Lab 2: CoMS 573

Eric Murphy

March 3, 2017

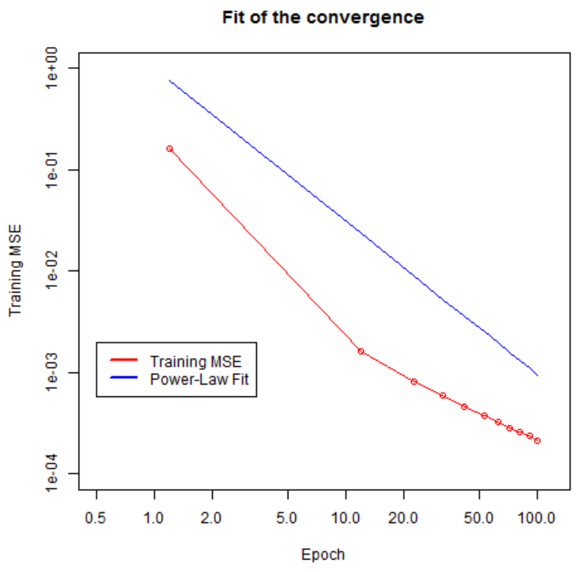
In this assignment we are tasked with tuning a neural network to identify handwritten digits, e.g. the number of hidden units and layers, and hyper-parameters such as learning rates and momentum rates. The sum-of-squares error and cross-entropy loss functions are also compared, alongside the activation functions, which are the hyperbolic tangent and ReLU function since I am using the h2o package with R.

My strategy in exploring this space is as follows. First, I have identified a combination of learning and momentum rates, i.e. learning rate=0.01 and momentum rate=0, which give stable solutions with the cross-entropy loss function and hyperbolic tangent. Next I seek to isolate, which combination of hidden units will yield the best solution. Note that the stopping time was not varied, because the backpropagation algorithm stops when the validation error begins to increase. However, the convergence rate of the MSE has been characterized at various epochs in the backpropagation algorithm.

Note that all cases *unless otherwise noted* were performed with the softmax function as an output, with the 1 of c encoding, without adaptive rates, with 20% of the training set used for validation, and with scaled input. See my h2o.deeplearning calls.

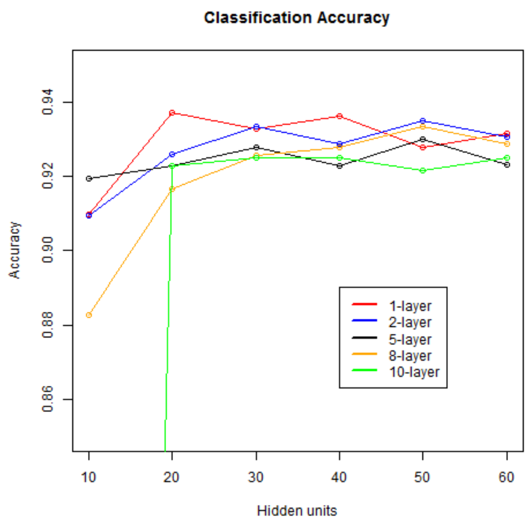
Convergence Rate:

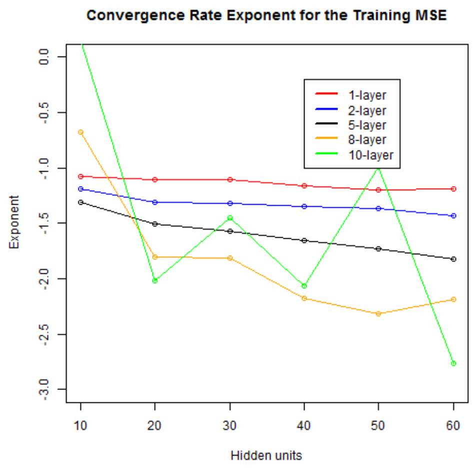
The convergence rate is extracted by fitting a power law to the scoring history, i.e. training_mse=f(epoch), output from the h2o.deeplearning function. This yields a scalar for quick comparison of the convergence rate of the loss function. See plot below for the cross-entropy loss function with Tanh activation function. Model has 2 hidden layers with 50 hidden units each. The observed power-law follows a power of -1.50878883843842, which means the training mean-squared error converges faster than linear scaling. We shall talk about convergence rates in terms of these exponents. Note that these measures are not quite perfect but they are easily automated. More negative exponents converge faster.

