Princess Tara Zamani
Adv Opt for Machine Learning
Homework 7
Accelerated Gradient Descent

<u>Simulation 1</u>: Accelerated gradient descent for least-mean-square problem.

The objective of this assignment was to apply accelerated gradient descent to solve the least-mean-square problem defined as:

$$min_{x\in R^n}\ f(x)\ = \frac{1}{2}\|Ax-b\|_2^{\ 2},\ \text{where } x \in R^n,\ A \in R^{m\times n},\ b \in R^m.$$

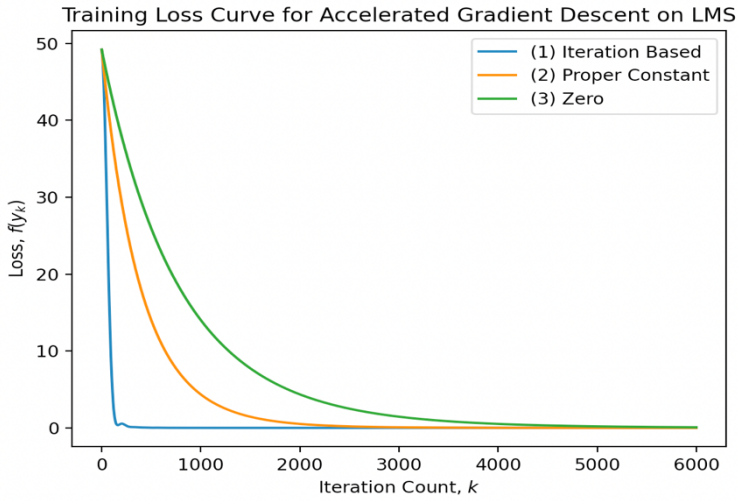
The gradient for the definition above is given as $\nabla f(x) = A^T(Ax-b)$.
The accelerated gradient algorithm implementation is iterative and takes the following form:

$$x_{k+1} = y_k - \eta\nabla f(y_k),\ \text{where } \eta \text{ is the learning rate.}$$
$$y_{k+1} = x_{k+1} + \theta_k\,(x_{k+1} - x_k),\ \text{where } \theta_k \text{ is the momentum coefficient.}$$

The underlying signal (z), and the initialization of x_0 were generated randomly using the numpy.random.randn function. The initial value of y_0 was set equal to x_0. The initial noise variance was chosen to be 0.1.

The value of θ_k was varied to explore how loss convergence is affected by the momentum coefficient. Three values of θ_k were explored: (1) $\theta_k = \frac{k}{k+3}$ (2) θ_k is a proper constant of 0.5, and (3) $\theta_k = 0$, which means a normal gradient descent method would be executed.

The results of the gradient descent implementation are shown below. The following training loss curve shows the functionality of the accelerated gradient descent algorithm implementation with the different θ_k values. The results were obtained with the following settings: n = 100, m = 10, learning rate = 1e-3, noise variance = 0.1, iterations k = 6000.



In general, the loss curves behave as expected. The loss decreased as training continues. Comparing the different methods of assigning a value to θ_k, we can clearly see that the Iteration Based approach of $\theta_k = \frac{k}{k+3}$ converged the fastest for the LMS model.

<u>Simulation 2</u>: Accelerated proximal gradient descent for compressive sensing problem.

The objective of this assignment was to apply accelerated gradient descent to solve the compressive sensing problem defined as:

$$min_{x\in R^n}\, F(x) \;= \frac{1}{2}\|Ax-b\|_2{}^2 \;+\; \lambda||x||_1 \;,$$ where $x \in R^n$, $A \in R^{m\times n}$, $b \in R^m$.

The proximal operator of $\lambda||x||_1$ is given as:

$$prox_{\lambda\|\cdot\|_1}(x) \;=\; (|x|-\lambda)_+ * sign(x) \;\;,\;\; where(u)_+ := \; max\{u,0\}$$

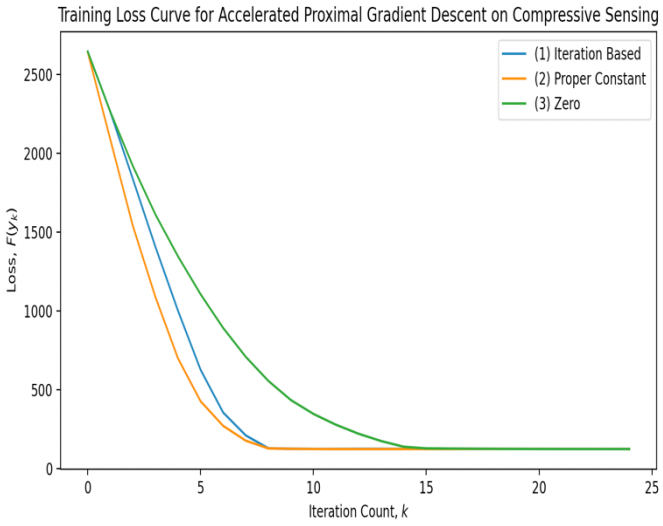
The underlying signal (z), and the initialization of x_0 were generated randomly using the numpy.random.randn function. The underlying signal was made sparse by randomly replacing 95% of its values with zeros. The initial noise variance was chosen to be 0.1. The variables A and b were generated in the same way as in HW2.

The accelerated proximal gradient algorithm implementation is iterative and takes the following update rule:

$$x_{k+1} = prox_{\lambda h}(\, y_k - \eta\nabla f(y_k)\,)$$
$$y_{k+1} = x_{k+1} + \theta_k\,(x_{k+1} - \; x_k)$$

where θ_k is the momentum coefficient. The λ was fined tuned in Homework 6 and was assigned to the best value of 0.1.

The results were obtained using the following parameters: n = 1000, m = 100, learning rate = 1e-3, λ = 0.1, variance = 0.1, iteration number k = 25. The same three methods for assigning the momentum coefficient were explored as in Simulation 1. The loss curve is shown below.



In general, the loss curves behave as expected. The loss decreased as training continues. Comparing the different methods of assigning a value to θ_k, we can see that the Proper Constant approach of $\theta_k \;= 0.5$ converged the fastest for the compressive sensing model. We can also observe that compared to the LMS in the first simulation, this approach takes significantly less iterations to reach a convergence, which is expected when using the proximal gradient method compared to regular gradient descent.