بسمه تعالی

**دانشگاه شهید بهشتی**

**گزارش تمرین سری پنج­ام**

**درس علوم داده**

**سید حسین محمدی - 96222096**

خرداد 99

## Theory

Ques.1

1. Given the value of y and x, the Likelihood of x as input causing the same y as output for our model is the likelihood of a correct prediction that We want to maximize. Since we have a dataset, the likelihood of all this datapoints have to be multiplied which can create many computational problems such as rounding errors on computers, hard derivative and other.

If we log this product series, we will end up with a summation. The log function is also Monotonic which mean the learning will not be interrupted. Since we usually minimize, and we can try to minimize the negative of this log likelihood.

1. Both cause the model to be less dependent on the input. Meaning for small datasets or datasets with many outliers, L1 and L2 cause the output to change less (especially for outliers) while also reducing overfitting by adding bias to the model.

In L2, the change in output with respect to change in input (the derivative) is computed and then squared. This value is added to the cost function along with a penalty control parameter.

For L1, instead of squaring, we use the absolute value.

The main difference is that, using L2, since the new part of the cost function is squared, it will never reach zero and its gradient will slowly reach zero. But for L1, this is not the case and can essentially remove the influence of unimportant features completely.

1. Removes gradient overlap (back and forth) while amplifying real learning by tying new gradient value to the old. Causes faster convergence and less randomness

Ques.2

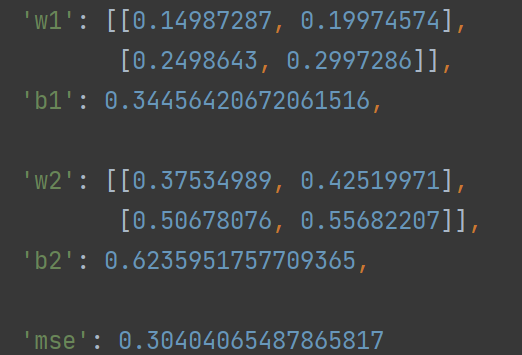
error after one feed-forward:



MSE error after one feed-forward:



New weights and biases:



Ques.3

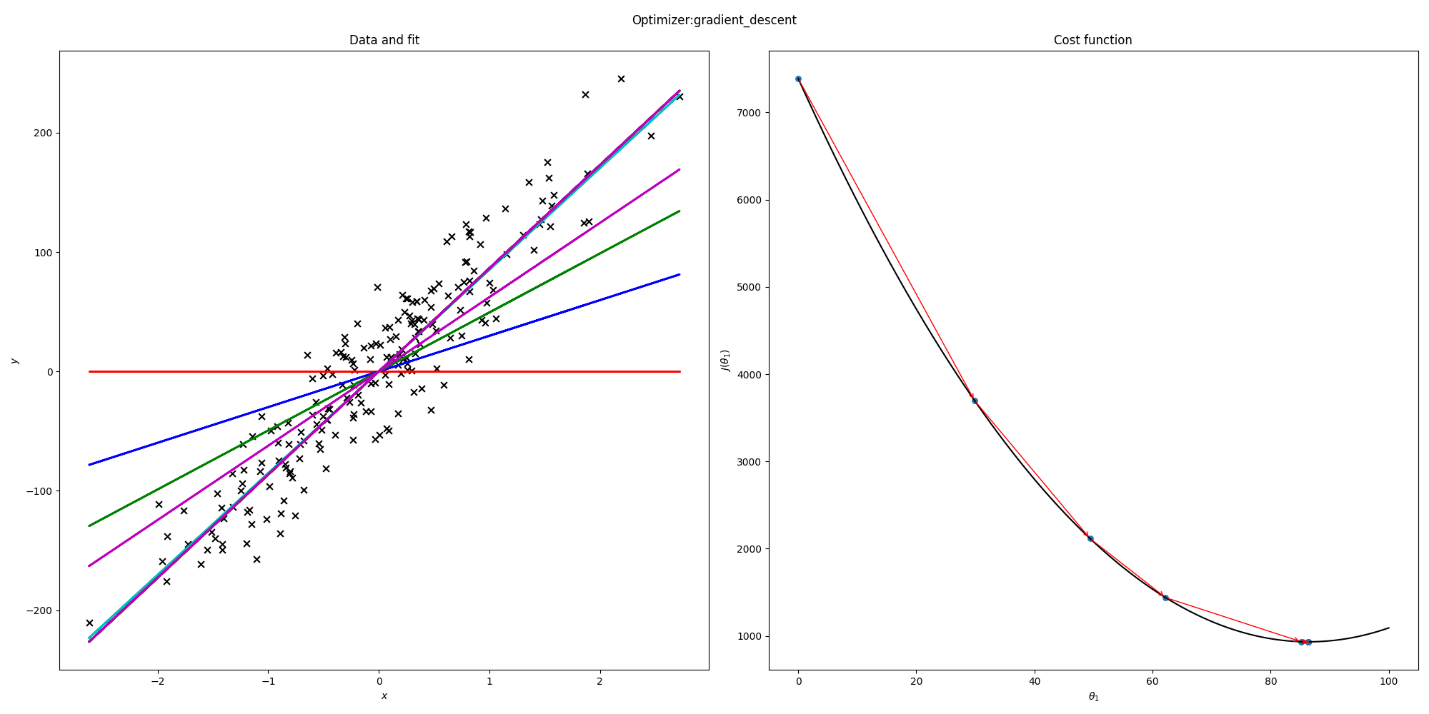
1. Better random init can help, but since the overall learning is taking too long, it might not have too great of an effect
2. Mini batch speeds up computation but at the cost of being at mercy of the data randomness. If a good batch size is chosen it can speed up computation without too much uncertainty
3. ✅Adam adds momentum and RMSprop which both help greatly and remove unnecessary back and froths in learning and allow for higher learning rate.
4. 🔯Will kill any learning since same weights cause the same gradient for all weights and causes the network to only learn one feature.
5. Increase learning rate can help but if not coupled with an optimization scheme, it most likely will worsen the problem

