CS 5984: Deep Learning

Homework 3

Due Date: November 16th, 2017 (5:00PM) **Total: 100 Points**

Note: All implementations are required to be accomplished and submitted using the *Jupiter notebook*. Show your results and necessary comments in the notebook. *No handwritten homework is accepted.*

Problem 1: Sequence prediction using RNN/LSTM: [30 Points]

In this problem, you are expected to implement a simple text generator using LSTM in Tensorflow. A sequence generator predicts the next word given a sequence of words.

[NOTE: For details, refer to the paper: Alex Graves, 2013. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.]

Given a sequence of n words = (w_1, w_2, ..., w_n), the model should predict the (n+1)th word w_{n+1} using the trained model. After generating w_{n+1}, use the sequence (w_2, w_3, ..., w_{n+1}) to generate (n+2)th word and so on. You are expected to generate documents of length 50.

Model:

You will build an M-layer LSTM network to predict the sequence using BasicLSTMCell(https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/BasicLSTMCell). Try experimenting with different values of M = {1,2,3}, and how it effects the prediction accuracy.

For the LSTM cell, use num_units = 512. The model should take input N words, and predict the next 50 words starting from N+1 to N+50. Try experimenting with different values of N = {2, 5, 10}.

Training:

The preprocessed text file to train the model has been provided. Use this file to train your LSTM model. The model might take as much as 50,000 iterations to converge.

Testing:

Calculate the prediction accuracy of your model after every 5000 iterations on the training data, and observe how the accuracy increases with the number of iterations. For generating the sequence, select a random index 'i' from the training data, and use the next 5 words as the start sequence. Predict the next 50 words for this sequence using your trained model.

Problem 2: Markov Decision Process [40 Points]

In this problem, you will need to implement Value Iteration and Policy Iteration Algorithms. Consider a grid of size (3,4) as our environment and the only actions allowed are {UP,DOWN,LEFT,RIGHT}. Note that M(1,4) and M(2,4) are the terminal states with a reward value of +10 and -10, respectively (which means that the only possible action that can be taken from these states is to exit the system and collect the reward). In addition, M(3,2) is a Null state which means that it will not have any value and when any of its neighbors take an action to reach this state, they will bounce back to the original state that they are currently in. For now, assume that the policy is deterministic in nature, i.e. when an action is taken to move to a particular next state, it will happen with a probability of 1. For convergence of these iterative algorithms, use a value of 0.01 as the threshold for stopping the iterations, i.e., when the difference between the sum of the values obtained between two consecutive iterations goes below 0.01, then the algorithm should be stopped. Assume a discount factor of 0.9. With this background information, perform the following tasks.

1. Implement Value Iteration (VI) algorithm and run it until convergence. Report the optimal values of each state obtained at convergence.
2. Tweak the VI algorithm and determine the Q-values of each state-action pair and report them in matrix form.
3. Implement Policy Iteration (PI) algorithm and determine the optimal policies. Start with a random policy that just goes up from the current state. Plot the quality of the policy (sum of the values of all the states) at each time a policy is updated using the PI method. Also, report the values obtained at the optimal policy (i.e., upon convergence) and compare with the values obtained using VI method. Are they same?
4. Reduce the discount factor to 0.5 and run the VI algorithm and report how many iterations it takes for convergence. Also, report the new values for each state in a vector form.
5. Now, consider that each transition will have a reward of -1 except for the transitions from the terminal states mentioned above.
6. So, far we assumed a deterministic policy which allows the agent to go to a state that it intended to with a probability of one. Now, let us make the probability as 0.8 for the action to reach the intended state and the probability of reaching the other valid states from the current states will be uniformly distributed (including staying back in the current state). Report the new values and also show if the optimal policy will change.

Problem 3: Q-Learning [30 Points]

Consider a two-state system where there are two possible actions one can take. You will need to download the data Q-learning.dat file which contains 20 samples and each sample is represented by (s_t, a_t, s_{t+1}, r_t). We follow the same notations described in our lecture. Set the discount factor $\gamma=0.9$. The learning rate parameter α is set to 0.1.

a. Sequentially sample the data and update the Q(s,a) values. Report the values after going through the data once.
b. Repeat (a) until Q-values converge i.e., cycle through the samples several times and update the Q-values one at a time. Report the values at convergence.
c. Repeat (a) and (b) by setting $\alpha=1/(w+1)$ where w denotes the number of past updates made to the Q(s,a) value. Note that this w value will be different for each Q(s,a) value as you cycle through the data.