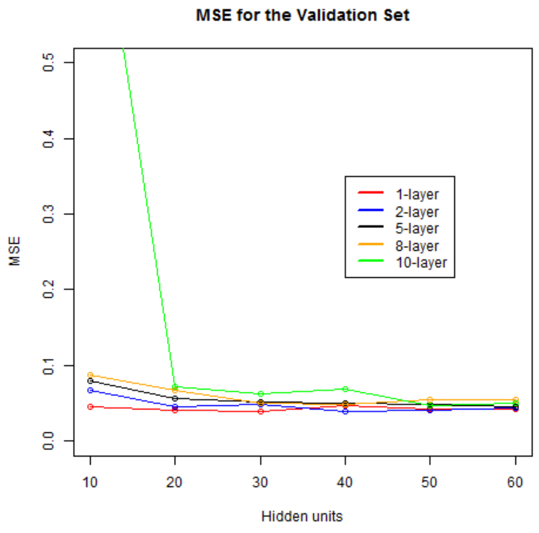


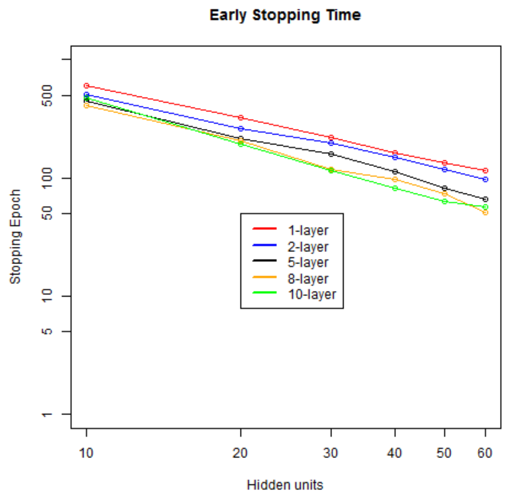
Cross-Entropy w/ tanh: Finding best number of hidden layers and units

The way to choose the proper number of hidden layers and units here is to choose the best accuracy. Failing that, we must choose the least complexity i.e. number of hidden layers and units. Smaller stopping times are also preferred. I have chosen to alter test the number of hidden layers with (1,2,5,8,10) and the number of hidden units per layer as (10,20,30,40,50,60). The number of hidden layers is constant on every layer. Below we plot the class accuracy on the test set, the convergence rate exponent fit to the response, the MSE for the validation set, and the stopping time.





