CS168 Project 3

Henry Lin, Kaylee Bement

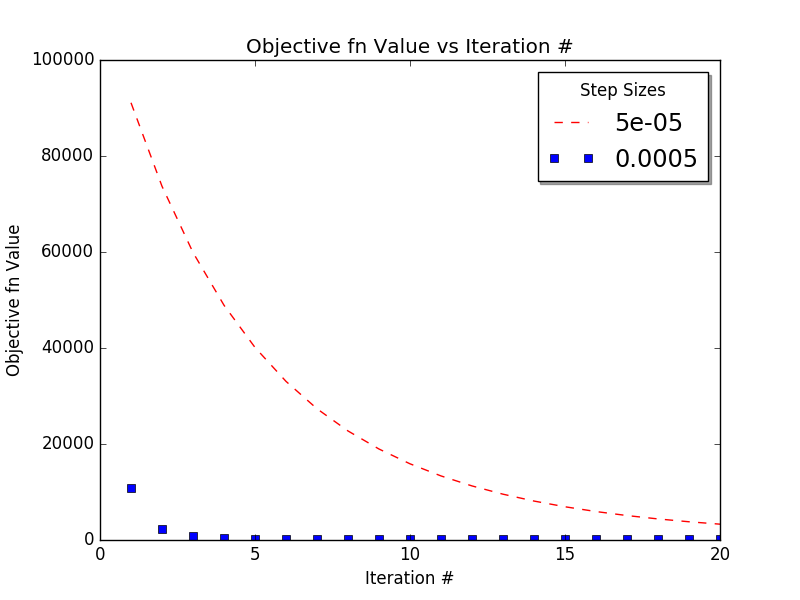
1.

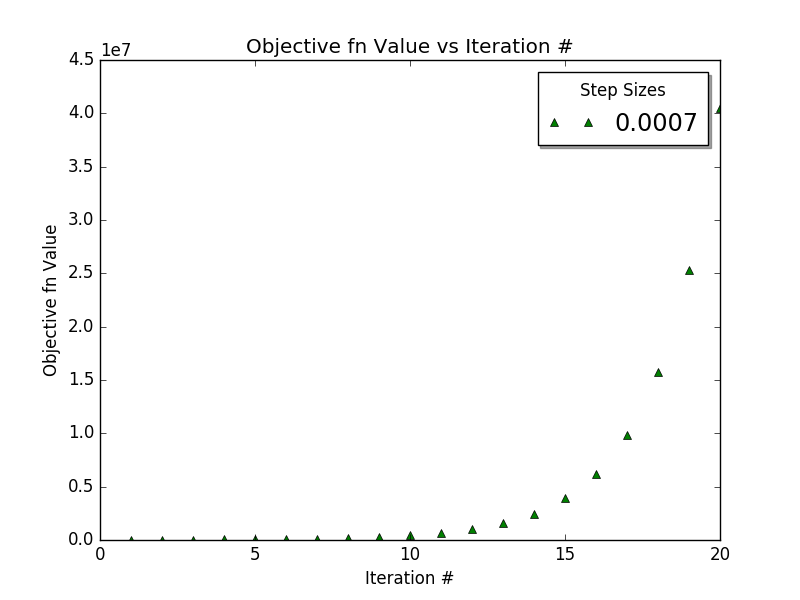
A. Objective function value: 220.34264027

Objective function value with a consisting of all 0s: 91827.65959497

Since there is randomness in some of the function values, these numbers fluctuate, but are close to the numbers given.

B. See code.py





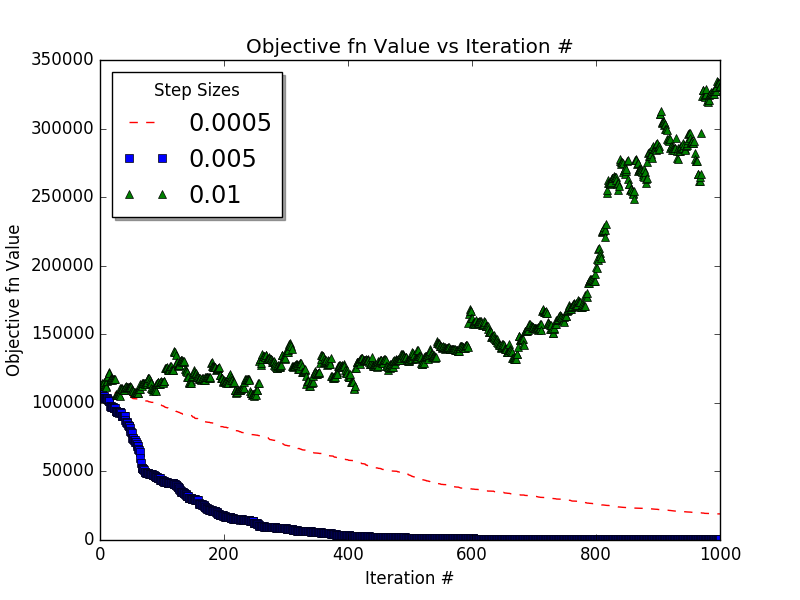
Gradient of f at at:

