more data to be processed simultaneously. As a result of this, the optimization process takes longer, even though each iteration covers more data points. On the other hand, a small batch size like 25 will process fewer data points per iteration, leading to faster iterations, yet potentially slower convergence due to noisy updates. The choice of batch size involves tradeoffs between computational efficiency and optimization stability. Larger batch sizes generally lead to more stable updates but slower convergence, while smaller batch sizes enable faster updates but may suffer from noisy gradients and slower convergence. The batch size of 500 strikes a balance between the advantages of larger and smaller batch sizes. It processes a sufficient number of data points per iteration to provide stable updates while still maintaining computational efficiency, resulting in faster convergence to a good model compared to both larger and smaller batch sizes.

Problem 3:

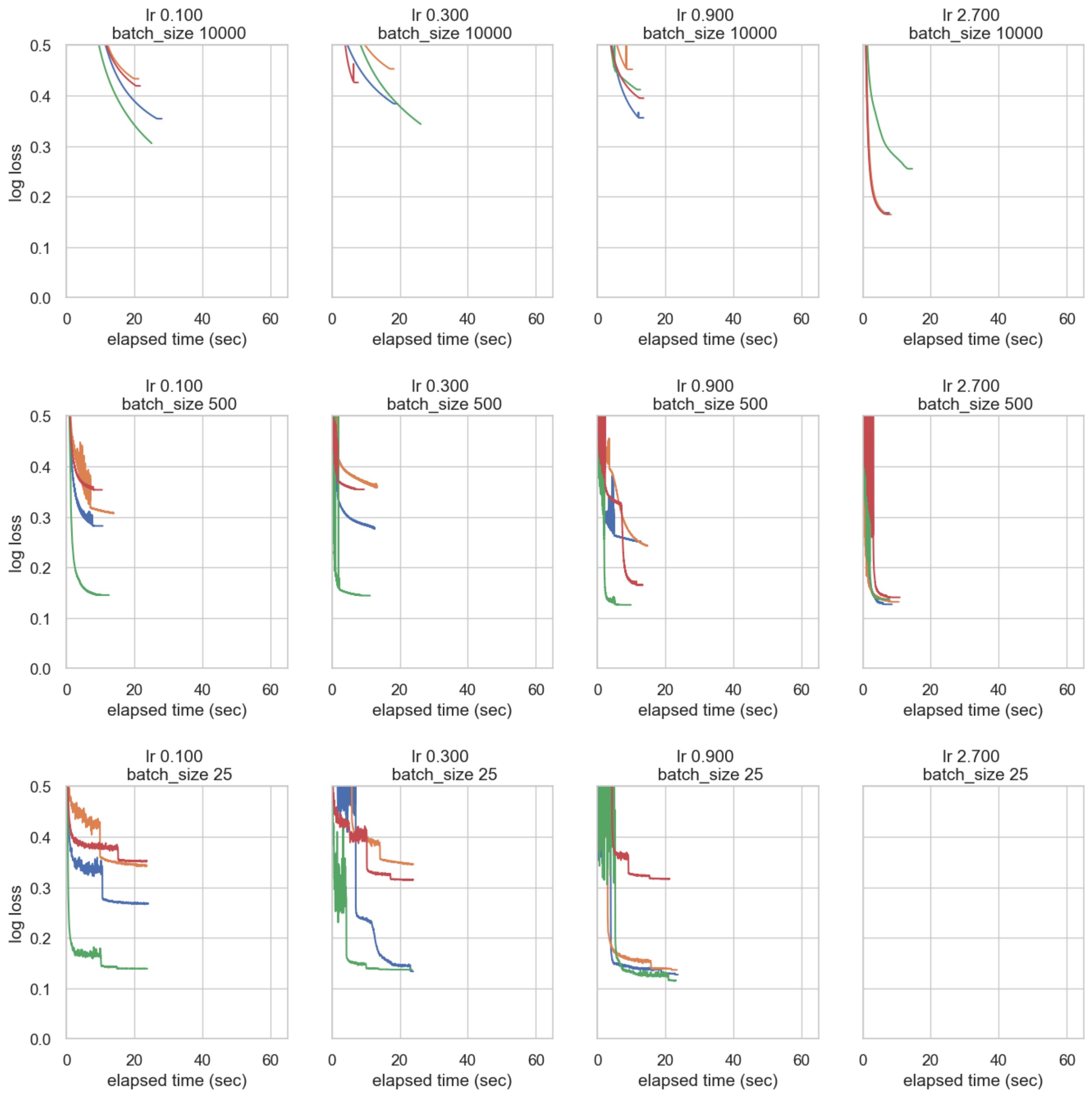


Figure 3A: Some key differences between figures 2 and 3A are seen in the batch size of 25, where the runs behave much more erratically in 3A compared to 2. For batch sizes of 10000 and 500, the largest learning rates that produced good, consistent results are the same, however another key difference is that the batch size of 10000 converges about as fast as the batch size of 500 does.