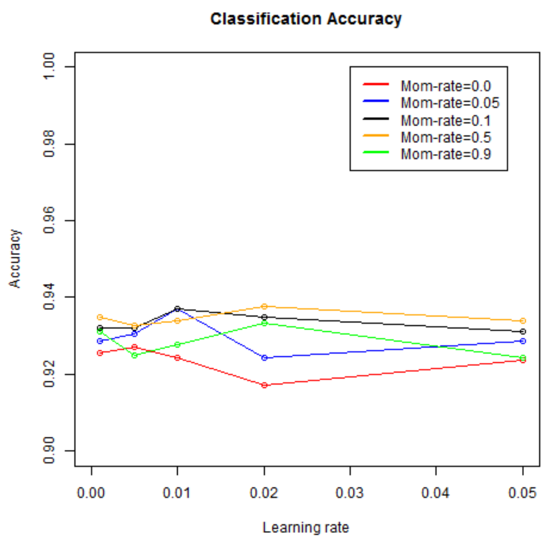


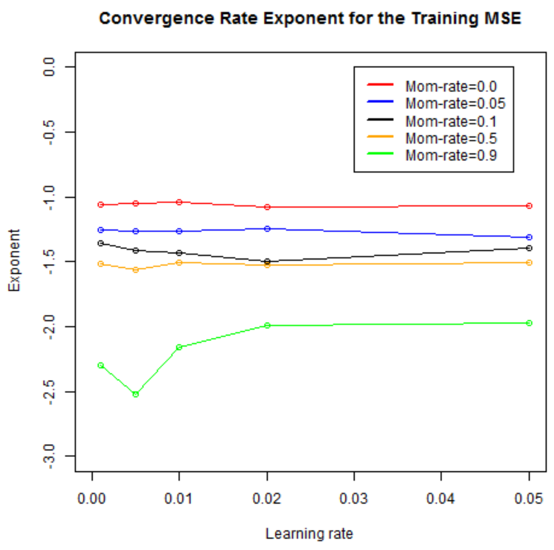


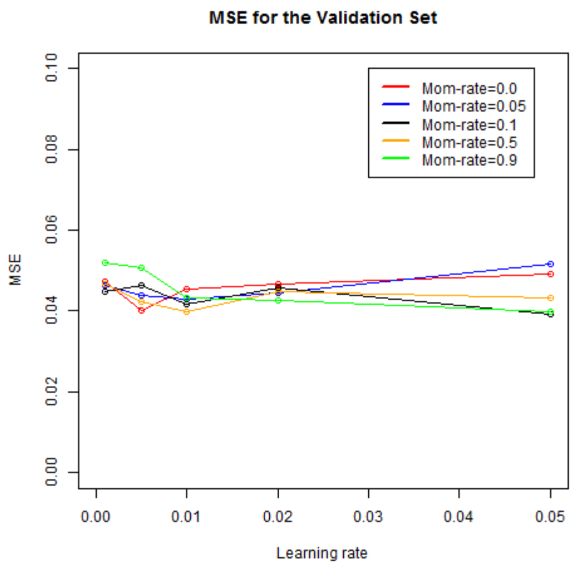
From these plots we see that the classification accuracy is pretty consistent, when there are at least 30 hidden units. I will choose **2 hidden layers with 50 hidden units**, which has good accuracy, small stopping time, and fewer fluctuations in the convergence rate exponent. The fluctuations in the response for systems with a larger number of layers leads me to believe that smaller learning rates may be required for such systems. For example, the response appears unstable with varying hidden units for 10 hidden layers. There was not sufficient time to explore this.

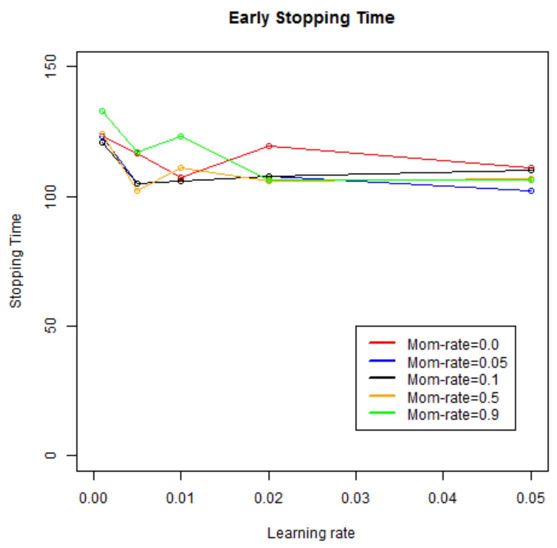
Cross-Entropy w/ tanh: Finding the best learning and momentum rates

Now we search for the best learning and momentum rates using 2 hidden layers and 50 hidden units. We now search for the best combination of learning and momentum rates among the values momentum rates in (0.0, 0.05, 0.1, 0.5, 0.9) and learning rates in (0.0001,0.005,0.01,0.02,0.05). Again, we are looking for the best class accuracy. Below we plot the class accuracy on the test set, the convergence rate exponent fit to the response, the MSE for the validation set, and the stopping time.





