Assignment 2

CS5480: Deep Learning
IIT-Hyderabad
Jan-Apr 2018

Max Marks: 40
Due: 9th Mar 2018 11:59 pm

This homework is intended to cover the following topics:

- Implementation of Multi-Layer Perceptrons on contemporary deep learning libraries
- Understanding/implementation of Convolutional Neural Networks and its hyperparameters
- Making deep neural nets work on data

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single file (ZIP), named `<Your_Roll_No>_Assign2`, with all your solutions.

- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 7 grace days for late submission of assignments (with a max of 4 per submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS5480 Marks and Grace Days document under the course Google drive.

- You should use Python for the programming assignments.

- Please read the department plagiarism policy. Do not engage in any form of cheating or plagiarism - if we find such behavior in your submission, both receiver and giver will be imposed with severe penalties. Please talk to instructor or TAs if you have concerns.

Learning PyTorch

Before you begin the questions in the next section, use `http://pytorch.org/tutorials/` to get started on PyTorch. The link above has a 1-hour blitz tutorial, and other examples for you to get introduced to PyTorch.

1 Programming Questions

1. *(24 marks)* **Occupancy Detection using MLPs:** Consider the Occupancy Detection dataset from the UCI Machine Learning repository. This dataset consists of experimental data used for binary classification of room occupancy (i.e., room is occupied versus empty) based on temperature, humidity, light, and CO2 sensors. The training and test data sets are each collected over a week period. Information about the data set and how it has been used in academic publications can be found at `https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+`.

   The training data (8144 in number) and test data (9753 in number) are provided with this document. The data set includes time stamps with date and hour/minute/second within the day. The goal is to determine whether occupancy can be sensed from: (1) temperature, expressed in degrees Celsius, (2) relative humidity, expressed as a %, (3) light, in lux, (4) CO2, in ppm, and (5) the humidity ratio, which is derive from the temperature and the relative humidity. You are not to use time stamp features for predicting occupancy. Since this is a commercial office building, the time stamp is a strong predictor of occupancy.

   (a) *(3 marks)* Use PyTorch to build a feedforward neural net with a single hidden layer and train using backprop and binary cross-entropy to predict occupancy. Your network will have 5 input units and 1 output unit. As a way of testing your code, set the learning rate very small and set N to be the training set size (i.e., you're doing batch training). In this situation, you should be guaranteed that the error monotonically decreases over epochs of training.

   (b) *(1 mark)* As a measure of getting a baseline performance, predict a random label for each input data point in the test. Rather, run 5 trials and use the max vote of the predictions as the baseline method, and obtain a baseline accuracy (for test data). We will use this in the upcoming questions.

   (c) *(4 marks)* Using a network with H=5 hidden units, and mini-batches of size N=100, select a learning rate (or a learning rate schedule) that results in fairly consistent drops in error from one epoch to the next, make a plot of the training error as a function of epochs. On this graph, show a constant horizontal line for the baseline error. (If your network doesn't drop below this baseline, there's something going awry.) Train your net until you're sure the training error isn't dropping further (i.e., a local optimum has been reached). Report the learning rate (or learning rate schedule) you used to produce the plot. Report training and test set performance in terms of % examples classified correctly.

   (d) *(2 marks)* What happens if you use batch gradient descent instead of SGD? Make a plot of the performance for batch GD, and report your observations.

   (e) *(4 marks)* Now train nets with varying size, H, in 1, 2, 5, 10 20. You may have to adjust your learning rates based on H, or use a method for setting learning rates to be independent of H (you can use Adam/Adagrad/RMS/other methods). Decide when to stop training the net based on training set performance. Make a plot, as a function of H, of the training and test set performance in terms of % examples classified correctly.

   (f) *(6 marks)* Report what happens to all the above graphs if you use mean-squared error as the loss function (instead of cross-entropy).

(g) *(4 marks)* Add a second hidden layer, and train a few architectures with 2 hidden layers. Report what architectures you tried (expressed as 5-h1-h2-1, i.e., 5 input, h1 hidden in first layer, h2 hidden in second layer, and one output unit), and which ones, if any, outperform your single-hidden-layer network.

To Submit: Your code and your observations as a report.

2. *(16 marks)* **Convolutional Neural Networks on MNIST:** For this question, you will experiment with convolutional Neural Networks, using PyTorch. You can use the provided PyTorch examples for this assignment (`https://github.com/pytorch/examples/blob/master/mnist/main.py`). Modify the code provided in the tutorials to answer the following questions.

   (a) *(4 marks)* Replace the ReLUs in the code with sigmoid units, and compare the training and testing accuracies of both the models. Discuss the results (i.e., explain why one type of unit performs better than the other).

   (b) *(4 marks)* Compare the accuracy achieved when varying the level of dropout in a CNN. Modify the CNN code to run with the DropOut probability set to 0.25, 0.5, 0.75 and 1 (documentation at `http://pytorch.org/docs/master/_modules/torch/nn/modules/dropout.html`). You can reduce the number of iterations to avoid waiting too long, if required. Report the training and testing accuracy for each DropOut setting. Discuss the results (i.e., explain how the parameter affects the classifier and whether the results are as expected).

   (c) *(4 marks)* Include batch normalization in your ReLU model with the best DropOut setting that you achieved. Report the training and testing accuracy, and your observations on the performance. Now, remove DropOut from your network - what do you see? Report the performance, and your observations.

   (d) *(4 marks)* Lastly, consider your model with ReLU units and batch normalization (no DropOut). Does batch normalization help in different weight initializations (documentation at `http://pytorch.org/docs/master/_modules/torch/nn/init.html`; ensure you include at least one of Xavier's and Kaiming's initializations from the documentation in your studies)? Report the training and testing accuracy, and your observations on the performance.

   To Submit: Your code and your observations as a report.