Wet assignment

Alexander Shender 328626114

Eliran Cohen 204187801

# Part 0.

## Question 1.

For a single measurement we have:

Where:

Inserting into :

In a vector multiplication form:

And for the different measurements we can write in a vector form:

Where contains the measurements (which include the noise).

By aiming to minimize the following problem:

We try to find the coefficients which can ‘describe’ the measurements in a best way.

## Question 2.

Convexity:

* The problem is defined over the whole range , without constraints. This range is a convex set
* The objective function is a squared Euclidean distance, which is a smooth convex function.

To find the minimum analytically, we need to make derivative w.r.t **.**

is of rank , thus is a full rank matrix. So the inverse exists, and we can write:

Which is the analytical solution.

# Part 1.

## Question 3+4.

The function with the desired parameters was created, and the corresponding noise was added. The function was written which has estimated the vector, according to the analytical solution derived in Question 2. The graph is the following:

Chart, scatter chart

Description automatically generated

We can indeed see that the Estimated function is not completely equal to the True original function. But this function minimizes the objective function that was defined.

## Question 5+6.

The function was written that performs the Projected Gradient Descent algorithm with 2 possible step size calculations as described in the HW assignment. The graphs obtained are the following:

Chart, line chart

Description automatically generated

The additional operation of taking absolute value was added to the Error function.

We can indeed see that the AdaGrad step size converges much faster and gives a much smaller error over smaller number of iterations. The reason is that it takes into account all the previous gradients, and adapts the step size accordingly.



## Question 7.

We shall find the smoothness parameter L of the objective function.

From the definition of smoothness:

Reminder:

Inserting:

**To be continued…**

**\*\*\*\*\*\*\***

**The matrix is symmetric, . The** eigenvalues of  *are squared* eigenvalues of X . so all the eigenvalues are positive , which makes X^2 positive definite.

For each PD matrix A and each :

So for each b,c :

So :

\*\*\*\*\*\*\*

## Question 8.

The following graph was obtained:

Chart, line chart

Description automatically generated

We can see that for the constant step size of 1/L we have the best convergence, indeed as was proved in the lectures and tutorials.

For step size of 10/L the algorithm “jumps over” the minimum point. The ‘update’ step is moving in the direction of the negative gradient, but the step is too big.

For 1/10L the algorithm would also converge, but it will take longer, as we may observe.



## Question 9.

The graph obtained is the following:

Chart, line chart

Description automatically generated

**FILL EXPLANATION**

**We know that the convergence rate of AdaGrad is , and the convergence rate of the PGD with const step size is**

**But since the norm of the subgradient is smaller than G , so the convergence rate is much faster than convergence rate of the PGD with const step size.**

****

## Question 10+11

The graphs obtained is the following:



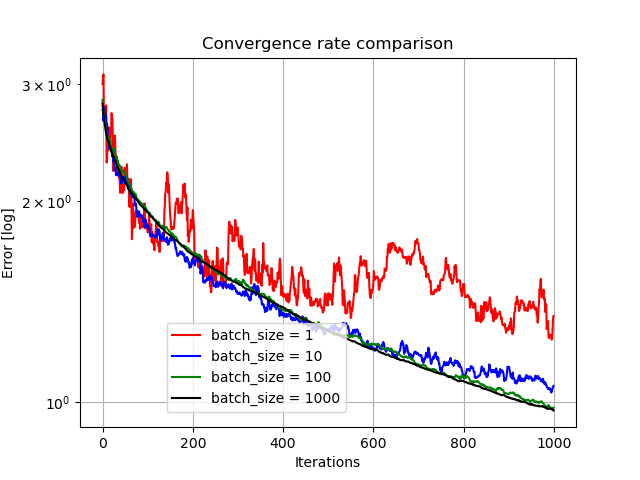
We can see that the bigger the batch the algorithm gives a smaller error, but takes more time.



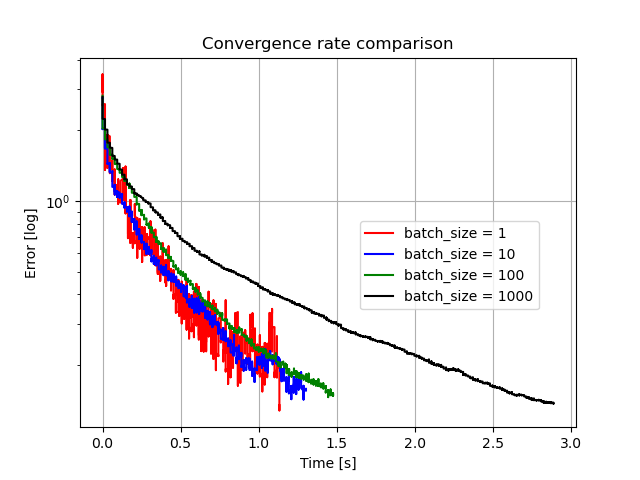
We can also see that the result of of the maximum batch (size m) looks alike the result of the PGD algorithm with decreasing step size

PYTHON GRAPHS:

Question 10:



Question 11.



The following conclusions can be made by looking at those graphs:

1. As the batch size decreases, the ‘error’ function progression becomes more ‘noisy’. This is explainable, since we choose random samples (which have noise element), their gradients differ at the same current solution. As the batch size increases, we average the gradient update with more samples, thus making it directed more precisely towards the minimum.
2. As batch size decreases, we reach less optimal solution over same number of iterations. Easily explainable – the gradient for the solution update is more noisy, since we take less samples to calculate the gradient. So, cumulatively it reaches less optimal solution
3. As the batch size increases, the current gradient calculation is more precise, but it takes longer to calculate it. Thus, for a given available time, it may be beneficial to use smaller batch size, that will make more (but less precise) updates, which will bring the error lower. For example, we can see that batch size of 1/10/100 performed better after 1 second of algorithm run than algorithm with batch size of 1000.
4. For a small batch size, we may never reach an optimal solution. We can see in the first graph that for a batch size of 1, the rate of error decrease slows down by a lot as number of iterations progresses. It is impossible to calculate precisely the optimal value, if we take only 1 sample at the time, and don’t account for all the other samples at this given step. Thus, we can reach some sub-optimal solution only.