Homework 5/6 – ECE690.06 Deep Learning

Aaron Williams

Due: Nov. 30, 2018

Problem 1: Mathematical Analysis of RNNs

a) Consider this "simple" RNN without a non-linearity.

$$h_t = Wh_{t-1} + Ax_t$$

Exploding gradients can occur across long iterative RNNs due to how backpropagation works. Simply, a W with eigenvalues just larger then 1 across a significant time iteration can cause that value to grow very large.

For each step, backpropagation operates such that $\frac{\partial h^t}{\partial h^{t-1}} = W$ as the function is linear.

After a series of timesteps T, $\frac{\partial h^T}{\partial h^1} = \frac{\partial h^T}{\partial h^{T-1}} * \ldots * \frac{\partial h^2}{\partial h^1}$ which is equal to W^{T-1}

If T = 30 or so, then W^{T-1} can grow large if only a little bit larger than 1. For a stable state, you want the maximum eigen values to be equal to 1 in W.

b) Consider this "simple" RNN with a tanh non-linearity.

$$h_t = \tanh(Wh_{t-1} + Ax_t)$$

While tanh solves the problem of exploding gradients by maintaining that any output h_t be maintained within the range (-1,1), vanishing gradients can still occur across a significant number of iterations.

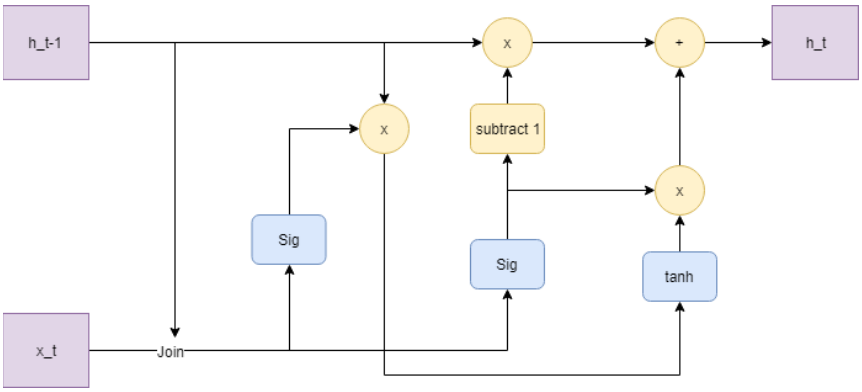
Our backpropagation is like the previous step, except we also add in our tanh.

$$\frac{\partial h^t}{\partial h^{t-1}} = \frac{\partial \tanh(\mathrm{h1} * \mathrm{W})}{\partial (h1 * W)} * W$$

In this case, even if our W is exactly 1, $\partial\tanh$ is bound to the range (0,1), which as shown in part (a), becomes exponentially applied. For this reason, these gradients vanish over time.

$\partial\tanh$ is bound to the range (0,.25), so if anything it has more difficulties in handling vanishing gradients.

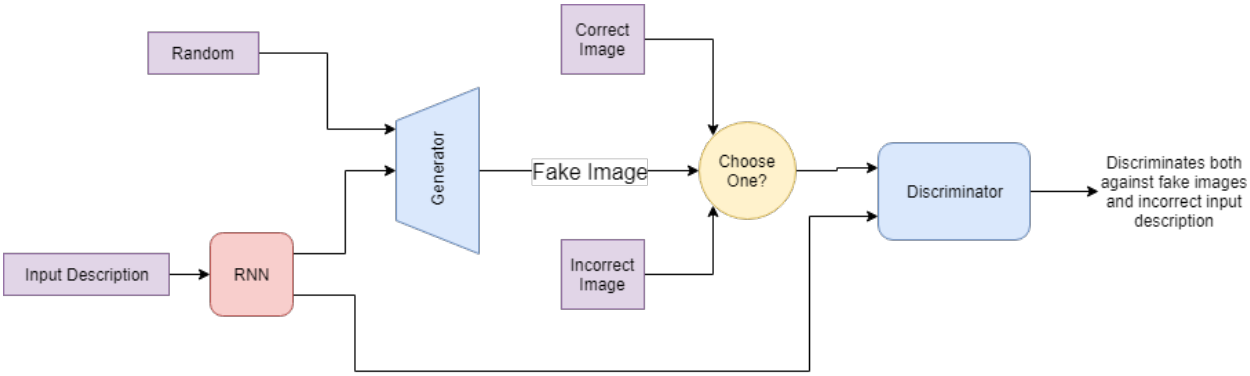
c) Sketch out the GRU and explain the difference between the GRU and the LSTM.



