CS7.601 (Spring 2020)
Deep Learning: Theory and Practices
Submission Deadline: 6.00 PM, March 14th, 2020

Instructions

1. Please submit your code in **Jupyter Notebook**. We will only consider submissions made using jupyter notebook with **detailed report** of the results included within the notebook using **markdown cells**.
2. You may use Python (Numpy, Pandas and Matplotlib) for your implementation.
3. **Use of Pytorch, Tensorflow or any other deep learning frameworks is not allowed.** If you are using any other languages, please contact the TAs before you proceed.
4. Write your own codes..
5. You have to turn in the well documented code along with a detailed report of the results (zip of .ipynb files) electronically in Moodle. The report should be precise and concise.

1 Gradient Descent Method [6 Points]

Consider learning a linear regression function (without bias term) as follows.

$$y = 2x_1 + 3x_2 + \epsilon$$

Here, $\epsilon \sim \mathcal{N}(0, 0.01)$ ($\mathcal{N}$ is Gaussian distribution). Let 400 samples are generated uniformly randomly from $[-4, 4]^2$. And then corresponding y values are generated using the equation above.

1. Consider learning a linear regression function (without bias term). Thus, there are 2 parameters only. Assuming squared error loss function, write the objective function J to be minimized. [0.5 Point]
2. Plot the error curve with respect to the parameters for this problem. [0.5 Point]

3. Consider minimizing J using gradient descent with constant step size. Calculate the optimum learning rate η_{opt}. How do you get this. [1 Point]

4. Try gradient descent with following step sizes $\eta = \frac{0.9\eta_{opt}}{2}, \frac{1.5\eta_{opt}}{2}, \eta_{opt}, 1.5\eta_{opt}$. For definiteness, we consider convergence to be complete $J < 0.001$. For every case,

   (a) Plot the error vs epoch curve. [2 Points (0.5 for each η)]

   (b) Using contour plots show the convergence path (similar to what we saw in the class). [2 Points (0.5 for each η)]

2 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.

- 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]

3 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]

4 Preprocessing the data speeds up learning [5 Points]

Demonstrate that pre-processing data can lead to significant reduction in time of learning. Consider a single linear output unit for a two-category classification task, with teaching values ± 1 and squared error criterion.

1. Write a program to train three weights based on training samples.
2. Generate training set having 400 samples, 200 from each of the two categories $P(w_1) = P(w_2) = .5$ and $p(\mathbf{x}|w_i) \sim \mathcal{N}(\mu_i, \Sigma)$, where $\mathcal{N}$ is Gaussian distribution, $\mu_1 = (-3, 4)^T$ and $\mu_2 = (4, -3)^T$ and

$$\Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix}$$

   Generate 200 test samples, 100 from each category.
3. Find the optimal learning rate empirically by trying a few values. Plot error versus epoch curve for both training and test sets. [2 Points]
4. Train to minimum error. Why is there no danger of overtraining in this case? [0.5 Point]
5. Why can we be sure that it is at least possible that this network can achieve the minimum (Bayes) error. [0.5 Point]
6. Now pre-process the data by subtracting off the mean and scaling with standard deviation in each dimension. Repeat the above, and find the optimal learning rate. [2 Points]

5 Back Propagation, Resilient Propagation (RProp) and Quickprop [6 Points]

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

The concrete compressive strength is the regression problem. There are a total of 1030 examples with 8 variables. Randomly split the data in training set (70% of total points) and testing set (30% of total points)

Train a 3-layer neural network (with three variations of number of hidden nodes 25, 50, 75) that can predict the value of Concrete compressive strength, given 8 inputs. Use following activation functions: (a)tanh, (b) Relu. Use squared error loss function at the output. Train using RProp, Quickprop and batch Back-propagation (BP) for 1000 epochs. Report the following.

1. For every setting (number of hidden nodes, activation function), plot MSE versus number of epochs curves for (a) Rprop, (b) Quickprop and (c) Back-propagation. Comment on your observation.

2. Find the final training and test MSE values for each setting. Report all of them using a table at the beginning of the notebook itself.