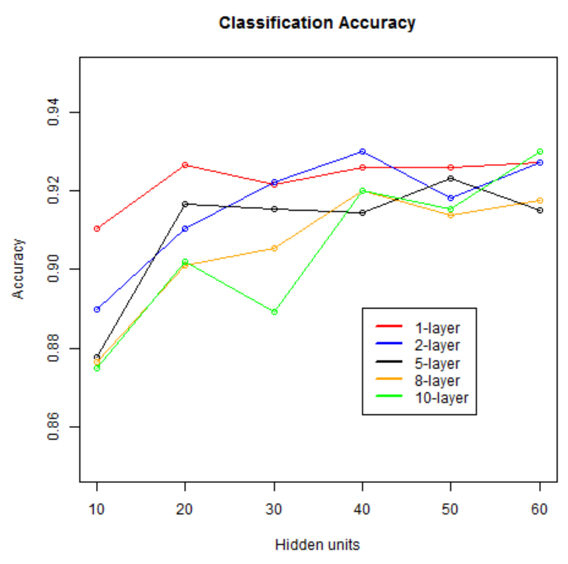


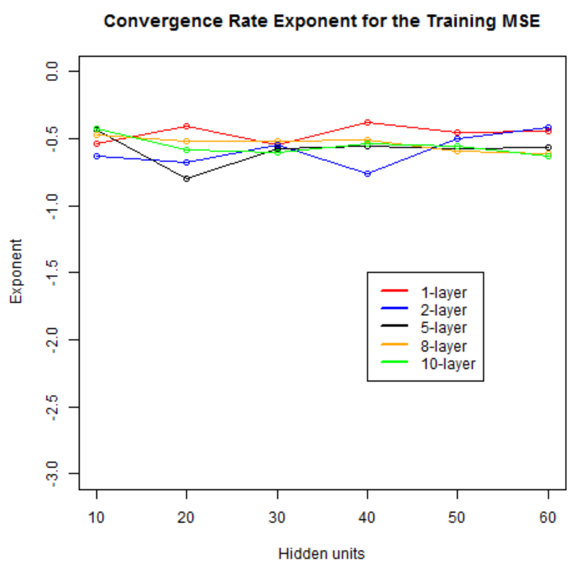


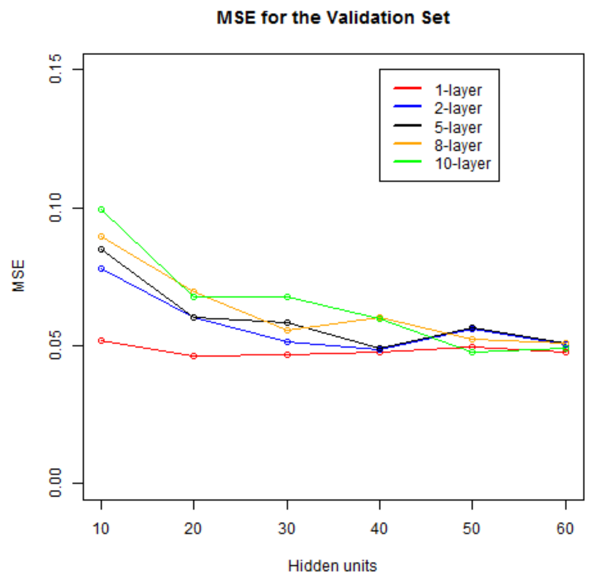
Here the Accuracy is much more easily chosen. It appears that the best combination from both the convergence rate and accuracy perspective is a **learning rate of 0.02 and a momentum rate of 0.5**. This gives the largest accuracy (~0.94) and one of the fastest convergence times, with and exponent ~-1.5. We do note that including momentum can gain us a near 2 percent gain in accuracy in this test. More info on the output of this model can be seen in the last section.

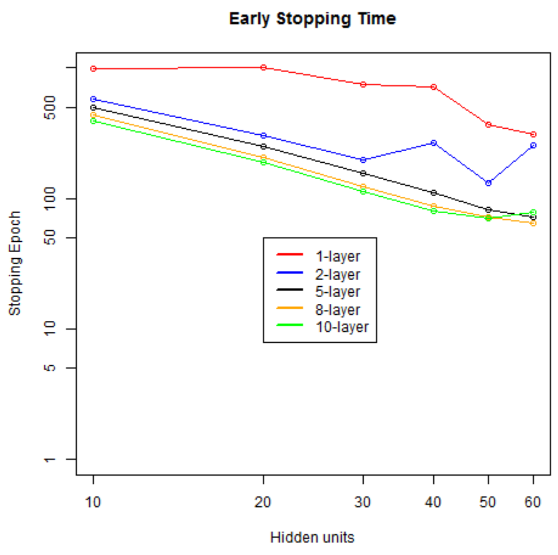
<u>Sum-of-squared-errors w/ tanh: Finding best number of hidden layers and units</u>

The same process was followed as outlined in the search for the number of hidden layers in the Cross-Entropy section. Below we plot the class accuracy on the test set, the convergence rate exponent fit to the response, the MSE for the validation set, and the stopping time.





