IST 597: Foundations of Deep Learning

Assignment 10

Instructor : Dr. C. Lee. Giles
TA : Neisarg Dave
Due Date : Tuesday, September 12, 2023

Course Policy

- Carefully read all the instructions before you start working on the assignment
- Give maximum explanation for each sub-problem. Please avoid single-line answers; submissions without any explanations will receive 0 points.
- Assignments are due before class at 02:29 pm. Please check the due date on Canvas.
- Late exercises will receive 50% credits for the first 24 hours and no credits thereafter.
- All exercise solutions must be turned in, even if late. Failure to do so can result in a deferred grade.
- All source materials must be cited. The University Academic Code of Conduct will be strictly enforced.
- All queries related to Assignment should have a subject line *IST597 : Assignment_01 Queries*

Assignment Instructions:

- The submission for this assignment must be a zipped folder: {*name*}*_assignment_10.zip*
- The folder must contain two files:
  1. {*name*}*_assignment_10.pdf*: Answers for all theory questions and explanations for all problems.
  2. {*name*}*_assignment_10.py*: Codes for programming problems

  A template Python file is provided. Feel free to make any suitable changes.

1 Linear Layer

A linear layer in a neural network is a affine transformation of its inputs, defined as :

$$z = Wx + b$$

where x and z are inputs and output respectively and W and b are parameters. W is called weight matrix and b is bias.

2 Multilayer Perceptron

A multilayer perceptron is constructed by placing multiple linear layers, combined with an activation function. Activation function provides non-linearity to the multilayer perceptron. A good example of activation function is Rectified Linear Unit (ReLU) function defined as :

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

An MLP of L layers can be written as:

$$\hat{z}^{(l)} = W^{(l)} z^{(l-1)} + b$$
$$z^{(l)} = Activation(\hat{z}^{(l)})$$
$$z^{(0)} = x, \quad l \in [0, L-1]$$

3 Loss Function : Cross Entropy

Cross Entropy Loss for a multiclass classification problem is defined as:

$$\mathcal{L}(W) = ce\_loss = \frac{1}{\#samples} \sum_{i}^{\#samples} \sum_{j}^{\#classes} -y_{ij} . \log(p_{ij})$$

where,

$$p_{ij} = \frac{e^{h_{ij}}}{\sum_k e^{h_{ik}}} \quad \text{and} \quad h_i = z_i^{(L-1)}$$

4 Gradient Calculation for Backpropagation

For a given affine transformation:

$$z^{(l)} = W^{(l)} x^{(l-1)} + b$$

We can calulate all the partial derivations in the following fashion

$$\frac{\partial z^{(l)}}{\partial x^{(l-1)}} = W^{(l)}$$

$$\frac{\partial z^{(l)}}{\partial W^{(l)}} = x^{(l-1)}$$

$$\frac{\partial z^{(l)}}{\partial b^{(l)}} = 1$$

To write the **backward** function, we need one more step. **backward** function takes **grad_output** as its input. It is the backward flow of gradients coming from the final output $\hat{y}$ and can be mathematically expressed as:

$$grad\_output = \frac{\partial \hat{y}}{\partial z^{(l)}}$$

backward functions gives out three values:

$$grad\_x = \frac{\partial \hat{y}}{\partial x^{(l-1)}} = \frac{\partial \hat{y}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial x^{(l-1)}}$$

$$grad\_w = \frac{\partial \hat{y}}{\partial W^{(l)}} = \frac{\partial \hat{y}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W^{(l)}}$$

$$grad\_b = \frac{\partial \hat{y}}{\partial b^{(l)}} = \frac{\partial \hat{y}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial b^{(l)}}$$

In the given code template update the **forward** and backward functions of class **LinearFunction** to create an affine transformation layer.

5 Parameter Update Equation

Backpropogation Algorithm is used for calculating the gradients of loss function with respect to the the parameters ($\nabla_W \mathcal{L}$). The parameters then can be updated at the end of each iteration:

$$\hat{W} = W - \alpha \nabla_W \mathcal{L}$$

6 Assignment

In this assignment, you are required to do the following:

1. Create an MLP — 3 marks
   (a) Write the **forward** function for affine transformation
   (b) Write the **backward** function (to calculate gradients) for the affine transformation
   (c) Create a Neural Network composed of multiple affine transformations.
2. Train the Neural Network on Fashion MNIST Dataset — 4 marks
   (a) Write code for Loss Function.
   (b) Write code for the parameter update routine
   (c) Write code to calculate loss on validation set. Validation should be run after every epoch.
   (d) Plot Train and Validation Loss Curves using Matplotlib
3. Testing the Neural Network — 2 marks
   (a) Change the **seed** and train the neural network 5 times
   (b) Report the mean and variance over 5 trials on the following metrics on test set:
      i. Accuracy
      ii. Precision
      iii. Recall
      iv. F1
      v. ROC AUC
4. Change the number of parameters in the Neural Network — 3 marks
   (a) Train an underfit model
   (b) Train a overfit model
   (c) Demonstrate the difference between overfitting and underfitting