## Implementation

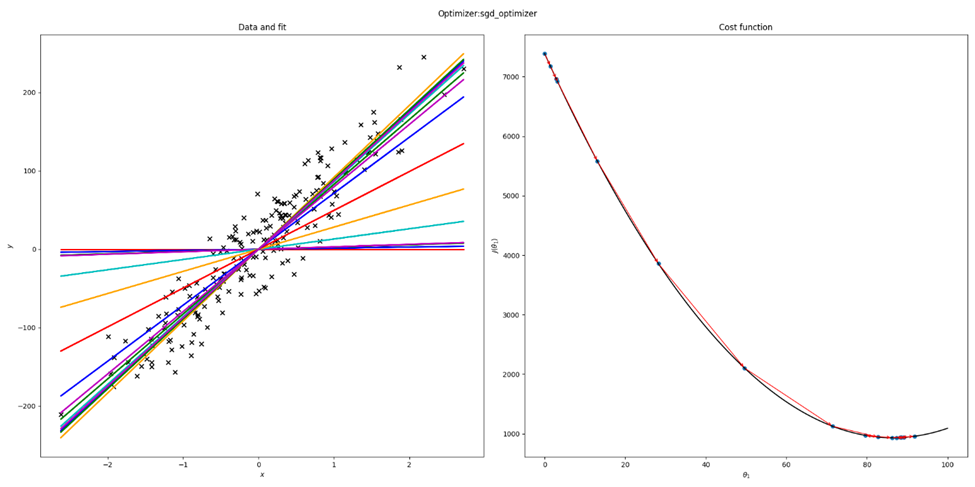
\*Gifs are placed in gif folder due to their big size

Below you can see the gradiant change for 500 iterations.