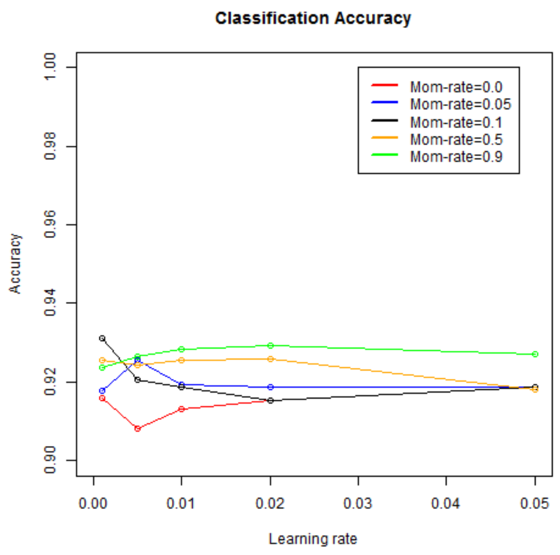


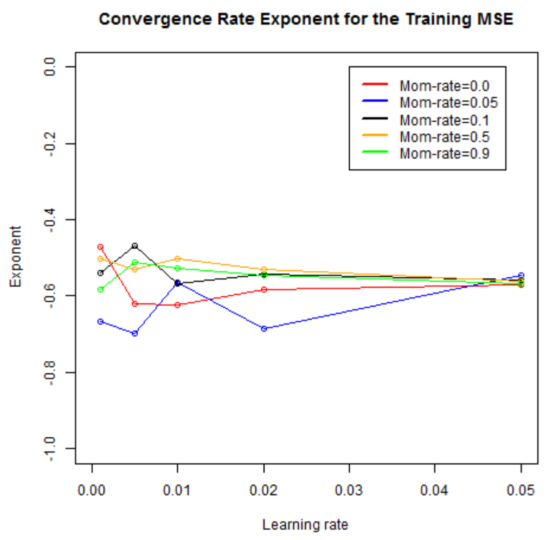


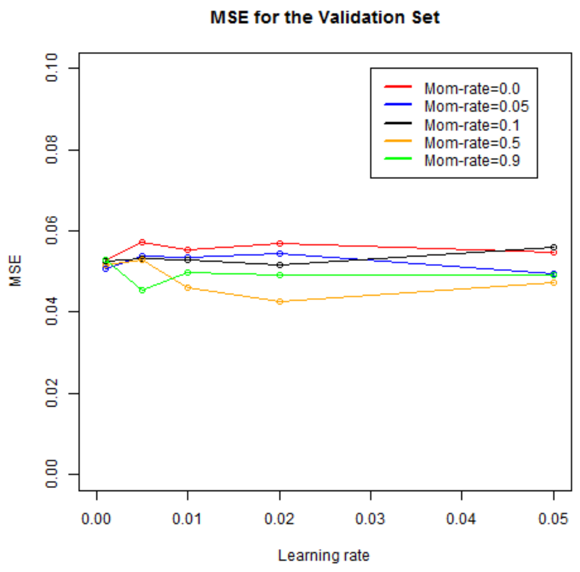
This case is easier to choose than in the cross-entropy case. From the class accuracy on the test set it appears that **<u>2-layers with 40 hidden</u>** units each is performs the best. Generically, we see much the same behavior between the cross-entropy and sum of squared errors. Interestingly, the exponents for the convergence rates for all cases appear to be very tightly distributed around $x = -0.5$. There were many more fluctuations observed in the cross-entropy case. In general, this exponent is smaller for the sum-of-squared errors, indicating that the **<u>cross-entropy measure has faster convergence</u>**.

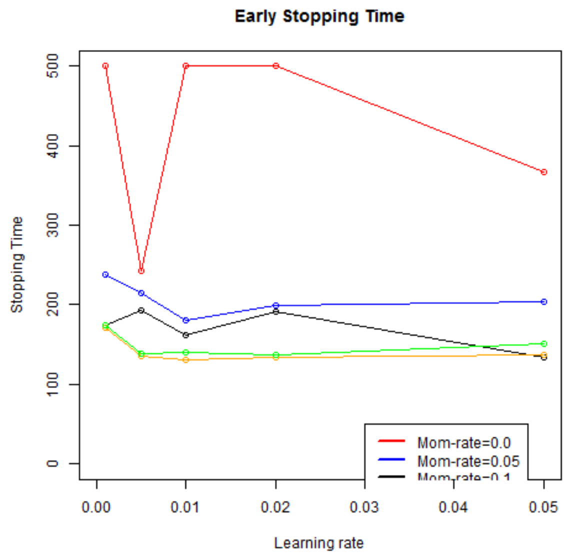
Sum-of-Squared-Errors w/ tanh: Finding the best learning and momentum rates

Now we search for the best learning and momentum rates using 2 hidden layers and 40 hidden units using the sum of squared errors as the loss function. We now search for the best combination of learning and momentum rates among the values momentum rates in (0.0, 0.05, 0.1, 0.5, 0.9) and learning rates in (0.001,0.005,0.01,0.02,0.05). Again, we are looking for the best class accuracy. Below we plot the class accuracy on the test set, the convergence rate exponent fit to the response, the MSE for the validation set, and the stopping time.





