CS11-747 Neural Networks for NLP

Reinforcement Learning for NLP

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phonotron.com/class/nn4nlp2017/
What is Reinforcement Learning?

• Learning where we have an

  • environment X

  • ability to make actions A

  • get a delayed reward R

• **Example of pong:** X is our observed image, A is up or down, and R is the win/loss at the end of the game
Why Reinforcement Learning in NLP?

• We may have a **typical reinforcement learning scenario**: e.g. a dialog where we can make responses and will get a reward at the end.

• We may have **latent variables**, where we decide the latent variable, then get a reward based on their configuration.

• We may have a **sequence-level error function** such as BLEU score that we cannot optimize without first generating a whole sentence.
Reinforcement Learning Basics:
Policy Gradient
(Review of Karpathy 2016)
Supervised Learning

• We are given the correct decisions

\[ \ell_{\text{super}}(Y, X) = - \log P(Y \mid X) \]

• In the context of reinforcement learning, this is also called “imitation learning,” imitating a teacher (although imitation learning is more general)
Self Training

• Sample or argmax according to the current model

\[ \hat{Y} \sim P(Y \mid X) \quad \text{or} \quad \hat{Y} = \arg\max_Y P(Y \mid X) \]

• Use this sample (or samples) to maximize likelihood

\[ \ell_{\text{self}}(X) = -\log P(\hat{Y} \mid X) \]

• No correct answer needed! But is this a good idea?

• One successful alternative: co-training, only use sentences where multiple models agree (Blum and Mitchell 1998)
Policy Gradient/REINFORCE

- Add a term that scales the loss by the reward

$$\ell_{\text{self}}(X) = -R(\hat{Y}, Y) \log P(\hat{Y} \mid X)$$

- Outputs that get a bigger reward will get a higher weight

- Quiz: Under what conditions is this equal to MLE?
Credit Assignment for Rewards

• How do we know which action led to the reward?

• Best scenario, immediate reward:

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \]
\[ 0 \quad +1 \quad 0 \quad -0.5 \quad +1 \quad +1.5 \]

• Worst scenario, only at end of roll-out:

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \]
\[ 0 \quad +1 \quad 0 \quad -0.5 \quad +1 \quad +1.5 \quad +3 \]

• Often assign decaying rewards for future events to take into account the time delay between action and reward
Stabilizing Reinforcement Learning
Problems w/ Reinforcement Learning

- Like other sampling-based methods, reinforcement learning is unstable.
- It is particularly unstable when using bigger output spaces (e.g. words of a vocabulary).
- A number of strategies can be used to stabilize.
Adding a Baseline

- Basic idea: we have expectations about our reward for a particular sentence

<table>
<thead>
<tr>
<th></th>
<th>Reward</th>
<th>Baseline</th>
<th>B-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>“This is an easy sentence”</td>
<td>0.8</td>
<td>0.95</td>
<td>-0.15</td>
</tr>
<tr>
<td>“Buffalo Buffalo Buffalo”</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

- We can instead weight our likelihood by B-R to reflect when we did better or worse than expected

$$\ell_{\text{baseline}}(X) = -(R(\hat{Y}, Y) - B(\hat{Y})) \log P(\hat{Y} \mid X)$$

- (Be careful to not backprop through the baseline)
Calculating Baselines

• Choice of a baseline is arbitrary

• Option 1: predict final reward using linear from current state (e.g. Ranzato et al. 2016)

  • **Sentence-level**: one baseline per sentence
  
  • **Decoder state level**: one baseline per output action

• Option 2: use the mean of the rewards in the batch as the baseline (e.g. Dayan 1990)
Increasing Batch Size

- Because each sample will be high variance, we can sample many different examples before performing update.

- We can increase the number of examples (roll-outs) done before an update to stabilize.

- We can also save previous roll-outs and re-use them when we update parameters (experience replay, Lin 1993).
Warm-start

- Start training with maximum likelihood, then switch over to REINFORCE

- Works only in the scenarios where we can run MLE (not latent variables or standard RL settings)

- MIXER (Ranzato et al. 2016) gradually transitions from MLE to the full objective
When to Use Reinforcement Learning?

• If you are in a setting where the **correct actions are not given, and the structure of the computation depends on the choices** you make:
  
  • Yes, you have no other obvious choice.

• If you are in a setting where **correct actions are not given but computation structure doesn’t change**.
  
  • A differentiable approximation (e.g. Gumbel Softmax) may be more stable.

• If you **can train using MLE, but want to use a non-decomposable loss function**.
  