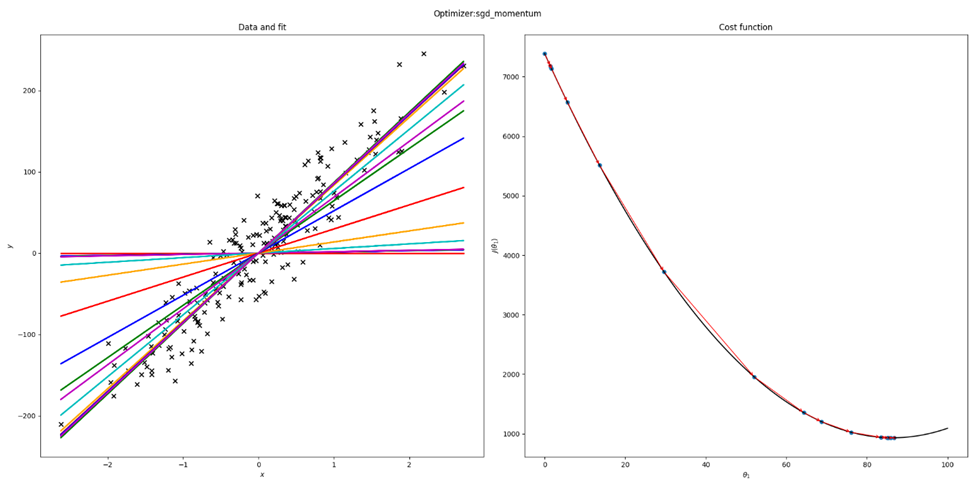
GD is the normal linear regression

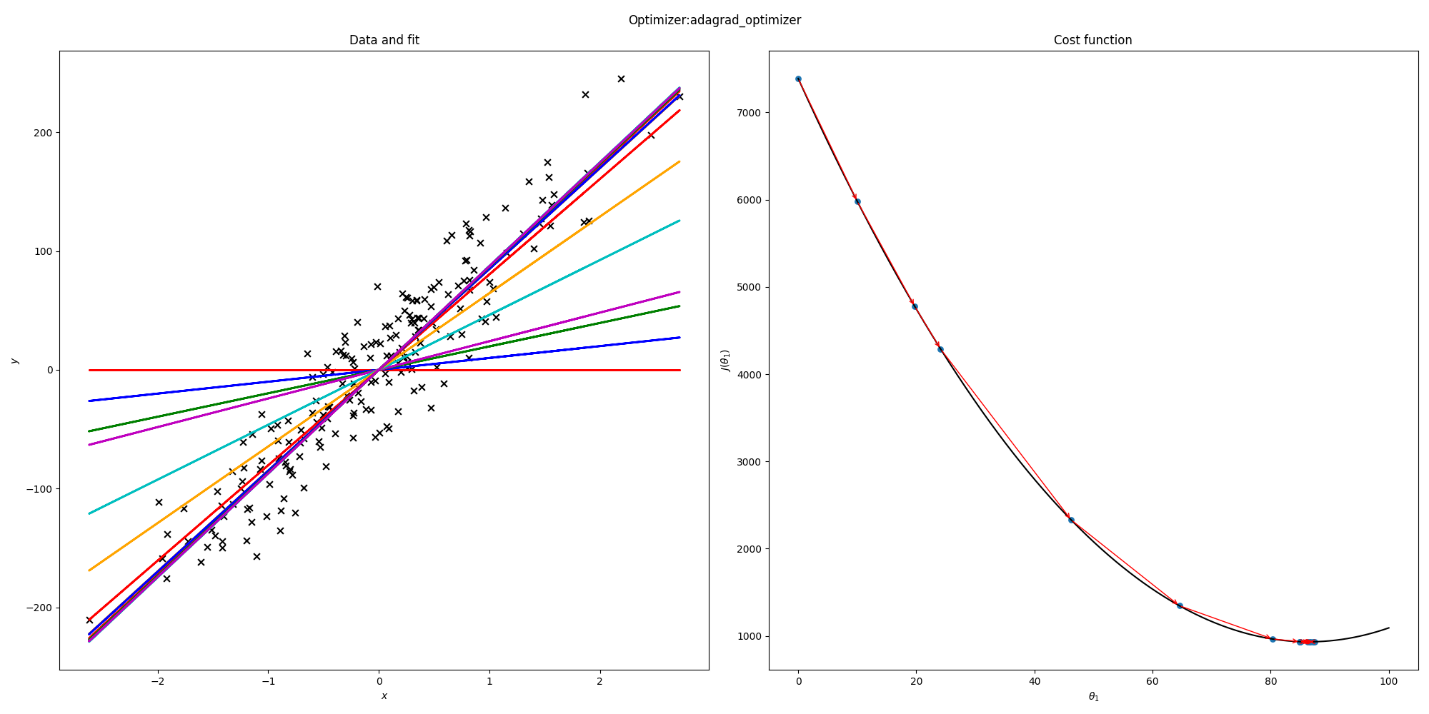


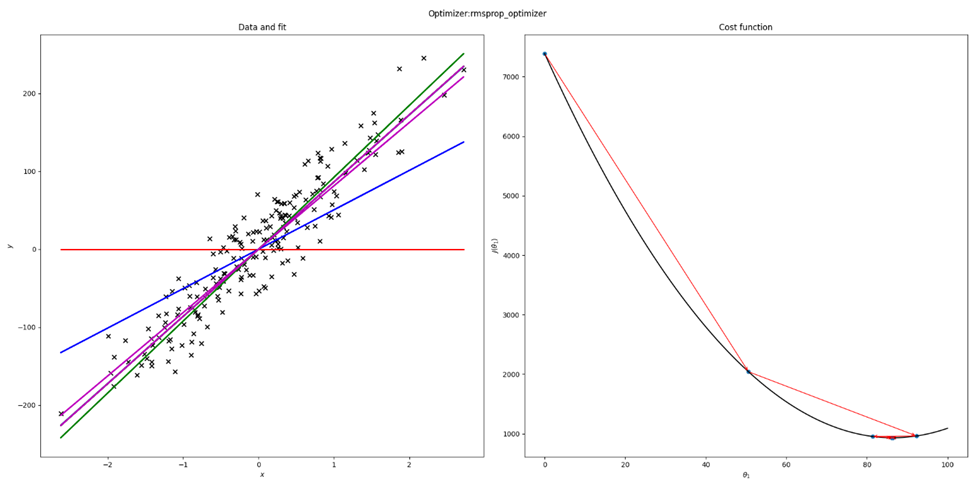
SGD is the same as GD with only one data point used for every iteration to pervent large data computation.