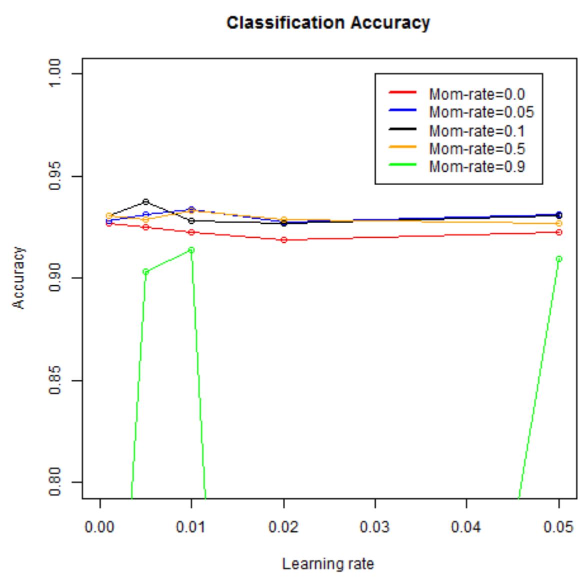


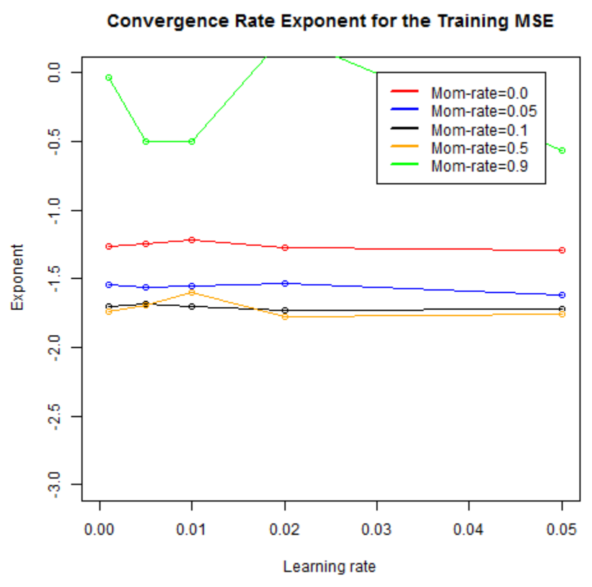


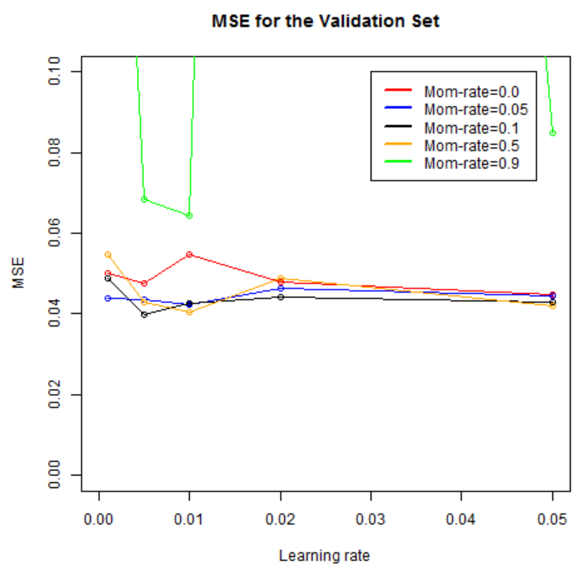
According to the accuracy plot, it appears that momentum only helps us. The final result is a **learning rate of 0.02 and a momentum rate of 0.9**. We note that it seems that the case without momentum required more epochs to properly train, since many of the data points give a stopping time of 500 epochs. We note that the accuracy is slightly less than what was achieved when using the Cross-Entropy loss functions, and the convergence exponents are all smaller in magnitude at around 0.5. Momentum is seen to improve each of these metrics.

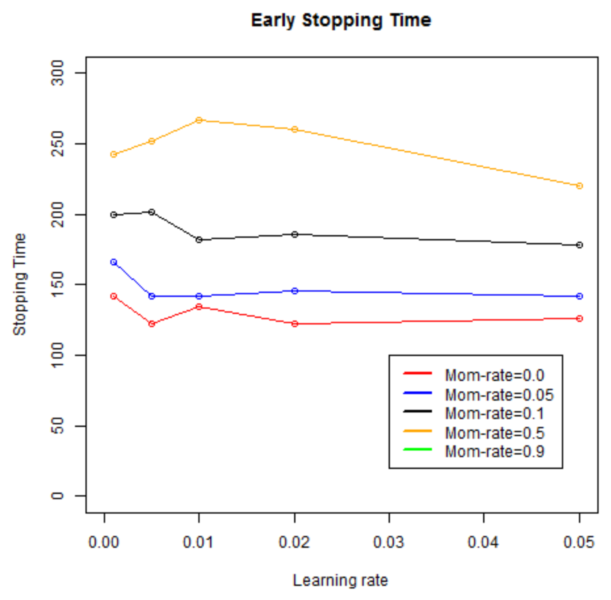
Cross-Entropy w/ ReLU: Finding the best learning and momentum rates

Now we search for the best learning and momentum rates using 2 hidden layers and 50 hidden units. We now search for the best combination of learning and momentum rates among the values momentum rates in (0.0, 0.05, 0.1, 0.5, 0.9) and learning rates in (0.001,0.005,0.01,0.02,0.05). Again, we are looking for the best class accuracy. Below we plot the class accuracy on the test set, the convergence rate exponent fit to the response, the MSE for the validation set, and the stopping time.









Note that we will ignore the green line, which appears to be unstable. This corresponds to the momentum rate of 0.9. This leads us to believe that too much momentum in the system may move the system past stable points too easily. Though momentum is seen to gain us accuracy in the class accuracy plot. Momentum also increases the convergence rate. Here it appears from the class accuracy plot that the best accuracy is obtained with a **momentum rate of 0.1 and a learning rate of 0.005**. This activation function seems to prefer both lower momentum and lower rates. This likely due to the fact that unlike the tanh, the ReLU function does not saturate at 1. Regardless, similar stopping times, accuracies, and convergence rates are achieved by both methods. More detailed information on the models will be discussed at the end.

