CSE 673 Assignment 1: Neural Network from scratch

The objective of this assignment is to get started with Neural Networks and understand the Backpropagation algorithm. The primary goal of the assignment is to create a toy deep learning framework called “PyTorchNano ” using only the Numpy library in python.

Requirements:

The framework should support the following:

Layers (30 Points + 15 Bonus Points)

- Dense Layer (5 pts)
    - *Definition*: Implement a fully connected layer.
    - *Input parameters*: The input, number of neurons
    - *Naming convention*: Dense
    - *Usage* : Dense(X, 100) -- > takes X as input size and is a hidden layer with 100 neurons.

- Convolutional Layer (10 pts)
    - *Definition*: Implement the convolutional operation.
    - *Input parameters*: The input, Number of output channels, Kernel size, Padding and Stride
    - *Naming convention*: Conv
    - *Usage* : Conv(X,10,(3x3),1,1) -- > takes X as input size, produces output with 10 channels, uses a 3 x 3 kernel with padding and stride both equal to 1

.

- Average Pooling (5 pts)
    - *Definition*: Implement the average pooling operation.
    - *Input parameters*: The input, Kernel size, Padding, and Stride
    - *Naming convention*: AvgPool
    - *Usage* : AvgPool(X,(2x2),1,1) -- >takes X as input size, uses a 2 x 2 kernel with padding and stride both equal to 1

- Max Pooling (5 pts)
    - *Definition*: Implement the Max pooling operation.
    - *Input parameters*: The input, Kernel size, Padding, and Stride
    - *Naming convention*: MaxPool
    - *Usage* : MaxPool(X,(2x2),1,1) -- >takes X as input size, uses a 2 x 2 kernel with padding and stride both equal to 1

- Flatten Layer (5 pts)
    - *Definition*: Implement a layer that flattens the input vector.
    - *Input parameters*: The Input
    - *Naming convention*: Flatten
    - *Usage*: Flatten() -> takes the input and flattens the last dimension

- Dropout (5 bonus pts):
    - Definition: Implement a layer with drop-out functionality.
    - Input parameters: percentage of input to drop
    - Naming convention: Dropout
    - *Usage*: Dropout(0.2) -> drops 20% of the input

- BatchNorm (10 bonus pts) :
    - Definition: Implement the batch normalization functionality.
    - Input parameters: The Input
    - Naming convention: BatchNorm
    - Usage: BatchNorm(X) ->takes X as input size

Activation functions (7 points)

- Sigmoid (2 pts)
    - *Definition*: Perform the sigmoid transformation on the inputs.
    - *Input parameters*: The input
    - *Naming convention*: Sigmoid
    - *Usage*: Sigmoid()

- Softmax (3 pts)
    - *Definition*: Perform the softmax transformation on the inputs
    - *Input parameters*: The input
    - *Naming convention*: Softmax
    - *Usage*: Softmax()

- ReLU (2 pts)
    - *Definition*: Perform the rectified linear transformation on the inputs
    - *Input parameters*: The input
    - *Naming convention*: ReLU
    - *Usage*: ReLU()

Loss Functions and Metrics(18 points)

- Mean Square Error (2 pts)
    - *Definition*: Implement the mean square error function
    - *Input parameters*: Predictions and Ground Truth
    - *Naming convention*: MSE
    - *Usage*: MSE(P, Y) -> takes P as the Prediction and Y as the Ground Truth

- Cross-Entropy (2 pts)
    - *Definition*: Implement the Cross-Entropy Loss formulation
    - *Input parameters*: Predictions and Ground Truth
    - *Naming convention*: CrossEntropy
    - *Usage*: CrossEntropy(P, Y) -> takes P as the Prediction and Y as the Ground Truth

- Hinge Loss (2 pts)
    - *Definition*: Implement the Hinge Loss formulation (SVM)
    - *Input parameters*: Predictions and Ground Truth
    - *Naming convention*: Hinge
    - *Usage*: Hinge(P, Y) -> takes P as the Prediction and Y as the Ground Truth

- Accuracy (2 pts)
    - *Definition*: Calculate Accuracy
    - *Input parameters*: Predictions and Ground Truth
    - *Naming convention*: Accuracy
    - *Usage*: Accuracy (P, Y) -> takes P as the Prediction and Y as the Ground Truth
    - Output: A single value in %

- Confusion Matrix (2 pts)
    - *Definition*: Calculate the confusion matrix
    - *Input parameters*: Predictions and Ground Truth
    - *Naming convention*: ConfusionMatrix
    - *Usage*: ConfusionMatrix (P, Y) -> takes P as the Prediction and Y as the Ground Truth
    - Output: A 10 X 10 matrix

- ROC (8 pts)
    - Definition: Calculate the ROC and plot the curve
    - Input parameters : Predictions and Ground Truth
    - Naming convention : ROC
    - *Usage*: ROC(P, Y) -> takes P as the Prediction and Y as the Ground Truth
    - Output: A Plot using matplotlib of the ROC curve and Report the AUC score

Model architecture: The model is expected to maintain the order to perform the forward and backward propagation. You are expected to create a Net object using which the layers will be defined. The Model will be defined in the model.py file (example shown in the Model.py file). The Net object will be imported from the GraphNet.py file. The layers will be defined in Layers.py. You can create Utils.py for defining any helper functions. Train.py should contain the code to train your model on the MNIST dataset. The model is expected to train using mini-batch SGD.

Test Cases (45 Points)

- Test cases would be in the form of a custom model file. Your code should be able to take the model file, train the network and return the accuracy, confusion matrix and ROC plot.
- The first test case (given in the model file) is worth 5 points
- 5 other test cases would be released later (in a week), each worth 4 points
- 5 hidden test cases used at evaluation, worth 4 points (just to make sure that the you have not hardcoded anything)

We also would be looking for the following while evaluation the code:

- Modularized and readable python code
- Proper use of forward and backward functions to define the Layers
- Vectorized implementation of computations
- Implementation should not use Pytorch, Tensorflow, Keras, Sklearn etc.

Appropriate penalties will be added if the code does not comply with the specifications.

Submission Instructions

You will submit Assignment1.zip or Assigment1.tar.gz, a compressed archive file containing the following files:

- Layers.py, Model.py, GraphNet.py, Train.py and Utils.py (if any).

Submission is due 09/30/2021,Thursday, 11:59 PM EST. Please use the submit_cse673 script in Timberlake to submit your assignment. You may resubmit your assignment as many times as you want before the deadline, we will use the most recent submission for grading.