Hands-on session 4

Convolution neural network

Instructor: Dr. Souvik Chakraborty

APL 745: Deep Learning for Mechanics

21 February, 2024

Submission due on 05 March, 2024

Instructions:
(i) You are allowed to discuss in a group; however, you must submit your own handwritten homework copy (no computer-typed submission will be accepted). Further, copying homework from your friends is forbidden. If found copied, the homework submission will be considered invalid for all the students involved and will be graded zero.
(ii) Write all the steps, including your reasoning and the formulae you referred to, if any. If sufficient reasoning is not provided and steps are unclear, step marks will not be given.
(iii) For practical submissions, the codes are accepted in *.ipynb* or *.py* format.
(iv) Unless mentioned otherwise, only *numpy*, *pytorch* and *matplotlib* libraries may be used. Direct commands for algorithms to be implemented are not allowed.
(v) The *.rar* file containing all submission related files shall be named in the format, **Name_Entrynumber.rar**

Question 1.
Prerequisites:

1) Download MNIST dataset using *torchvision*. The documentation for the same can be found in this link.
2) Transform the labels provided in the dataset using one hot encoding.
3) Create three datasets with following specifications,
   - Training dataset: 5000 samples
   - Validation dataset: 1000 samples
   - Testing dataset: 2000 samples

Networks:

1) Network **A**:
   Input → Conv1 (with output channels = 16) → Conv2 (with output channels = 32) → Maxpool → Flatten → linear1 (512 hidden units) → linear 2 (128 hidden units) → output.
2) Network **B**:
   Input → Conv1 (with output channels = 16) → ReLU → Conv2 (with output channels = 32) → ReLU → Maxpool → dropout (0.5) → Flatten → linear1 (512 hidden units) → ReLU → dropout (0.5)→ linear 2 (128 hidden units) → output.
3) For both networks: The output layer should have softmax activation. For all conv layers, use a kernel of size 5x5 with no padding and stride = 1, and for all max pool layers, use a window of size 2x2 with no padding and stride = 2. Normalize the input data using min-max scaling. Batch size should be taken as 100.

Tasks:

1) Print the number of trainable parameters for both network configurations.
2) Train both the networks using ADAM optimizer and with the following learning rate and weight decay training configurations:
   - learning rate: 5×10^{-2}, weight decay: 1×10^{-3}
   - learning rate: 1×10^{-4}, weight decay: 1×10^{-5}

3) Compare the performance of Network B given above with that of Network B given in Hands-on 2. Use ADAM optimizer with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-5}. Batch size to be kept equal to 100. Use normalization for inputs.

Additional notes: [**IMP**]

- Pytorch may be used to perform the above tasks.
- Accuracy obtained for all the datasets must be reported. For training and validation datasets, accuracy obtained at only the final epoch may be reported. Also, report training time in all the tasks.
- Cross entropy loss is to be used while training, and a plot showing the epoch vs training loss must be plotted.
- The number of nodes in the input and output layers must be selected according to the dataset under consideration, wherever necessary.

Misc. information for Task 3:
Network B from Hands-on 2: Train a network with 2 hidden dense layers. Each hidden layer must have 150 nodes and ReLU activation. The output layer should have softmax activation.

Question 2. Stock Price Prediction with Recurrent Neural Networks (simple RNN, LSTM and GRU) from scrach
In this task, you will develop a stock price prediction model using simple RNN, LSTM, and GRU architectures implemented from scratch. Below are the detailed instructions:

1) **Dataset:** Download historical stock market data for Netflix (ticker symbol: NFLX) from this link Netflix Stock Data. The dataset includes features such as date, closing price, highest price, and lowest price for 1006 days.
2) **Model Implementation:** Implement simple RNN, LSTM, and GRU architectures from scratch using Python and NumPy. Ensure that your implementations do not use any pre-built RNN model found in *PyTorch* or *TensorFlow*. However, you can use the basic *PyTorch* functionalities, including automatic differentiation (*autograd*).
3) **Data Preprocessing:(5 marks)**
   - Load the dataset and examine its contents to understand the available features (closing price, highest price, and lowest price). Split the data into training and testing data sets such that training data contains features of 700 days and testing data contains features of the rest of the days, which is to be considered for future predictions.
4) **Model Construction:(15 marks)**
   - Implement baseline RNN, LSTM, and GRU models with appropriate configurations for predicting stock prices.
5) **Training Configuration:**($2 \times 5 =$**10 marks)**
   - Define two training configurations:
     - Configuration 1: Learning Rate = 0.005
     - Configuration 2: Learning Rate = 0.001
   - Train each model configuration using the training data. Monitor the training loss and validation accuracy.
6) **Loss Function:(5 marks)**
   - Choose a suitable loss function for training the models. Mean squared error (MSE) or mean absolute error (MAE) are commonly used for regression tasks like stock price prediction.
7) **Evaluation Metric:**
   - After training each model configuration, evaluate the models' performance on a separate validation and test dataset.
8) **Plotting:(10 marks)**
   - Plot the epoch-wise loss for each training configuration (1 and 2) for all three model architectures (RNN, LSTM, GRU). Create separate plots for each model configuration.
9) **Comparison and Analysis:(5 marks)**

- Compare the performance of the three models based on the (accuracy) on the validation and test dataset. Provide an analysis of your findings on the strengths and weaknesses of each model.