Deep Learning Lab 2 Report

Jonas Wechsler

April 2018

1 Analytic Gradient Computations

I found the max difference between my gradient computations and the analytic gradient computations for 10 random initializations of the matrix and 10 randomly selected vectors of size 100. I found that the max difference was almost always less than `1e-06`.

$\lvert W_{grad} - W_{slow}\rvert$	$\lvert b_{grad} - b_{slow}\rvert$	$\lvert W_{grad} - W_{fast}\rvert$	$\lvert b_{grad} - b_{fast}\rvert$
4.860548e-11	4.034763e-11	9.529103e-11	4.504980e-07
6.078456e-11	4.765436e-11	8.780116e-11	4.504921e-07
6.450508e-11	3.599582e-05	9.963655e-11	5.445306e-06
6.562769e-11	3.765227e-11	1.238103e-10	4.504952e-07
2.712996e-06	1.653451e-05	5.426026e-06	4.505854e-07
7.030398e-11	3.677468e-06	9.201002e-11	7.354927e-06
6.995059e-11	2.129719e-05	9.815890e-11	4.259442e-05
6.718896e-06	2.332479e-05	1.343774e-05	4.664962e-05
6.560693e-11	3.951748e-06	1.382000e-10	4.505262e-07
6.248607e-11	1.486871e-06	9.499379e-11	4.504766e-07

2 Momentum training

I found that momentum with $\rho = 0.9$ increased accuracy from 16.4% to 43.1% over 30 epochs with the following parameters: `batch size: 100, eta: 0.0100, lambda: 1.000000e-04, nodes: 50, eta_decay: 0.95`. Specifically, the model achieved an accuracy of `1.637778e+01` and a cost of `2.278824e+00` without momentum, and an accuracy of `4.313333e+01` with a cost of `1.600535e+00` with momentum.

3 Coarse search for lambda and eta

I searched with `eta` values from 0.001 to 0.1 and `lambda` values from $1e-10$ to $1e-5$, taking 100 samples, with 10 epochs per sample. I ran with the following hyper-parameters:

`batch_size: 200, epochs: 10, hidden: 50, rho: 0.9, decay: 0.95`
These 3 values for `eta` and `lambda` achieved the highest accuracy:

Eta	Lambda	Accuracy (%)	Cost
4.87E-02	8.60E-10	4.56E+01	1.62E+00
5.56E-02	3.33E-09	4.50E+01	1.65E+00
4.08E-02	3.46E-10	4.47E+01	1.62E+00

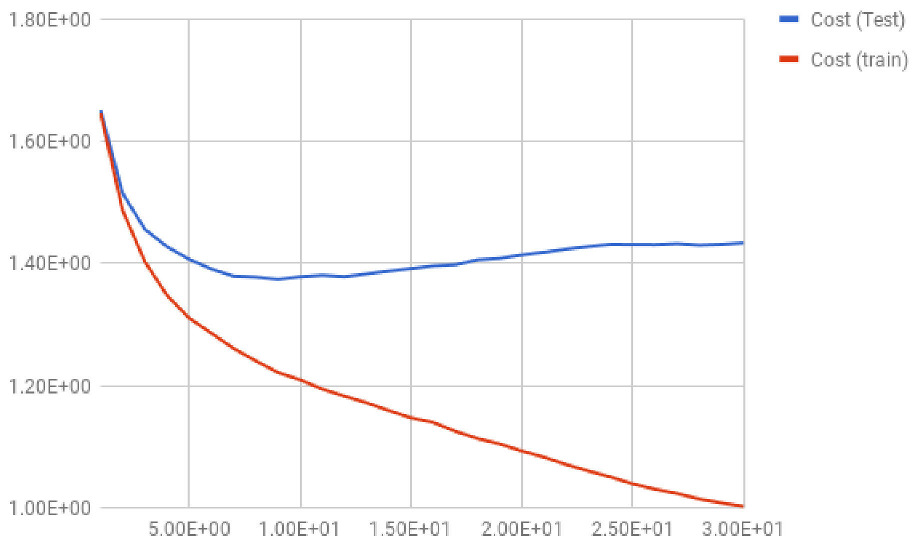
4 Fine search for lambda and eta

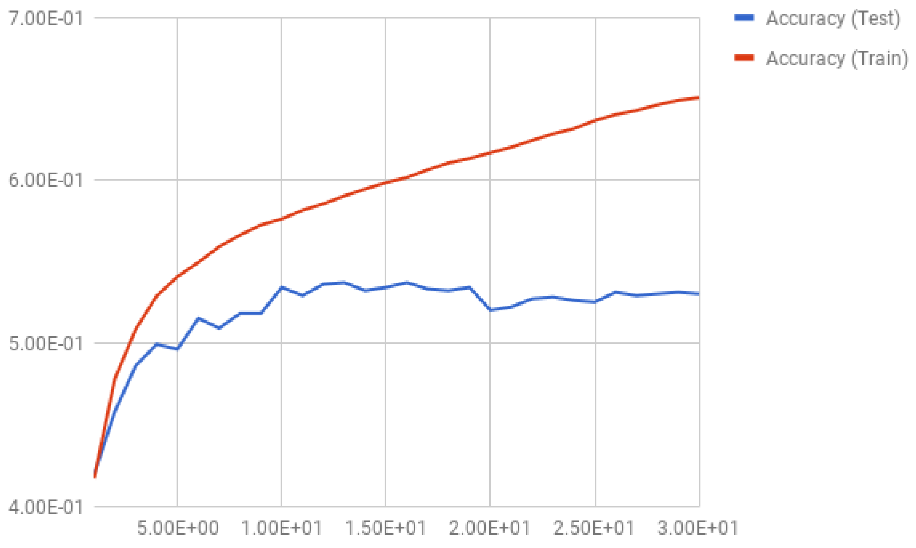
I searched with `eta` values from 0.04 to 0.06 and `lambda` values from $1e-20$ to $1e-8$, taking 100 samples, with 10 epochs per sample. I ran with the following hyper-parameters:
`batch_size: 200, epochs: 10, hidden: 50, rho: 0.9, decay: 0.95`
These 3 values for `eta` and `lambda` achieved the highest accuracy:

Eta	Lambda	Accuracy (%)	Cost
5.24E-02	1.66E-15	4.60E+01	1.62E+00
4.56E-02	1.48E-11	4.57E+01	1.60E+00
4.12E-02	4.70E-11	4.56E+01	1.60E+00

5 Training with all batches

I trained the network with the following hyper-parameters:
`batch size: 200, eta: 0.0449, epochs: 30, lambda: 4.419090e-09, nodes: 50, rho: 0.90, eta_decay: 0.95`
I chose `eta = 0.0449` because it yielded the lowest cost in the fine search, and the cost seemed to be less prone to noise when gathering data. The following are graphs of the cost and accuracy after each epoch for the test and training data.





These were the final results of the network:

Accuracy (Test)	Cost (Test)	Accuracy (Train)	Cost (train)
5.30E-01	1.43E+00	6.51E-01	1.00E+00

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:

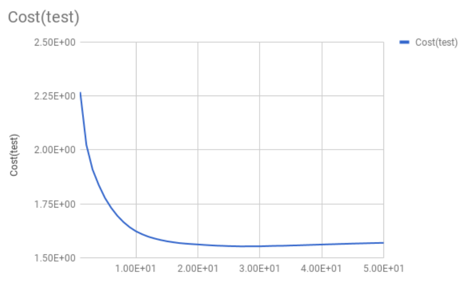
eta	lambda	accuracy	cost
3.14E-02	2.11E-07	4.44E+01	1.57E+00
3.53E-02	2.35E-12	4.42E+01	1.57E+00
3.59E-02	1.93E-12	4.49E+01	1.57E+00

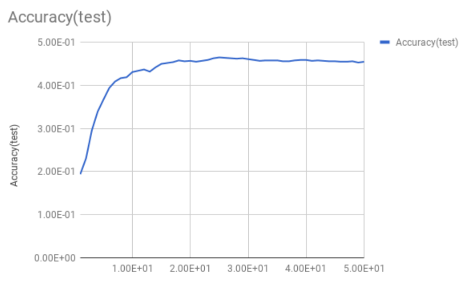
I thought that the regularization might increase if I increased the number of epochs, so I searched with 30 epochs with `eta` from 0.025 to 0.04, and `lambda` from $1e-6$ to $1e-12$. After 50 searches, these were the results:

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

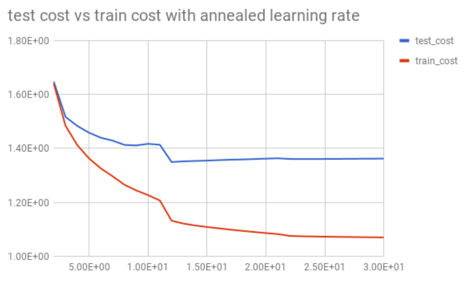
I ran the best settings from **6.1 200 hidden nodes** for 50 epochs and recorded the results, using the best weights to calculate the final accuracy. The final accuracy was 46.45% and the final cost was 1.554370.

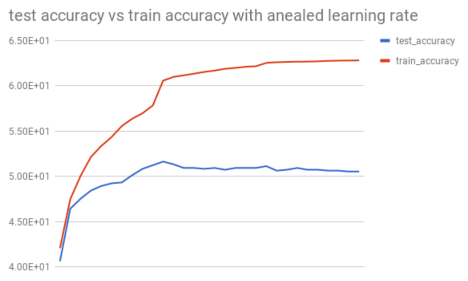




6.3 Annealed learning rate

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

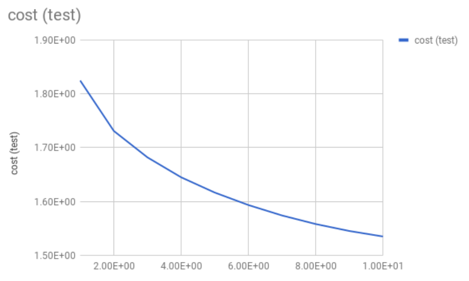
I used the activation function

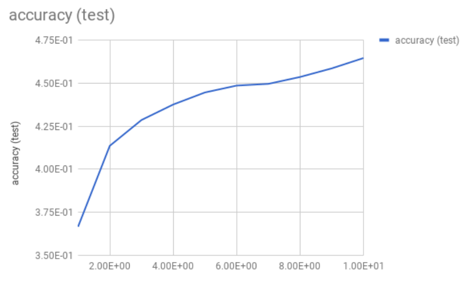
$$\varphi(x) = \frac{2}{1+e^{-x}} - 1$$

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.