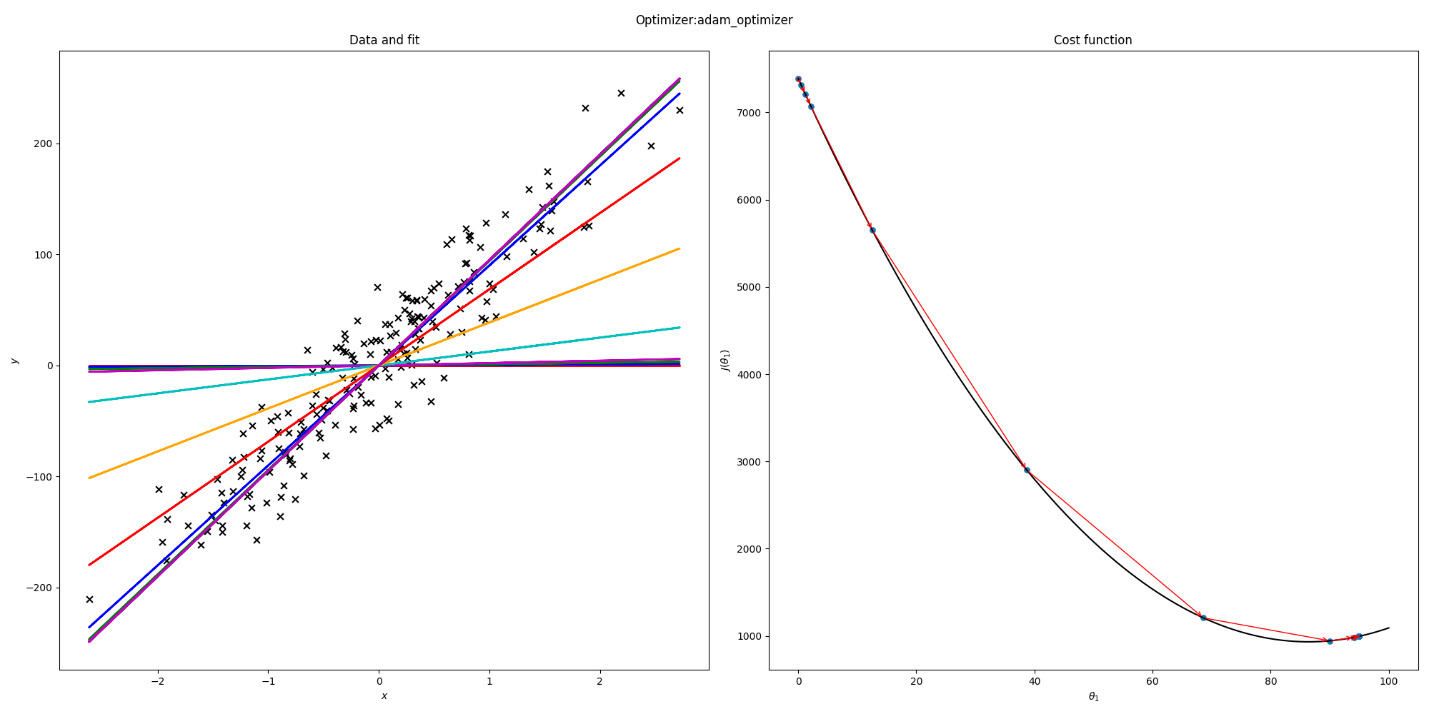


SGD momentum is like SGD (though I used batch size of 3) which pervents back and forth by considering the previous vlues for the gradiant. 1 Hyperparameter is added to decide how much of the previous momentums to keep.



ADAgrad uses the sum of the square of the all previous gradiant to change the learning rate and help with leaarning rate schaduling.

Similar to ADAgrad but we use an exponentily decaying sum of square of all previous values. This helps to give more importance to new graadiants, speeding up the overall process. 1 Hyperparameter is added to decide how much of the previous second momentums to keep. 

Combines both RMSprop and SGD momentum meaning it uses a decaying sum of first and second momentums to speed up the process. But in action I found this optimizer to be extremly unstable and hard to tune. 2 Hyperparameter are added to decide how much of the previous first and second momentums to keep. 

For Adam optimizer, no matter what I did, for a beta1 (first momentum) value of higher than 0.47, there was no real learning. but it did converge on to (a wrong) value.