  • Maybe yes, but many other methods (max margin, min risk) also exist.
An Alternative: Value-based Reinforcement Learning
Policy-based vs. Value-based

- **Policy-based learning**: try to learn a good probabilistic policy that maximizes the expectation of reward

- **Value-based learning**: try to guess the “value” of the result of taking a particular action, and take the action with the highest expected value
Action-Value Function

• Given a state $s$, we try to estimate the “value” of each action $a$

• Value is the expected reward given that we take that action

$$Q(s_t, a_t) = E\left[\sum_{t}^{T} R(a_t)\right]$$

• e.g. in a sequence-to-sequence model, our state will be the input and previously generated words, action will be the next word to generate

• We then take the action that maximizes the reward

$$\hat{a}_t = \text{argmax}_{a_t} Q(s_t, a_t)$$

• Note: this is not a probabilistic model!
Estimating Value Functions

- Tabular Q Learning: Simply remember the Q function for every state and update

\[ Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha R(a_t) \]

- Neural Q Function Approximation: Perform regression with neural networks (e.g. Tesauro 1995)
Exploration vs. Exploitation

- Problem: if we always take the best option, we might get stuck in a local minimum
  
  - Note: this is less of a problem with stochastic policy-based methods, as we randomly sample actions

- Solution: every once in a while randomly pick an action with a certain probability $\varepsilon$
  
  - This is called the $\varepsilon$-greedy strategy

- **Intrinsic reward**: give reward to models that discover new states (Schmidhuber 1991, Bellemare et al. 2016)
Examples of Reinforcement Learning in NLP
RL in Dialog

- Dialog was one of the first major successes in reinforcement learning in NLP (Survey: Young et al. 2013)

- Standard tools: Markov decision processes, partially observed MDPs (to handle uncertainty)

- Now, neural network models for both task-based (Williams and Zweig 2017) and chatbot dialog (Li et al. 2017)
User Simulators for Reinforcement Learning in Dialog

• Problem: paucity of data!
• Solution, create a user simulator that has an internal state (Schatzmann et al. 2007)
• Dialog system must learn to track user state w/ incomplete information

\[
C_0 = \begin{bmatrix}
type = \text{bar} \\
drinks = \text{beer} \\
area = \text{central}
\end{bmatrix}
\]

\[
R_0 = \begin{bmatrix}
name = \quad \\
addr = \quad \\
phone = \quad
\end{bmatrix}
\]

Sys 0
Hello, how may I help you?

\[
A_1 = \begin{bmatrix}
inform(type = \text{bar}) \\
inform(drinks = \text{beer}) \\
inform(area = \text{central}) \\
request(name) \\
request(addr) \\
request(phone) \\
bye()
\end{bmatrix}
\]

Usr 1
I'm looking for a nice bar serving beer.

Sys 1
Ok, a wine bar. What pricerange?

\[
A_2 = \begin{bmatrix}
negate(drinks = \text{beer}) \\
inform(pricerange = \text{cheap}) \\
inform(area = \text{central}) \\
request(name) \\
request(addr) \\
request(phone) \\
bye()
\end{bmatrix}
\]

Usr 2
No, beer please!
Mapping Instructions to Actions

- Following windows commands with weak supervision based on progress (Branavan et al. 2009)

- Visual instructions with neural nets (Misra et al. 2017)
Reinforcement Learning for Making Incremental Decisions in MT

• We want to translate before the end of the sentence for MT, agent decides whether to wait or translate (Grissom et al. 2014, Gu et al. 2017)
RL for Information Retrieval

• Find evidence for an information extraction task by searching the web as necessary (Narasimhan et al. 2016)

<table>
<thead>
<tr>
<th>ShooterName: Scott Westerhuis</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumKilled: 6</td>
</tr>
</tbody>
</table>

A couple and four children found dead in their burning South Dakota home had been shot in an apparent murder-suicide, officials said Monday.

Scott Westerhuis’s cause of death was "shotgun wound with manner of death as suspected suicide," it added in a statement.

The six members of a South Dakota family found dead in the ruins of their burned home were fatally shot, with one death believed to be a suicide, authorities said Monday.

AG Jackley says all evidence supports the story he told based on preliminary findings back in September: Scott Westerhuis shot his wife and children with a shotgun, lit his house on fire with an accelerant, then shot himself with his shotgun.

• Perform query reformulation (Nogueira and Cho 2017)
RL for Coarse-to-fine Question Answering (Choi et al. 2017)

• In a long document, it may be useful to first pare down sentences before reading in depth
RL to Learn Neural Network Structure (Zoph and Le 2016)

- Generate a neural network structure, try it, and measure the results as a reward
Questions?