<u>Input scaling:</u>

We make one comparison here to how input scaling affects the results. The cross-entropy with the tanh activation function was used with and without input scaling, all with the number of hidden layers, hidden units, momentum rate, learning rate, etc. The accuracies are below.

With scaling:

```
"Accuracy given scaled data"
0.930996104618809
```

Without scaling:

```
"Accuracy given unscaled data"
0.931552587646077
```

From this test, there does not appear to be much of a distance between whether data is scaled or not. In fact the without scaling produces slightly better accuracy, although it is not likely statistically significant. We note that there is only one ***type*** of input variable, that varies in intensity from 0 to 16. As such, the scale is not likely affect the results in this study.

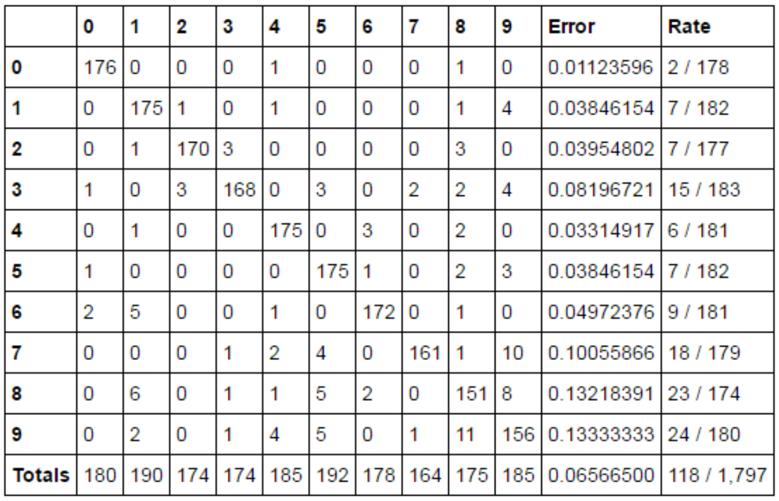
<u>Final Comparisons:</u>

Here we compare the results of all of the final models of the three main situations: Tanh w/ Cross-Entropy, ReLU with Cross-Entropy, and Tanh w/ Quadratic loss function.

Note that there is some stochasticism in the h2o.deeplearning function, as I always obtain slightly different answers.

<u>The output of the Tanh w/ Cross-Entropy:</u>

Confusion matrix:



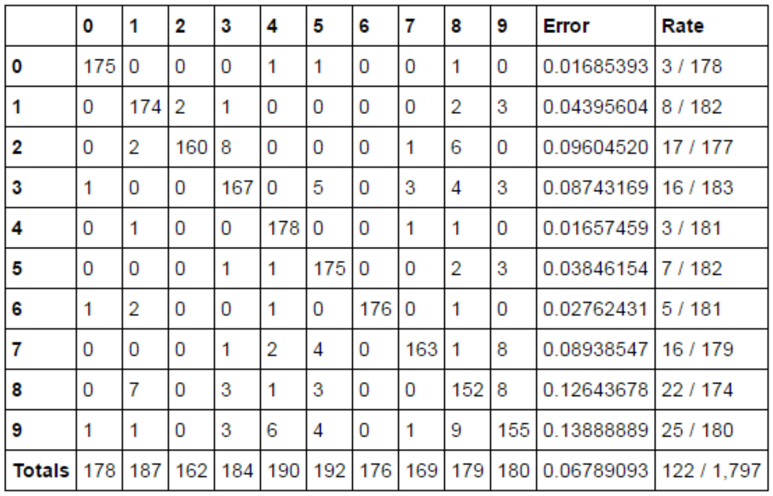
	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	176	0	0	0	1	0	0	0	1	0	0.01123596	2 / 178
1	0	175	1	0	1	0	0	0	1	4	0.03846154	7 / 182
2	0	1	170	3	0	0	0	0	3	0	0.03954802	7 / 177
3	1	0	3	168	0	3	0	2	2	4	0.08196721	15 / 183
4	0	1	0	0	175	0	3	0	2	0	0.03314917	6 / 181
5	1	0	0	0	0	175	1	0	2	3	0.03846154	7 / 182
6	2	5	0	0	1	0	172	0	1	0	0.04972376	9 / 181
7	0	0	0	1	2	4	0	161	1	10	0.10055866	18 / 179
8	0	6	0	1	1	5	2	0	151	8	0.13218391	23 / 174
9	0	2	0	1	4	5	0	1	11	156	0.13333333	24 / 180
Totals	180	190	174	174	185	192	178	164	175	185	0.06566500	118 / 1,797

