Assignment 1

CS6360: Advanced Topics in Machine Learning
IIT-Hyderabad
Jan-Jun 2018

Max Points: 60
Due: 1st Jun 2018 11:59 pm

This homework is intended to cover the following topics:

- Basic understanding of implementation of neural networks
- Understanding of training and optimization strategies used to train neural networks
- Understanding of implementation of CNNs and RNNs

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>_Assign1`, 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 12 grace days for late submission of assignments (including project reports). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS6360 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.

1 Programming Questions

1. *(3 + 1 = 4 marks)* **Learning PyTorch:**

   (a) Assuming we have defined `t` as: `t = torch.Tensor([[1,2,3],[4,5,6],[7,8,9]])`. List 3 expressions that can replace the first line below to slice (extract) the middle column from `t`:
   `col = ⋯` *# extract the middle column from t*
   `print col` *# should print the 1-D tensor: 2,5,8*

(b) What is the difference between a `Tensor`, a `Storage` and a `Variable`?

2. *(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.

3. *(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.

4. *(16 marks)* **Implement an RNN:** Choose the text you want your neural network to learn, but keep in mind that your data set must be quite large in order to learn the structure! RNNs have been trained on highly diverse texts (novels, song lyrics, Linux Kernel, etc.) with success, so you can get creative. As one easy option, Gutenberg Books is a source of free books where you may download full novels in a .txt format. We will use a *character-level* representation for this model. To do this, you may use extended ASCII with 256 characters. As you read your chosen training set, you will read in the characters one at a time into a one-hot-encoding, that is, each character will map to a vector of ones and zeros, where the one indicates which of the characters is present:

$$char \rightarrow [0, 0, \cdots, 1, \cdots, 0, 0]$$

   Your RNN will read in these length-256 binary vectors as input.

   (a) *(1 + 2 marks)* **Backpropagation Through Time:** In an RNN with a single hidden layer, you should have three set of weights: W_{xh} (from the input layer to the hidden layer), W_{hh} (the recurrent connection in the hidden layer), and W_{ho} (from the hidden layer to the output layer). Suppose you use *Softmax* units for the output layer and *Tanh* units for the hidden layer, show:

   i. Write the equation for the activation at time step t in the hidden layer and the equation for the output layer in the *forward propagation* step.

   ii. Write the equation for the *weight update rule* at time step t for W_{xh}, W_{hh}, and W_{ho} in a vectorized notation. Suppose the backpropagation goes back to time step $k(0 < k < t)$.

(b) *(3 + 3 + 3 marks)* **Network Training:** Train your recurrent neural network using the dataset you created earlier in this question. You are free to choose learning parameters (sequence length, learning rate, stopping criteria, etc.). Complete the following task:

   i. Report your training procedure. Plot the training loss vs. the number of training epochs.

   ii. During training, choose 5 breaking points (for example, you train the network for 100 epochs and you choose the end of epoch 20,40,60,80,100) and show how well your network learns through time. You can do it by feeding in the network a chunk of your training text and show what is the output of the network. Report your result.

   iii. To add randomness into the outputted sequence, we can either change the beginning character, we can randomly initialize the hidden state or we can introduce randomness via the Softmax function. In this case, if we have already seen sequence $x = (x_1, \cdots, x_t)$ of characters, then the probability that the output will be character j at time-step $t + 1$, i.e. $y_{t+1} = j$, is given by the Softmax function,

$$p(y_{t+1} = j|x) = \frac{e^{x^T w_j}}{\sum_k e^{x^T w_k}} \to \frac{e^{(x^T w_j)/\tau}}{(\sum_k e^{x^T w_k})/\tau}$$

   where $\tau \in (0, \infty)$ is the temperature. Intuitively, at higher temperatures, $\tau \to \infty$, all characters will be chosen equally likely and at low temperatures $\tau \to 0$, only the most probable character will be chosen with nearly probability 1.0. So in order to generate the next character of the sequence, simply sample from this probability. Please report your result of 5-different generated texts of length-100 once your model is fully trained. Try this for three different temperatures t. What dynamics do you expect and what do you see?

(c) *(2 + 2 marks)* **Experiment with Network Structure:** As before, we want to explore how the network learns when we change parameters:

   i. **Number of hidden units.** Try doubling and halving your number of hidden units. Like in part (b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

   ii. **Sequence length.** Try doubling and halving your length of sequence that feeds into the network. Like in part (b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

(d) [**OPTIONAL, EXTRA CREDIT, 10 marks**] **Extension to LSTM:** Generalize your RNN to utilize LSTM units. This is a non-trivial generalization, but you may find the equations of Chapter 4 of Alex Graves thesis to be very useful. http://www.cs.toronto.edu/graves/preprint.pdf. As an additional resource for RNNs, the link `https://github.`

`com/kjw0612/awesome-rnn` is extremely helpful. Show your result in the format of part (b)(i), and discuss your observations.

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