GRU is slightly more simplistic than LSTM. LSTM, for example, has 3 gates and an additional stored set of values which is updated along with h. GRU just has 2 gates, a reset gate and an update gate. Over larger iteration counts, LSTM has more operation time but is better able to preserve long term memory. GRU still does have long term memory components, but operates faster and shorter iteration counts can perform equally as well.

Problem 2: Problem 2: Recurrent Neural Networks

a) 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.



Some output from an RNN is given the generator in addition to the discriminator. Likely, you would use a unique RNN for each and train them through backpropagation as a GAN is normally trained. You give some additional random information to the generator and produce a fake image. You give to the discriminator either the correct image for the given description, another real image but for a different description, or the fake image. You then train loss on for the discriminator against the whatever set of images you selected. This proceeds as a typical GAN is trained.

Problem 3: Understanding the decay term in Q-learning

a) Consider the 3-node system:

i) What is the optimal policy when γ = 1? Why?

Policy is determined by the equation:

$$Q^{new}(s_t, a_t) \leftarrow (1 - a) \cdot Q^{old}(s_t, a_t) + a \cdot \left[r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)\right]$$

When γ = 1, the term $\max_a Q^{old}(s_{t+1}, a)$ is fully considered when determining the value of a future action. Because of that, the algorithm will select the future action which maximizes total reward regardless of the number of state transitions required to earn that reward.

For that reason, the optimal policy would be $r_{12} \rightarrow r_{23} \rightarrow r_{31} \rightarrow r_{12} \rightarrow r_{23} \rightarrow \cdots$

ii) What is the optimal policy when γ = 1? Why?

This time, the estimate of future value $\max_a Q^{old}(s_{t+1}, a)$ is not considered, so the optimal policy will be immediate gratification $r_{11} \rightarrow r_{11} \rightarrow r_{11} \rightarrow \cdots$

iii) At what value of γ are both actions from node 1 equally valuable in the Q-function?

We can estimate the value of a future action by looking at the value of the reward over the number of state transitions required to earn it. Future actions grant a reward of $\frac{4}{3}$ if r_{12} is chosen while r_{11} grants a reward of $\frac{1}{1}$. If set our two rewards $\left[r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)\right]$ equal, it looks like this:

$$1 + \gamma \cdot 0 = 0 + \gamma \cdot \frac{4}{3}$$

So, our γ is $\frac{3}{4}$ if we want those to be qual.

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

If γ is set to 0 in this case, then the state transition at s=1 will be chosen at random as the immediate next reward for s=1 will always be 0. If γ is any larger than 0, than the more long-term policy $r_{12} \rightarrow r_{23} \rightarrow r_{31} \rightarrow \cdots$ will be favored as the estimated reward given by $\max_a Q^{old}(s_{t+1}, a)$ will be higher for that action.

c) Consider the following systems, what is the breakeven γ value in these two systems?

We can apply the same formula as we did in part a with $\left[r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)\right]$.

For system 1:

$$1 + \gamma \cdot 0 = 0 + \gamma \cdot \frac{6}{4}, so\ the\ breakeven\ value\ for\ \gamma \text{ is } \frac{2}{3}$$

For system 2:

$$1 + \gamma \cdot 0 = 0 + \gamma \cdot \frac{110}{100}, so\ the\ breakeven\ value\ for\ \gamma \text{ is } \frac{10}{11}$$

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

A lot of these algorithms are used to play games or solve puzzles, often in competitive settings. If we want to mimic human behavior or keep a chess match competitive, then we want to limit the size of γ. This is averse to making only senseless 30-move in the future actions which to a human opposition is just untenable. A human action, however, will consider a current reward for the next move and potentially some justifiable action some moves down the line.

Problem 5: Bookkeeping

a) How many hours did this assignment take you?

Problem 1: 2.5 hours

Problem 2: 5 hours

Problem 3: 2 hours

Problem 4: 6 hours

Total: ~16 hours

(b) Verify that you adhered to the Duke Community Standard in this assignment.

I adhered to the Duke Community Standard in the completion of this assignment