Introduction to Deep Learning, Fall 2018

Homework 5/6: 120 Total Points

Due: November 30th, 11:59PM

Submissions are required electronically through Sakai. Non-code is required in pdf format, and code is required in a Jupyter notebook.

Problem 1: Mathematical Analysis of RNNs (25 points)

(a) Consider a "simple" RNN without a non-linearity:

$$h_t = \boldsymbol{W} h_{t-1} + \boldsymbol{A} x_t$$

Create a situation on $\boldsymbol{W}$ that will lead to an exploding values of the state h_t. What are the general requirements on $\boldsymbol{W}$ for a stable sequence if we don't have a non-linearity?

(b) Consider a "simple" RNN with a tanh non-linearity:

$$h_t = \tanh\left(\boldsymbol{W} h_{t-1} + \boldsymbol{A} x_t\right)$$

Explain how vanishing gradients can occur by considering backprop through a hyperbolic tangent function. Would using sigmoid non-linearity instead help this function?

(c) In class we talked about the "simple" RNN and the Long Short Term Memory (LSTM), but a third commonly used unit is the Gated Recurrent Unit (GRU) (originally proposed here but can be learned about from a variety of sources). Using the same type of structure as we used for the LSTM in class, sketch out the GRU and explain the difference between the GRU and the LSTM.

Problem 2: Recurrent Neural Networks (30 points)

Here we will work on several problems in recurrent neural networks and NLP. First, you should just play with word embeddings to get a greater understanding of what they are and how they work (and when normalized and unnormalized vectors are better).

We would like to evaluate whether the word embeddings are helping us on our sentiment analysis task. On one of the movie review datasets, do the following:

(i) Train an MLP off of the average word embedding to predict sentiment (as done in class) but optimize the network settings to maximize performance

(ii) Train a RNN from the word embeddings to predict sentiment (as done in class) and optimize the network settings to maximize performance

(iii) Encode each vocabulary word as a one-hot vector. Train an MLP on the average of the one-hot vectors.

(iv) As in (iii), but use an RNN on the one-hot encodings.

(v) Why did the word embeddings work better (hint: the word embeddings will work better…)

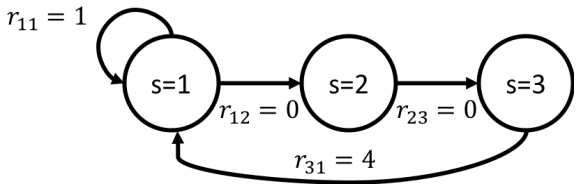
Time-series prediction is an important task, which is commonly approached with RNNs.

(vi) How does cross-validation change when considering a time-series instead of multiple instances (as in our movie reviews)? Only a description is needed.

Much of the strength of deep learning comes from the ability to stitch together different deep learning modules and approaches into a larger framework. In class, we discussed training an RNN against a loss function. Instead, CNNs, RNNs, GANs can all be combined.
(vii) In our previous homework assignment we considered the conditional GAN. In that case, the conditional label was known and given. Instead, consider generating images to match text. One approach could be to use an RNN to encode text to a vector that is fed to a conditional GAN (e.g. http://proceedings.mlr.press/v48/reed16.pdf). Draw a graph (but do not implement) how such a system could work. Any implementation here is completely optional, we are only looking for a description of how this could work.

Problem 3: Understanding the decay term in Q-learning (30 points)
When dealing with reinforcement learning, we have to consider how much to consider the present versus the future. In this question, we will explore simple systems with analytic solutions to understand what the decay term is doing.

(a) Consider the 3-node system below:

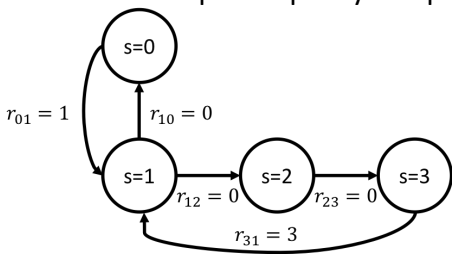


The actions here lead are deterministic state transitions (i.e. heading right from state 1 goes to state 2) and the rewards are deterministic. The only feasible actions are determined by the arrows in the system. The instantaneous rewards are written on each arrow, or can be written in matrix form (infeasible transitions are marked as "-":

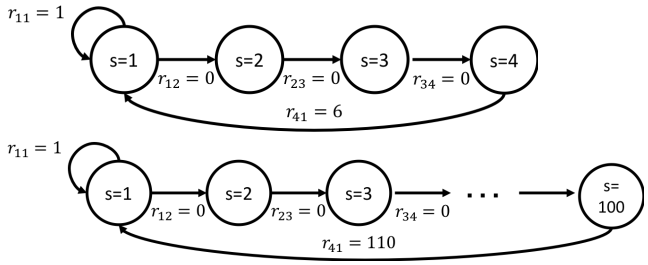
Old\New State	1	2	3
1	1	0	-
2	-	-	0
3	4	-	-

Other than from node 1, there is only a single action from each node. Therefore, the only choice in the policy is from whether to stay still (self-transition) for an instantaneous reward or to move to s=2 for a delayed reward. In this situation, the Q-function can be determined analytically rather than learned.
(i) What is the optimal policy when $\gamma = 1$? Why?
(ii) What is the optimal policy when $\gamma = 0$? Why?
(iii) At what value of γ are both actions from node 1 equally valuable in the Q-function?

(b) How does the optimal policy's dependent on γ change in the setup below?



(c) In part (a) you should see that the decay term controlled whether to take the instantaneous reward or the better reward more infrequently. In that system, it probably seems more natural to take the better reward more infrequently. However, consider the following systems:



In both of these systems, in infinite time you will get more reward by heading to the right from the initial node rather than taking the instantaneous reward. What is the breakeven γ value in these two systems?

(d) Why might we prefer the constant reward rather than the larger reward less frequently?

Problem 4: SARSA Q-Learning (30 points)

SARSA is an alternative way of learning a Q-Value function with a lot of similarity to the simpler policy introduced during the code exercises.

Q-learning update rule:
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)]$$

SARSA update rule:
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot Q^{old}(s_{t+1}, a_{t+1})]$$

Unlike Q-learning, which is considered an *off-policy* network, SARSA is an *on-policy* algorithm. When Q-learning calculates the estimated future reward, it must "guess" the future, starting with the next action the agent will take. In Q-learning, we assume the agent will take the best possible action. SARSA, on the other hand, uses the action that was actually taken next in the episode we are learning from. In other words, SARSA learns from the next action he actually took (on policy), as opposed to what the max possible Q value for the next state was (off policy).

Here, we will use the same setup that we discussed in class with the cart pole problem.

(a) Implement a deep Q-learning approach to the cart pole problem (was done in class) using an ϵ-Greedy approach

(b) Implement a SARSA approach with a deep network to solve the cart pole problem using an ϵ-Greedy approach

(c) Evaluate the impact of γ in the cart pole problem. How important is this parameter? How does it affect stability and learning?

(d) Evaluate the impact of ϵ on the learning over the feasible range (0-1). What values seem reasonable?

(e) Pick one other small agent from the OpenAI gym (https://gym.openai.com/envs/#classic_control) and apply the same techniques

Problem 5: Bookkeeping (5 points)

(a) How many hours did this assignment take you? (There is **NO** correct answer here, this is just an information gathering exercise)

(b) Verify that you adhered to the Duke Community Standard in this assignment (https://studentaffairs.duke.edu/conduct/about-us/duke-community-standard). (I.E. write "I adhered to the Duke Community Standard in the completion of this assignment")