CS7.601 (Monsoon 2023)
Deep Learning: Theory and Practices
Deadline: 11.55 PM, September 7th, 2023

Instructions

1. Please submit your code in 2 **Jupyter Notebooks**. We will only consider submissions made using jupyter notebook with **detailed report** of the results included within the notebook using **markdown cells**. The first notebook should contain both questions 1 and 2 and the second notebook should contain question 3.
2. Use Python (Numpy, Pandas and Matplotlib) for your implementation.
3. **Use of Pytorch, Tensorflow or any other deep learning frameworks is not allowed.**
4. Plagiarism will not be tolerated.
5. Write well documented code. You should be able to explain or modify the codes during your evaluations.
6. Your figures should be clearly labeled and easy to understand.
7. **Use 42 as the random seed for both the notebooks** (math.random.seed(42) and np.random.seed(42)) to ensure your results are reproducible during the evaluations.
8. Submission will be online on moodle. Submit a single zip file with the name "rollnumber_A1" which contains both the jupyter notebooks.

1 Gradient Descent and Variants - 1 [4 Points]

Consider the Rosenbrock function $f(x, y) = x^2 + 100(y - x^2)^2$, which is used to benchmark optimization algorithms and the following variant admits a global minimum at $(0, 0)^T$. Use random initialization of the parameters.

For reference : Stochastic Gradient Descent

- Run gradient descent with constant step size to minimize $f(x, y)$. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Use gradient descent with Polyak's momentum method to minimize $f(x, y)$. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Minimize $f(x, y)$ using Nesterov accelerated gradient descent. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Minimize $f(x, y)$ using Adam optimizer. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]

2 Gradient Descent and Variants - 2 [4 Points]

Consider the following function

$$f(x, y) = \frac{50}{9}(x^2 + y^2)^3 - \frac{209}{18}(x^2 + y^2)^2 + \frac{59}{9}(x^2 + y^2).$$

This function has global minimum at $(0, 0)^T$ and local minima at $x^2 + y^2 = 1$. Consider minimizing $f(x, y)$ using the methods below. Use random initialization of the parameters.

- Use gradient descent with constant step size to minimize $f(x, y)$. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Use Polyak's momentum method to minimize $f(x, y)$. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Minimize $f(x, y)$ using Nesterov accelerated gradient descent. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]
- Minimize $f(x, y)$ using Adam optimizer. Show contour plot of the function. After every update, using arrow show the movement in the contour plots. Do it till convergence. [1 Point]

3 Back Propagation, Resilient Propagation (RProp) and Quickprop [12 Points]

Download **Concrete Compressive Strength** dataset from the link below.
https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

The concrete compressive strength prediction is a regression problem. There are a total of 1030 examples with 8 input variables and 1 target variable. Randomly split the data in training set (70% of total points) and testing set (30% of total points). You will have to train a 3-layer neural network (i.e, input layer, hidden layer, and output layer) which will predict the value of concrete compressive strength given the 8 input variables. Consider the following 4 network variations

1. 25 hidden units, tanh activation function.
2. 50 hidden units, tanh activation function.
3. 25 hidden units, Leaky ReLU activation function.
4. 50 hidden units, Leaky ReLU activation function.

Train each of these variations with batch BackProp, QuickProp, and RProp. Use Mean Squared Error (MSE) as the loss function. Train the network for 1e3 epochs. Use 1e-3 as the LR. Report the following for every setting of (hidden units, activation function, optimization algorithm):

- Lowest MSE on train data and test data.
- Graph of MSE over train data vs epochs.
- Graph of MSE over test data vs epochs.

Additionally, compare and analyze the results (loss, epochs to convergence, approximate time for all epochs) obtained on the following basis:

1. Choice of optimization algorithm.
2. Choice of activation function.
3. Choice of number of hidden units.

You may want to explore data preprocessing techniques, additionally, if any prepossessing is used, state why you are using the particular data-processing technique.

Leaky ReLU activation function

$$LeReLU(x) = \begin{cases} x, & x \geq 0 \\ 0.01x, & x < 0 \end{cases}$$