Scalable Data Mining
Programming Assignment
Large Scale Machine Learning

1. You have been provided with a regression dataset "LinearRegdata.txt' on which you will implement some of the optimisation methods studied in class, from scratch.

   Below is the explanation for some notations and derivations which will help you through the implementation.

   Let the data $\mathcal{D}$ be in the form (x, y) where $x =$ input, $y =$ target

   $\hat{y} = f(x) = ax + b$ where a, b are the parameters

   $\mathcal{L}(\mathcal{D}, \theta) = (y - \hat{y})^2$ where $\theta \in a, b$

   $\frac{\partial \mathcal{L}}{\partial a} = (y - \hat{y})\frac{\partial \hat{y}}{\partial a} * (-1) = (y - \hat{y})(-x)$

   Similarly, $\frac{\partial \mathcal{L}}{\partial b} = (y - \hat{y})(-1)$

   Following are the equations for different gradient descent algorithms and their corresponding parameter values:

   (a) **Batch Gradient Descent**
   $\theta = \theta - \eta \nabla_\theta \mathcal{L}(\mathcal{D}, \theta)$

   (b) **Mini Batch Gradient Descent** (Maintain a batch size of 10 , $\eta = 0.02$)
   $\theta = \theta - \eta \nabla_\theta \mathcal{L}((x^{i:i+n}, y^{i:i+n}), \theta)$

   (c) **Stochastic Gradient Descent** ($\eta = 0.02$)
   $\theta = \theta - \eta \nabla_\theta \mathcal{L}((x^i, y^i), \theta)$

   (d) **Momentum Gradient Descent** ($v_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \gamma = 0.9, \eta = 0.02$)
   $v_t = \gamma v_{t-1} + \nabla \mathcal{L}(\mathcal{D}, \theta)$
   $\theta = \theta - v_t$

   (e) **Adam Optimization** ($m_0 = 0, v_0 = 0, \beta_1 = 0.9, \beta_2 = 0.999, \eta = 0.2$)
   $g_{t,i} = \nabla_\theta \mathcal{L}(\mathcal{D}, \theta_{t,i})$
   $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
   $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
   $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
   $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
   $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$

A sample programming template has been given for **Batch Gradient Descent** in the link `https://github.com/cs60021SDM/LargeScaleML.git` under Q1_template.py. Implement the below-mentioned functions from scratch following the equations above:

(a) minibatch_gd()

(b) stochastic_gd()

(c) momentum_gd()

(d) adam_gd()

Report the following in a document:

(a) Plot training loss with each update for each of the optimisation methods. (5 figures in total)

(b) Plot training loss vs epochs for all optimisation methods in one figure. (5 curves in total)

(c) Bar-plot of test Root Mean Square Error (RMSE) vs optimizer using the best model for each of the gradient optimisation methods.

Note: Best model corresponds to the model having the lowest validation error. (In this case, consider the test set for tracing validation error).

2. The task is to perform image classification using MNIST handwritten digits. The code template is available in this link: `https://github.com/cs60021SDM/LargeScaleML.git` under Q2_template.py

You should be able to complete the functions train(), test(), main() using four different optimizers - Mini Batch Gradient Descent, Momentum Gradient Descent ($\gamma = 0.9$), Nesterov Gradient Descent, Adam Optimizer. You can use the available optimizers in PyTorch library. You should also use a Step Decay Learning Rate scheduler while training.

Report the following in a document for the learning rates $\rho = \{0.002, 0.02, 0.2\}$:

(a) Plot training loss with each update for each of the optimisation methods. (4 figures in total)

(b) Plot training loss vs epochs for all optimisation methods in one figure. (4 curves in total)

(c) Bar-plot of test accuracy vs optimizer using the best model from each optimizer.

Submission Instructions:

1. Submit your code using the filename RollNo_AssignmentNo_QuesNo.py

2. Submit the write-up with all plots using the filename RollNo_AssignmentNo_QuesNo.pdf. Explain the plots and the observations briefly. Mention the assumptions, if any.