Accuracy: 0.934335002782415

Convergence Exponent: -1.51550365499253

The output of the ReLU w/ Cross-Entropy:

Confusion Matrix:



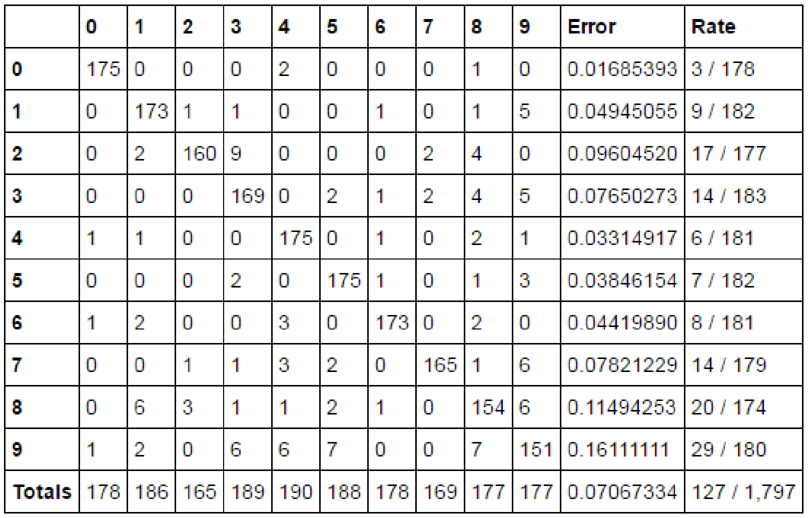
	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	175	0	0	0	1	1	0	0	1	0	0.01685393	3 / 178
1	0	174	2	1	0	0	0	0	2	3	0.04395604	8 / 182
2	0	2	160	8	0	0	0	1	6	0	0.09604520	17 / 177
3	1	0	0	167	0	5	0	3	4	3	0.08743169	16 / 183
4	0	1	0	0	178	0	0	1	1	0	0.01657459	3 / 181
5	0	0	0	1	1	175	0	0	2	3	0.03846154	7 / 182
6	1	2	0	0	1	0	176	0	1	0	0.02762431	5 / 181
7	0	0	0	1	2	4	0	163	1	8	0.08938547	16 / 179
8	0	7	0	3	1	3	0	0	152	8	0.12643678	22 / 174
9	1	1	0	3	6	4	0	1	9	155	0.13888889	25 / 180
Totals	178	187	162	184	190	192	176	169	179	180	0.06789093	122 / 1,797

Accuracy: 0.932109070673344

Convergence Exponent: -1.54576603825324

The output of the Tanh w/ Quadratic loss function:

Confusion matrix:



	0	1	2	3	4	5	6	7	8	9	Error	Rate
0	175	0	0	0	2	0	0	0	1	0	0.01685393	3 / 178
1	0	173	1	1	0	0	1	0	1	5	0.04945055	9 / 182
2	0	2	160	9	0	0	0	2	4	0	0.09604520	17 / 177
3	0	0	0	169	0	2	1	2	4	5	0.07650273	14 / 183
4	1	1	0	0	175	0	1	0	2	1	0.03314917	6 / 181
5	0	0	0	2	0	175	1	0	1	3	0.03846154	7 / 182
6	1	2	0	0	3	0	173	0	2	0	0.04419890	8 / 181
7	0	0	1	1	3	2	0	165	1	6	0.07821229	14 / 179
8	0	6	3	1	1	2	1	0	154	6	0.11494253	20 / 174
9	1	2	0	6	6	7	0	0	7	151	0.16111111	29 / 180
Totals	178	186	165	189	190	188	178	169	177	177	0.07067334	127 / 1,797

Accuracy: 0.929326655537006

Convergence Exponent: -0.50343659947291

Discussion:

Amazingly all 3 models come in with very similar accuracies. Note that the quadratic loss function does seem to produce consistently lower class accuracy on the test set than the two that use the Cross-Entropy Function. Additionally, while the cross-entropy versions have convergence exponents of ~ -1.5, the quadratic loss function always converges much slower. The convergence exponent for the quadratic loss function is always ~ -0.5. It seems that the only difference between the ReLU and Tanh functions is the sensitivity to learning and momentum rates. The minimization of the ReLU activation function is observed to behave wildly in areas of parameter space where the tanh function produces consistent results. It is surprising that although these approaches are very different similar results were always achieved not only in terms of overall accuracy but also more fine-grained error. For example, if we look at the confusion matrices for all three systems, the number 9 is most consistently miss classified, while the 0 is misclassified least often.

Please see the attached files to generate the plots above.