Wet assignment

Alexander Shender 328626114
Eliran Cohen 204187801

Contents

Part 0.

Question 1.

For a single measurement we have:

$$y_i = f(x_i) + \omega_i \ ; \ \ i = 1, \dots, m$$

Where:

$$f(x_i) = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n$$

Inserting $f(x_i)$ into y_i:

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n + \omega_i \ ; \ \ i = 1, \dots, m$$

In a vector multiplication form:

$$y_i = [1 \quad x_i \quad \dots \quad x_i^n][a_0 \quad a_1 \quad \dots \quad a_n]^T + \omega_i$$

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

$$\boldsymbol{y}_{\boldsymbol{m}\times\boldsymbol{1}} = \underbrace{\begin{bmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \dots & \dots & \dots & \dots \\ 1 & x_m & \dots & x_m^n \end{bmatrix}}_{\boldsymbol{X}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix}}_{\boldsymbol{a}}$$

$$\boldsymbol{X_{ij}} = (\boldsymbol{x_i})^{\boldsymbol{j-1}} \ \boldsymbol{for} \ , \boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{m}, \boldsymbol{1} \leq \boldsymbol{j} \leq \boldsymbol{n} + \boldsymbol{1}$$

Where $\boldsymbol{y}_{\boldsymbol{m}\times\boldsymbol{1}}$ contains the measurements (which include the noise).

By aiming to minimize the following problem:

$$(P) \min_{a} \left\{ h(\boldsymbol{a}) = \frac{1}{2m} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}\|_2^2 \right\}$$

We try to find the coefficients $\boldsymbol{a}$ which can 'describe' the measurements in a best way.

Question 2.

Convexity:

- The problem is defined over the whole range R^n , 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 $\boldsymbol{a}$.

$$\frac{\partial h(a)}{\partial a} = \frac{\partial}{\partial a}\left(\frac{1}{2m}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}\|_2^2\right) = \frac{1}{2m}\left(-2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})\right) = \boldsymbol{0}$$

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}) = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{a} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{y} = \underbrace{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}}_{\boldsymbol{n}\times\boldsymbol{n}}\boldsymbol{a}$$

$\boldsymbol{X}$ is of rank $n + 1$, thus $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ is a full rank matrix. So the inverse exists, and we can write:

$$\boldsymbol{a} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\boldsymbol{1}}\boldsymbol{X}^T\boldsymbol{y}$$

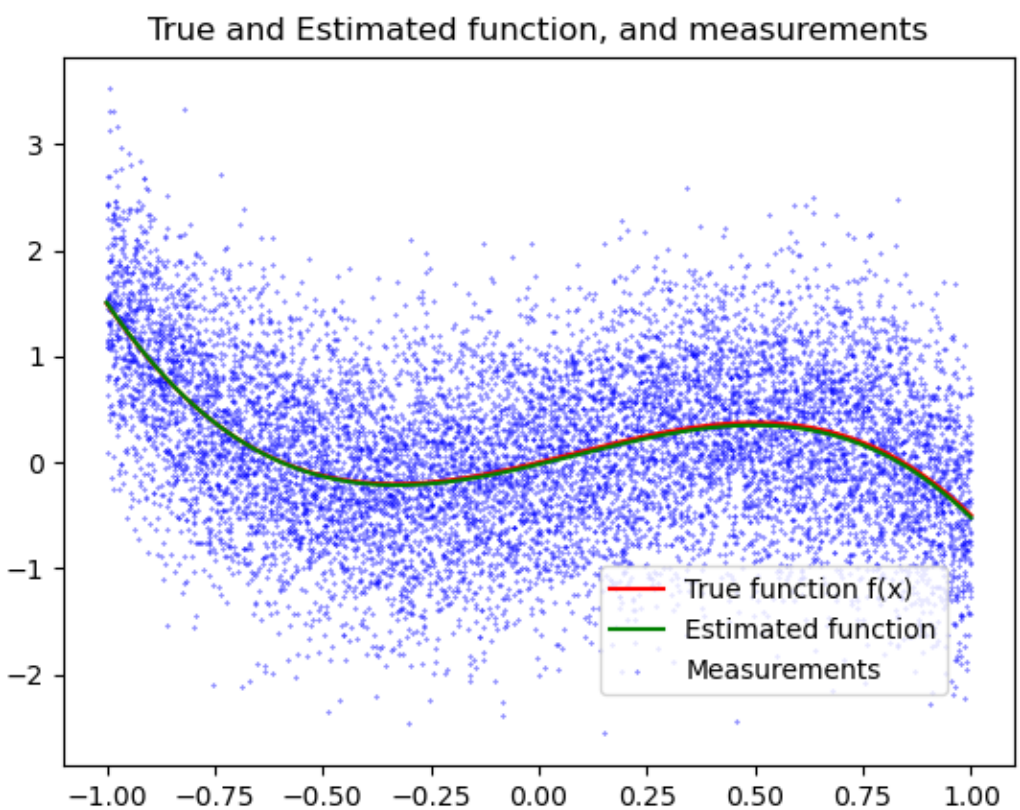
Which is the analytical solution.

