5 Extra Credit - Optimizing Network

I added early stopping, multiple layers, and an additional fine-tuning training cycle. I also did an exhaustive search for eta. My best accuracy was 55.24%, though the architecture did not always reach this high an accuracy. The architecture and training method could reliably reach an accuracy of 53.95%, which is a 2% improvement over 1 hidden layer of 50 nodes.

I first added early stopping, where the network keeps track of its best weights and returns that. Since this can only improve a network, I added it first. I then implemented the alternate activation function described in the next section. Since it overtrains very quickly, I tried it with hidden layers of size 30 and 40 instead, but this did not change the effectiveness.

I trained with early stopping, ReLu, all data, and the following hyperparameters:
`batch_size: 200, epochs: 10, hidden: [50 50], rho: 50, decay: 0.9`
for several values of eta. I searched for `eta` and `lambda` for the next few architectures, because I wasn't sure how changing the layers would change the optimal hyper parameters. I got the following results:

eta	lambda	accuracy	cost
6.226471e-02	1.543103e-13	5.314685e+01	1.371600e+00
5.455565e-02	7.246277e-12	5.354645e+01	1.320756e+00
8.256417e-02	9.705649e-12	5.374625e+01	1.341339e+00
8.599156e-02	9.526210e-13	5.374625e+01	1.363473e+00
5.155291e-02	5.989056e-12	5.464535e+01	1.317072e+00

The best accuracy was 54.6%. I ran the same architecture, but with 30 nodes in the second layer. This performed worse.

eta	lambda	accuracy	cost
7.902129e-02	1.019798e-12	5.304695e+01	1.360670e+00
5.544915e-02	1.036286e-11	5.304695e+01	1.365088e+00
7.343569e-02	6.439861e-12	5.324675e+01	1.340537e+00
9.268782e-02	8.139354e-13	5.344655e+01	1.333692e+00
5.075492e-02	1.489555e-14	5.374625e+01	1.330342e+00

I then ran the architecture with three layers of 50 nodes:

eta	lambda	accuracy	cost
7.769074e-02	1.679157e-11	5.354645e+01	1.329145e+00
6.617725e-02	1.266021e-14	5.354645e+01	1.366043e+00
5.581979e-02	2.417803e-12	5.384615e+01	1.346099e+00
5.709754e-02	1.012062e-12	5.414585e+01	1.340506e+00
7.319992e-02	2.264835e-13	5.414585e+01	1.354794e+00

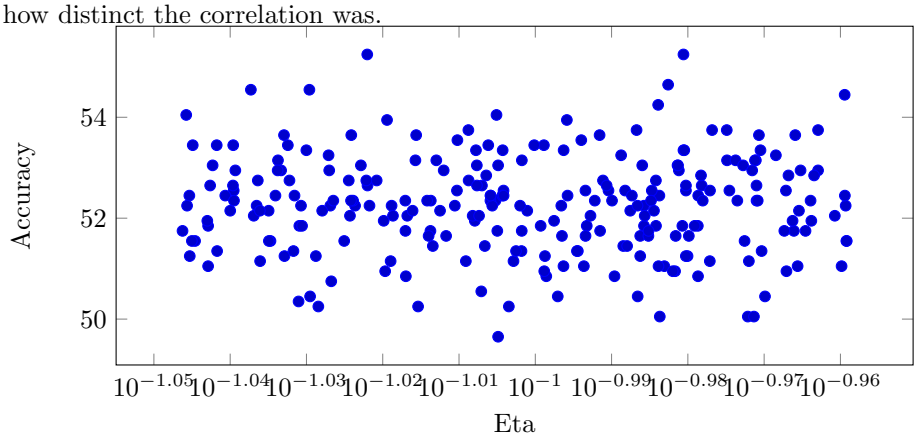
The article mentioned alternating large and small layers, so I tried 50 nodes, 75 nodes, and then 50 nodes again:

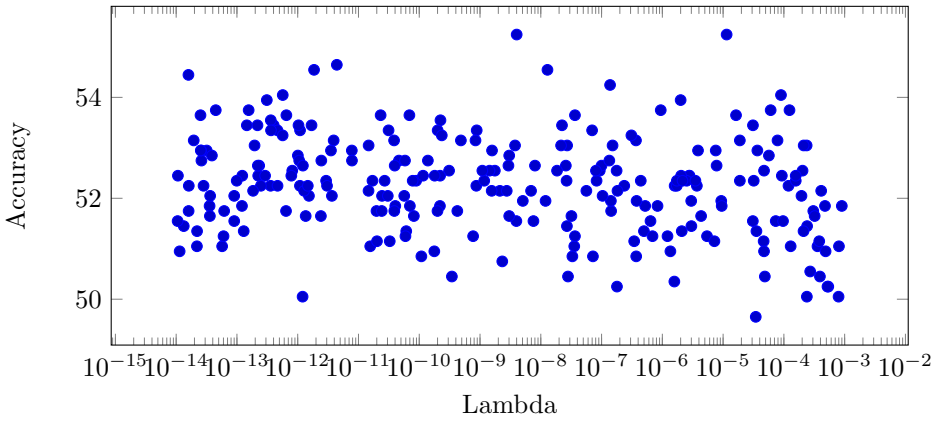
eta	lambda	accuracy	cost
9.473716e-02	1.902223e-14	5.314685e+01	1.339430e+00
8.879906e-02	5.986626e-11	5.324675e+01	1.339534e+00
1.131646e-01	8.952013e-14	5.324675e+01	1.339695e+00
1.034871e-01	1.551209e-14	5.354645e+01	1.348949e+00
1.060341e-01	1.985480e-14	5.484515e+01	1.367991e+00

Then I tried 50 nodes, 100 nodes, and then 50 nodes again. This had the best performance. I conducted another very fine search between 0.09 and 0.11 and got a max accuracy of 55.24%:

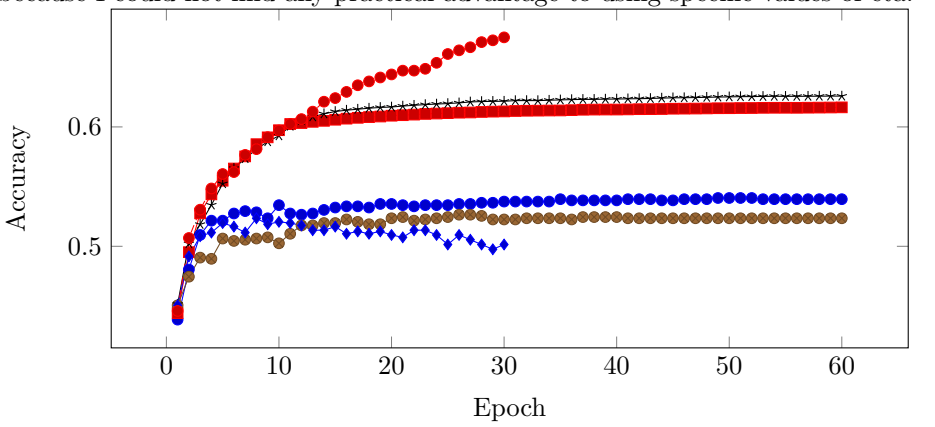
eta	lambda	accuracy	cost
1.099213e-01	1.591922e-14	5.444555e+01	1.348196e+00
9.353371e-02	1.857539e-12	5.454545e+01	1.333234e+00
9.189513e-02	1.287525e-08	5.454545e+01	1.341914e+00
1.042203e-01	4.401982e-12	5.464535e+01	1.387249e+00
9.517717e-02	4.009366e-09	5.524476e+01	1.323682e+00
1.047042e-01	1.142718e-05	5.524476e+01	1.350019e+00

Since this network had the highest performance, and a good bit of variance in values for lambda, I plotted lambda vs accuracy and eta vs accuracy to see how distinct the correlation was.





Each range of values for lambda and eta has a lot of variance, and when I trained with any of these values I found that I could not reliably get a high accuracy. However, the paper given described pre-training with a large eta (like I have done here) and then training again with a very small eta that is several places smaller. So, I pre-trained with `eta=0.1` and `lambda = 1e-12`, and then fine-tuned with `eta=1e-5` and `eta=1e-4` for an additional 50 epochs. I used `eta=0.1` because I could not find any practical advantage to using specific values of eta.