Optimal step size: 0.0005 with final objective function value of 233.09586745

The step size has a great impact on the convergence of gradient descent. If the step size is too small, gradient descent might be heading in the right direction, but never hit convergence, such as with step size = 0.00005, or it might find a local minimum and converge at the wrong value. If the step size is too large, gradient descent might overstep the minimum and step towards an incorrect local minimum, such as with step size 0.0007. However, step size 0.0005 seemed to work well, and gave us an objective function value close to the one in part A.

C. See code.py

Optimal step size: 0.005 with final objective function value of 499.53798953



Step size influences the convergence of stochastic gradient descent in the same way it affects regular gradient descent, but it takes more iterations for SGD to converge. Regular gradient descent has a better final value than SGD, but in SGD, each data point is only used once on average since there are 1000 iterations to choose a random point and there are 1000 data points, while in regular gradient descent, each data point is used 20 times since every point is used in each iteration.

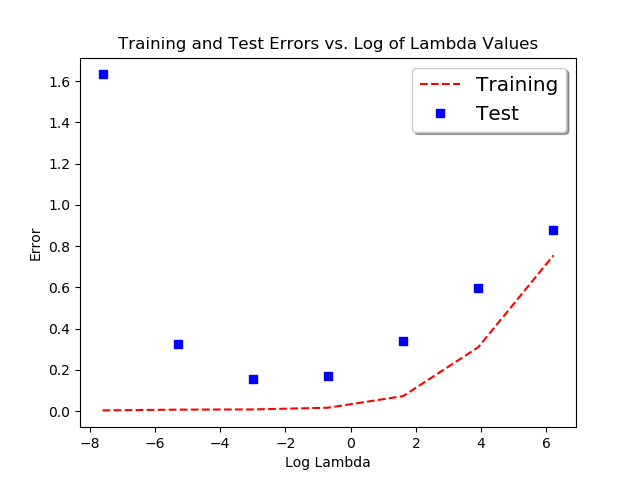
2.

A.

Avg train err: 1.08482770211125681e-14

Avg test err: 0.5823632339833625

B.



Based on the graph, we can see that different values of lambda will drastically affect the test error reported. From part A, we can see a drastic decrease in test error once we implement regularization at the cost of slightly higher training error. This tradeoff is necessary if we want the model to be able to generalize to an unknown dataset however. Not all values of lambda work however, so it is clear that lambda is a hyperparameter that needs to be tuned to have the optimal results. There seems to be an ideal around 0.005, 0.05 where the test error is at its lowest.

C.

Step size: 0.005

Average Training Error: 4.3481452074957776e-14

Average Test Error: 0.8165234129641803

Step size: 0.0005

Average Training Error: 4.423861977227671e-14

Average Test Error: 0.8165234129641805

Step size: 5e-05

Average Training Error: 5.3175481716220075e-14

Average Test Error: 0.8165234129641832

The different step sizes lead to marginally different training errors, but for the testing error, the error is almost identical for each step size used. In comparison to the model with l2 regularization, it does do better than some initializations of lambda, but compared to the optimal values, it does significantly worse. In comparison to part A, the SGD is able to fit the training data nearly as well as the baseline model, but we can see that for the testing, it has difficulty generalizing. This may be because of getting stuck in a local optimum and being unable to get out of that local optimum. TODO?

D.

TODO TODO TODO TODO

From the plots above, we can see the different step size makes in training the model. TODO

E.



From the plot, we can see that the best course of initialization for a is small weights. Larger weights seem to linearly increase both the training and test error. This may be because it is hard to affect larger weights with SGD and it is likely that the original initialization is far from the local optimum. This combined with a small step size of 0.00005 means the original initialization mostly remains with large r and affects the error.

3.

Average test error: 0.7062

We decided to use l2 regularization because the dimensionality is greater than the size of the training data, and we wanted to avoid overfitting. After trying lambdas ranging from 5e-8 to 5e2, we found that a lambda of 5e-6 worked best. We were disappointed that the test error was so high given the results we found in 2b, thus we believe that a better test error is achievable. However, the test error won’t be as low as the results in 2b given the challenge that our dimensionality is so high relative to the training dataset size.

Here is our pseudocode:

set λ to 0.000005

set training error and testing error to 0

for each trial:

pick a true, X train and test, and y train and test

set a to (X\_trainTX\_train + λI)-1X\_trainTy\_train

find current normalized train and test error for a

increment training and testing error by current train and test error

divide training and testing error by number of trials to find average training and testing error