Part 1.

Please Note that some graphs appear twice, since the assignment was first done by each of us independently. And code for both is supplied.

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 $\boldsymbol{a}$ vector, according to the analytical solution derived in Question 2. The graph is the following:



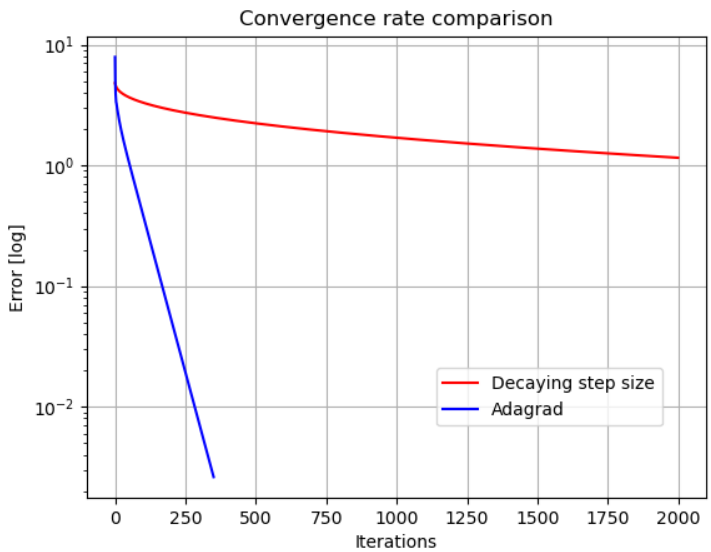
We can indeed see that the Estimated function is not completely equal to the True original function. But this function minimizes the objective function that was defined.

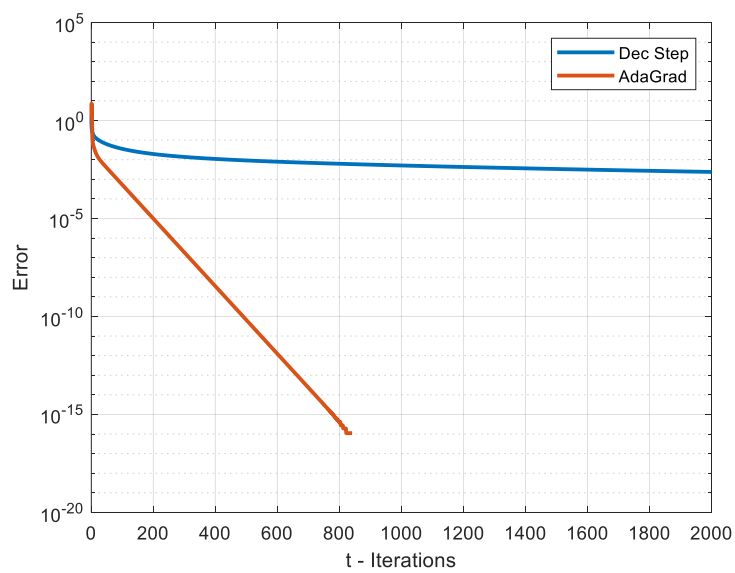
Part 2.

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 (PY & MATLAB):





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.