I got the most reliable and best results with an initial eta of 0.1 and a fine-tune eta of $1e-5$. I achieved a test accuracy of 53.95%, and a test cost of 1.323132 with this method. For comparison, here's the results with just one hidden layer of 50:

eta	lambda	accuracy	cost
9.300533e-02	2.548970e-12	5.144855e+01	1.375621e+00
7.390973e-02	2.192645e-13	5.144855e+01	1.376336e+00
6.772294e-02	2.049641e-12	5.154845e+01	1.386340e+00
5.652835e-02	2.562800e-14	5.154845e+01	1.393548e+00
5.278486e-02	6.282998e-14	5.194805e+01	1.382188e+00

6 Extra Credit - Different Activation function

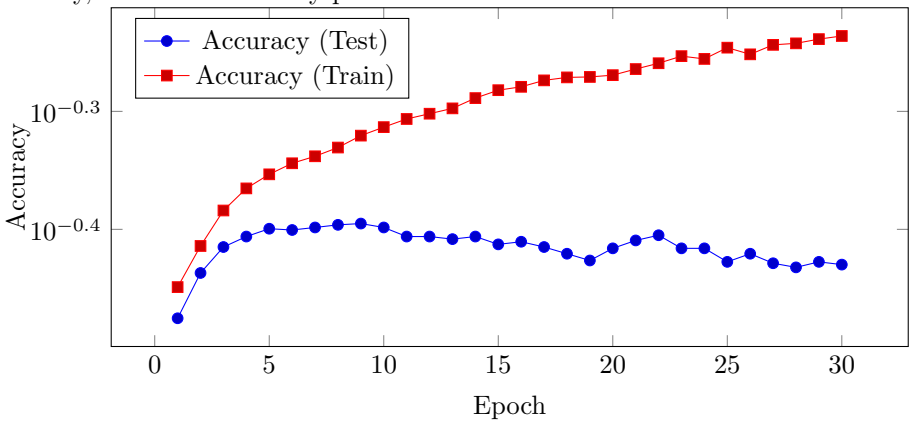
I used the activation function

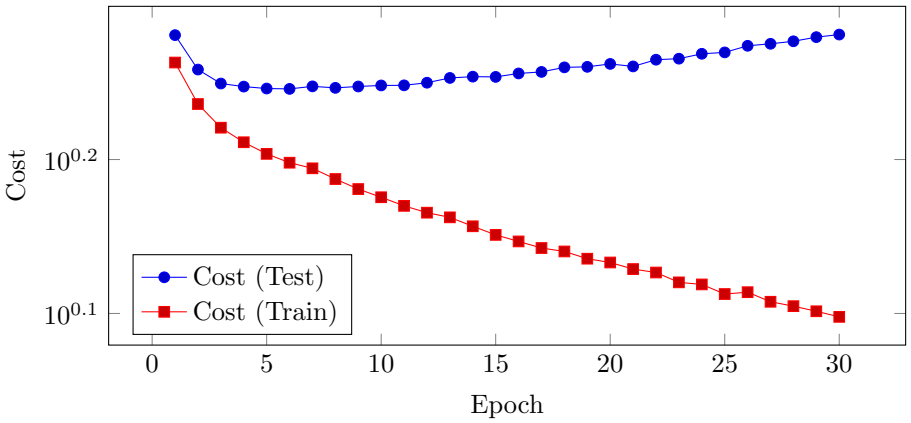
$$\varphi(x) = \frac{2}{1+e^{-x}} - 1$$

with derivative

$$\varphi'(x) = \frac{(1+\varphi(x))(1-\varphi(x))}{2}$$

with the following hyper-parameters:
`batch size: 200, eta: 0.0725, epochs: 30, lambda: 1.662508e-10, rho: 0.90, eta_decay: 0.95, nodes: 3072 50 10`
It trained much faster ReLu, but it also overtrained immediately after reaching its peak. With the above hyperparameters, it peaked in 9 epochs with a test accuracy of 40.3%. As seen below, training accuracy continues to increase steadily, while test accuracy plummits.




Cost
$10^{0.2}$
$10^{0.1}$
Cost (Test)
Cost (Train)
0
5
10
15
20
25
30
Epoch