6 Extra Credit - Extra optimizations

6.1 200 hidden nodes

I ran with 200 hidden nodes, searching 100 times for `eta` from 0.001 to 0.1 and `lambda` from $1e-1$ to $1e-15$, with the following other parameters:
`batch size: 200, epochs: 10, nodes: 200, rho: 0.90, eta_decay: 0.99`
These were the results, sorted by minimal test cost:

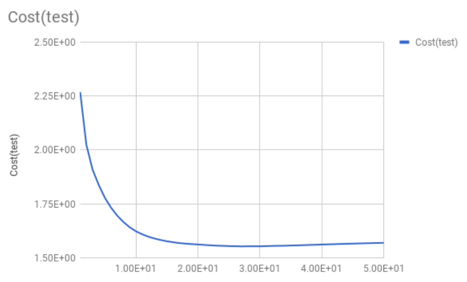
eta	lambda	accuracy	cost
3.14E-02	2.11E-07	4.44E+01	1.57E+00
3.53E-02	2.35E-12	4.42E+01	1.57E+00
3.59E-02	1.93E-12	4.49E+01	1.57E+00

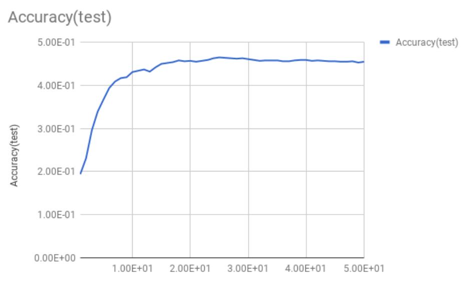
I thought that the regularization might increase if I increased the number of epochs, so I searched with 30 epochs with `eta` from 0.025 to 0.04, and `lambda` from $1e-6$ to $1e-12$. After 50 searches, these were the results:

eta	lambda	accuracy	cost
2.53E-02	1.52E-10	4.67E+01	1.67E+00
2.58E-02	1.07E-11	4.59E+01	1.68E+00
2.58E-02	3.89E-11	4.72E+01	1.68E+00

6.2 Early Stopping with 200 hidden nodes

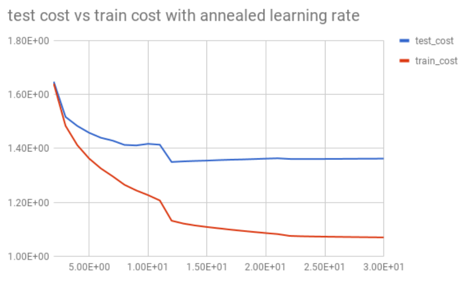
I ran the best settings from **6.1 200 hidden nodes** for 50 epochs and recorded the results, using the best weights to calculate the final accuracy. The final accuracy was 46.45% and the final cost was 1.554370.

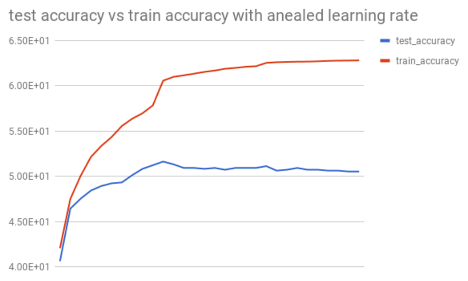




6.3 Annealed learning rate

I made a network that decreased `eta` by a factor of 10 after every 10 epochs. I used the following hyper-parameters, which is mostly from the previous section:
`batch size: 200 eta: 0.0449 epochs: 30 lambda: 4.419090e-09 nodes: 50 rho: 0.90 eta_decay: 0.10 eta_rate: 10`





The final results were:

test accuracy	test cost	train accuracy	train cost
5.16E+01	1.35E+00	6.06E+01	1.13E+00

7 Extra Credit - Alternate activation function

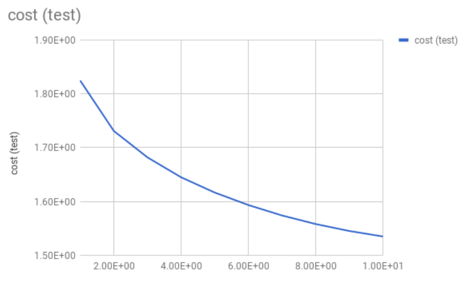
I used the activation function

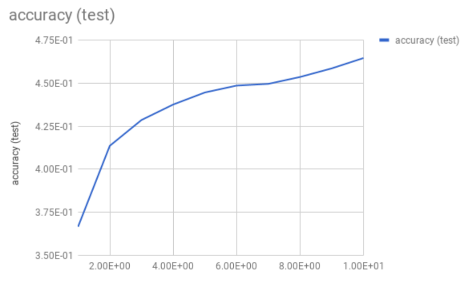
$$\varphi(x) = \frac{2}{1+e^{-x}} - 1$$

with derivative

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

with the following hyper-parameters:
`batch size: 200, eta: 0.0449, epochs: 10, lambda: 4.419090e-09, nodes: 50, rho: 0.90, eta_decay: 0.95`
It has these results:





With a final test accuracy of 0.4645355 and a final test cost of 1.5351. This is less effective than ReLu, but I also did not do the same hyper-parameter search. I suspect that, with proper hyper-parameters, this could perform comparably or better. Also, with longer training it may have performed better with these hyper-parameters, since the test cost had not yet plateaued / increased.