Gradients through RHS of bellman equation.

—

Anirudh Goyal, Alex

Training Deep Q Networks

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

- Backprop through RHS seems to have very detrimental effect.
- My intuition on why it is bad is because this way we update a better estimate q_{t+1} using a worst estimate q_t
- It does lower the TD error but the Q function ends up being not very good for anything.

Does this make sense ?

- No, we should backpropagate through both sides
- BUT, the empirical evidence proves me wrong.

Why should we backpropagate both sides ?

- We still would be minimizing the same *loss* function, and not some approximation.
- So, doing SGD with a very small learning rate should optimize the loss function.

What TD Learning is doing ?

- TD Learning is not just minimizing the TD error.
- Suppose you have 2 Q functions, Q1 and Q2 and they have TD errors E1 and E2
- If E2 $>$ E1, it does not mean Q1 would be better than Q2 for training the actor.

Has Reward any role to Play ?

- Since, RHS of bellman equation involves reward, what if backpropagating errors to RHS, results in something not good for expected rewards i.e decreases the value of expected reward.
- But I don't think so!
- In fact, updating Q has quite an indirect influence on the expected reward.
- Q is there just for the purpose of estimating the gradient for the policy.
- One issue, that I can imagine is that TD error is not quite the same as the error in estimating the policy gradient, that's why by the way we need the L2 penalty of Q values of rare actions.

Results

(Still training as it takes lot of time to train DQN)