The matrix $\boldsymbol{X^T X}$ is symmetric, $\boldsymbol{X^T X} = \boldsymbol{X^2}$. The eigenvalues of $\boldsymbol{X^2}$ are squared eigenvalues of X . so all the eigenvalues are positive , which makes X^2 positive definite.

For each PD matrix A and each $x \in R^n$:

$$A \succ 0 \rightarrow \; x^T A x \leq \; \lambda_{max}(A) \|\boldsymbol{x}\|_2^2$$

So for each b,c :

$$\|\nabla h(\boldsymbol{b}) - \nabla h(\boldsymbol{c})\|_2^2 = \left\| -\frac{1}{m}\boldsymbol{X^T}(\boldsymbol{y} - \boldsymbol{Xb}) + \frac{1}{m}\boldsymbol{X^T}(\boldsymbol{y} - \boldsymbol{Xc}) \right\|_2^2 = \left\| \frac{1}{m}\boldsymbol{X^T X}(\boldsymbol{b} - \boldsymbol{c}) \right\|_2^2$$

$$= \left\| \frac{1}{m}\boldsymbol{X^T X}(\boldsymbol{b} - \boldsymbol{c}) \right\|_2^2 = \left(\frac{1}{m}\boldsymbol{X^T X}(\boldsymbol{b} - \boldsymbol{c}) \right)^{\boldsymbol{T}} \left(\frac{1}{m}\boldsymbol{X^T X}(\boldsymbol{b} - \boldsymbol{c}) \right)$$

$$= \frac{\boldsymbol{1}}{\boldsymbol{m^2}}(\boldsymbol{b} - \boldsymbol{c})^{\boldsymbol{T}}\boldsymbol{X^T X X^T X}(\boldsymbol{b} - \boldsymbol{c}) = (\boldsymbol{b} - \boldsymbol{c})^{\boldsymbol{T}} \left(\frac{\boldsymbol{X^T X}}{\boldsymbol{m}} \right)^{\boldsymbol{2}} (\boldsymbol{b} - \boldsymbol{c})$$

$$\leq \boldsymbol{\lambda_{max}^2} \left(\frac{\boldsymbol{X^T X}}{\boldsymbol{m}} \right) \|(\boldsymbol{b} - \boldsymbol{c})\|_2^2$$

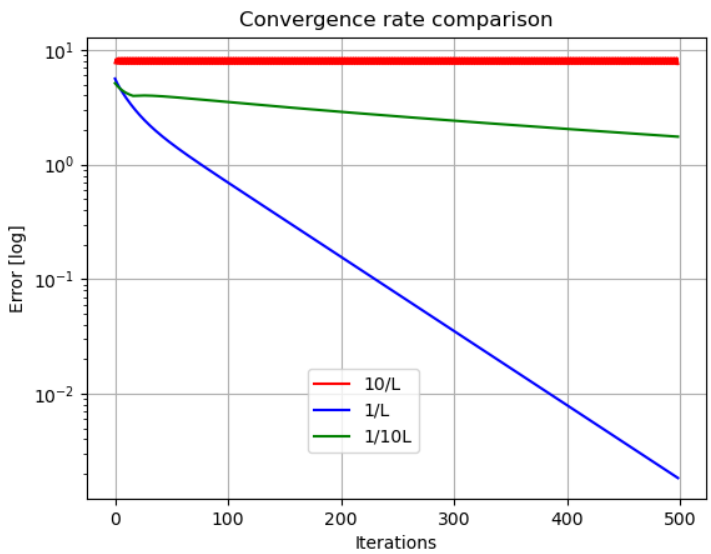
So :

