Scalable Data Mining (Autumn 2023)

Assignment 1: Optimizers (Full Marks: 100)

Question 1: (100 marks)

Task: The aim of this assignment is to train and test your own custom-made model for regression task on Boston Housing Dataset using PyTorch nn.Module.

Data:

The Boston Housing Dataset is derived from information collected by the U.S. Census Service concerning housing in the area of Boston. The following describes the dataset columns:

- CRIM - per capita crime rate by town
- ZN - the proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - the average number of rooms per dwelling
- AGE - the proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

There are 506 data points in the dataset. MEDV is the target variable.

Code Snippet to Load the Dataset:

```
import pandas as pd
import numpy as np

data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
boston_data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2,
:2]])
boston_target = raw_df.values[1::2, 2]
```

Implementation Details:

1. **Load Data:** Use the above code snippet to load the data. Divide it into training and test splits of your preference and perform the experiments.

2. **Model**: Create a 2 Layer Neural Network for performing Regression. You can use Conv Layers, Linear Layers, etc in the network. You can use Relu or Sigmoid as your activations.

3. **Training Module**: Use the following configurations while training:

    Part A: With unnormalized data
    1. Use **SGD optimizer** with **learning rates = 0.1, 0.01, 0.001,** and **MSE-Loss** as the loss function.
    2. Use **SGD optimizer** with **Nesterov parameter True** and an **initial momentum value** that you feel is suitable with **learning rates = 0.1, 0.01, 0.001,** and **MSE-Loss** as the loss function.
    3. Use Adadelta optimizer with **learning rates = 0.1, 0.01, 0.001,** and **MSE-Loss** as the loss function.

    Part B: With normalized data.
    1. Perform all the above experiments with features that are mean-variance normalized.

Do you observe any differences between the different training configurations? State your observations in the report.

You can use early stopping too if loss converges beforehand.

For each configuration above, save the best model (yielding the best test set accuracy).
Use the best-saved models to report the final test set accuracies for all the configurations.

Submission Details:

You should submit the following in zipped format (Rollnumber_AssignmentNo.zip):

- **Report** with all the contents as mentioned under the '**Submission Details**'. Analyze the observations and explain them. **(60 marks)**
- **Python codes:** Code in .py and .ipynb are acceptable. **(40 marks)**

Your report should contain the following details:

1. State the differences and observations regarding each of the training settings.
2. For all the configurations (under Training Module), plot

    i. Training loss on the y-axis with epochs on the x-axis
    ii. Test loss on the y-axis with epochs on the x-axis
    iii. Training accuracy on the y-axis with epochs on the x-axis
    iv. Test accuracy on the y-axis with epochs on the x-axis