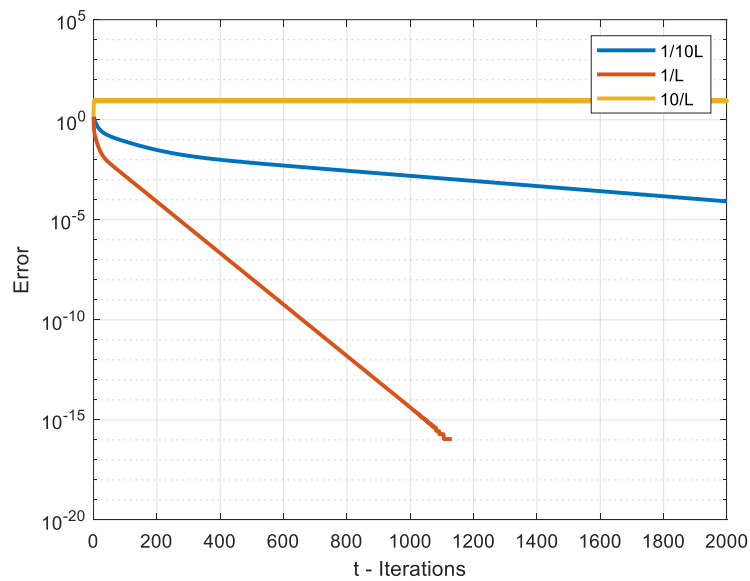
$$\|\nabla h(\boldsymbol{b}) - \nabla h(\boldsymbol{c})\|_2^2 \leq \boldsymbol{\lambda_{max}^2} \left(\frac{\boldsymbol{X^T X}}{\boldsymbol{m}} \right) \|(\boldsymbol{b} - \boldsymbol{c})\|_2^2$$

$$\|\nabla h(\boldsymbol{b}) - \nabla h(\boldsymbol{c})\|_{\boldsymbol{2}} \leq \lambda_{max} \left(\frac{\boldsymbol{X^T X}}{\boldsymbol{m}} \right) \|\boldsymbol{b} - \boldsymbol{c}\|_{\boldsymbol{2}} \rightarrow \boldsymbol{L} = \lambda_{max} \left(\frac{\boldsymbol{X^T X}}{\boldsymbol{m}} \right)$$

Question 8.

The following graphs were obtained:





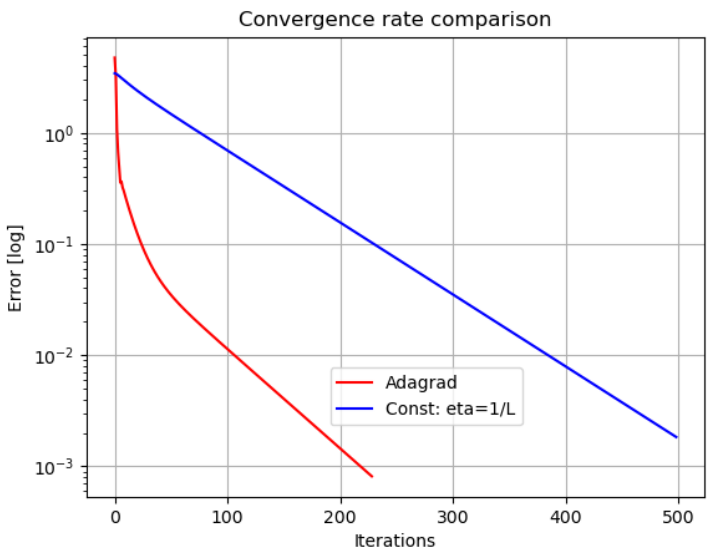
We can see that for the constant step size of 1/L we have the best convergence, indeed as was proved in the lectures and tutorials.

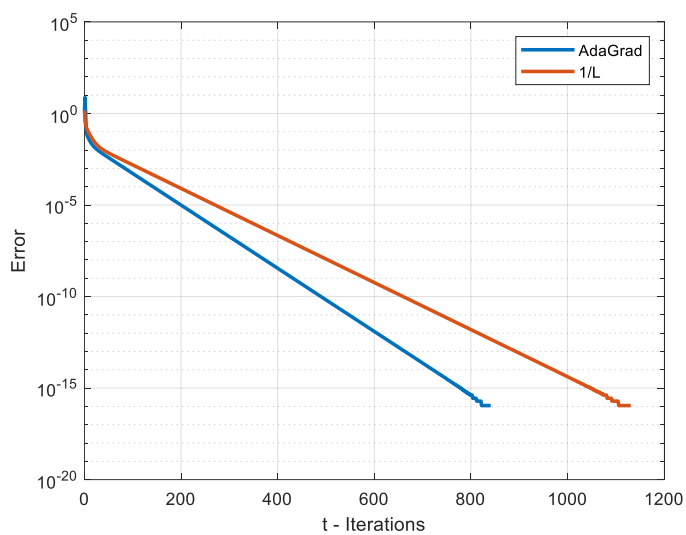
For step size of 10/L the algorithm "jumps over" the minimum point. The 'update' step is moving in the direction of the negative gradient, but the step is too big.

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

Question 9.

The graphs obtained is the following:





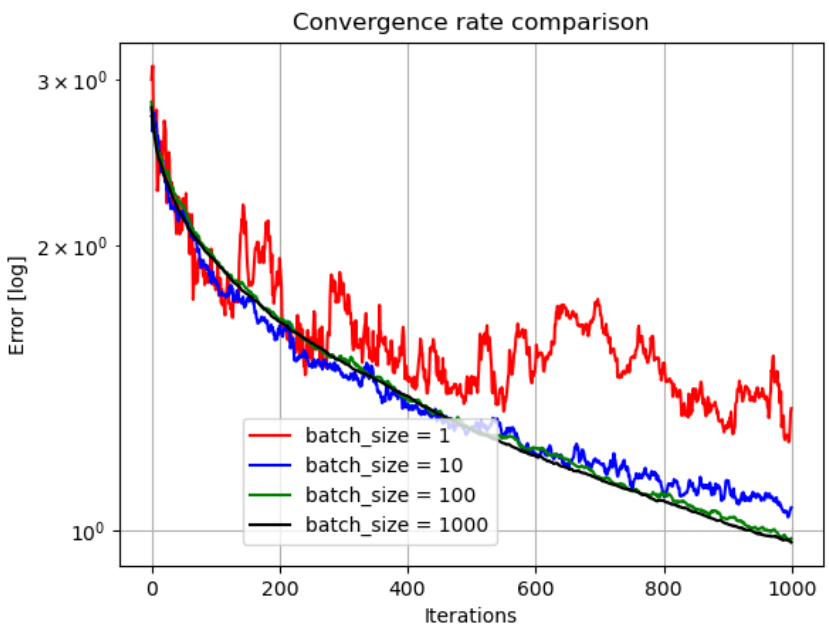
We know that the convergence rate of AdaGrad is $\boldsymbol{O}\left(\frac{1}{\sqrt{t}}\right)$, and the convergence rate of the PGD with const step size $\frac{1}{L}$ is $\frac{LD^2}{t}$

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.

Part 3.

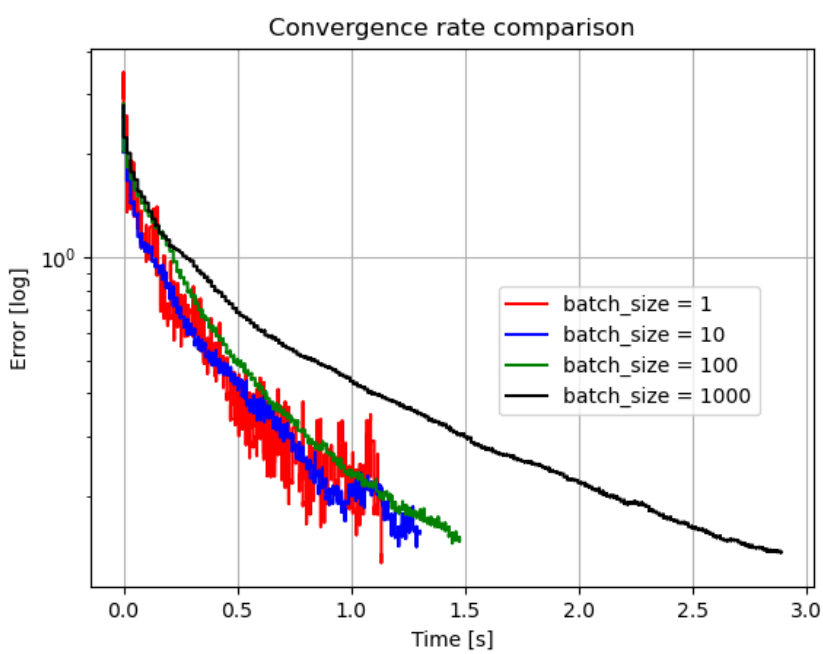
Question 10:

The graph obtained is the following (only PY):



Question 11.

The graph obtained is the following (only PY):



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.