An Actor-Critic Algorithm for Sequence Prediction

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Ryan Lowe, Joelle Pineau, Aaron Courville, Yoshua Bengio
• Have states $s$, actions $a$, rewards $r$, policy $\pi = p(a|s)$

• Return: $R = \sum_{t=0}^{T} \gamma^t r_{t+1}$

• Value function: $V(s_t) = E_{a \sim \pi}[R|s_t]$

• Action-value function: $Q(s_t, a_t) = E_{a \sim \pi}[R|s_t, a_t = a]$
TD learning

• Methods for policy evaluation (i.e. calculating the value function for a policy)

• Monte Carlo learning: wait until end of the episode to observe the return $R$

$$V(s_t) = V(s_t) + \alpha [R - V(s_t)]$$

• TD(0) learning: bootstrap off your previous estimate of $V$

$$V(s_t) = V(s_t) + \alpha [(r_t + \gamma V(s_{t+1})) - V(s_t)]$$

• $\delta_t = [(r_t + \gamma V(s_{t+1})) - V(s_t)]$ is the TD-error
Actor-Critic

• Have a parametrized value function $V$ (the critic) and policy $\pi$ (the actor)

• Actor takes actions according to $\pi$, critic ‘criticizes’ them with TD error

• TD error drives learning of both actor and critic

(Sutton & Barto, 1998)
Actor-Critic

- Critic learns with usual TD learning, or with LSTD

- Actor learns according to the policy gradient theorem:

\[
\frac{dR}{d\theta} = E_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)]
\]
Actor-Critic for Sequence Prediction

• Actor will be some function with parameters $\theta$ that predicts sequence one token at a time (i.e. generates 1 word at a time)

• Critic will be some function with parameters $\phi$ that computes the TD-error of decisions made by actor, which is used for learning
Why Actor-Critic?

1) Sequence prediction models usually trained with teacher forcing, which leads to discrepancies between train and test time. With actor-critic, can condition on actor’s previous outputs.

2) Allows for the direct optimization of a task-specific score, e.g. BLEU, rather than log-likelihood.
Actor-Critic for Sequence Prediction

• Since we are doing supervised learning, there are a couple differences to the RL case:

  1) We can condition the critic on the actual ground-truth answer, to give a better training signal

  2) Since there is a train/test split, don’t use critic at test time

  3) Since there is no stochastic environment, we can sum over all candidate actions
Notation

• Let $X$ be the input sequence, $Y = (y_1, ..., y_T)$ be the target output sequence
• Let $\hat{Y}_{1,...,t} = (\hat{y}_1, ..., \hat{y}_t)$ be the sequence generated so far

• Our critic $\hat{Q}(a; \hat{Y}_{1,...,t}, Y)$ is conditioned on outputs so far $\hat{Y}_{1,...,t}$, and ground-truth output $Y$

• Our actor $p(a; \hat{Y}_{1,...,t}, X)$ is conditioned on outputs so far $\hat{Y}_{1,...,t}$, and the input $X$
Policy Gradient for Sequence Prediction

- Denote $V$ as the expected reward under $\pi_\theta$

**Proposition 1** The gradient $\frac{dV}{d\theta}$ can be expressed using $Q$ values of intermediate actions:

$$\frac{dV}{d\theta} = \mathbb{E}_{\hat{Y} \sim \hat{p}(\hat{Y})} \sum_{t=1}^{T} \sum_{a \in A} \frac{dp(a|\hat{Y}_{1...t-1})}{d\theta} Q(a; \hat{Y}_{1...t-1})$$
2: while Not Converged do
3: 
4: 
5: 

Receive a random example \((X, Y)\).
Generate a sequence of actions \(\hat{Y}\) from \(p'\).
Compute targets for the critic

\[
q_t = r_t(\hat{y}_t; \hat{Y}_{1:t-1}, Y) + \sum_{a \in \mathcal{A}} p'(a|\hat{Y}_{1:t}, X)Q'(a; \hat{Y}_{1:t}, Y)
\]
Update the critic weights $\phi$ using the gradient

$$\frac{d}{d\phi} \left( \sum_{t=1}^{T} \left( \hat{Q}(\hat{y}_t; \hat{Y}_{1 \ldots t-1}, Y) - q_t \right)^2 + \lambda C \right)$$
7: Update actor weights $\theta$ using the following gradient estimate

$$
\frac{dV(X, Y)}{d\theta} = \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} dp(a|\hat{Y}_{1\ldots t-1}, X) \frac{dQ(a; \hat{Y}_{1\ldots t-1}, Y)}{d\theta} \hat{Q}(a; \hat{Y}_{1\ldots t-1}, Y)
$$
Deep implementation

• For the actor, use an RNN with ‘soft-attention’ (Bahdanau et al., 2015)

• Encode source sentence $X$ with bi-directional GRU

• Compute weighted sum over $x$’s at each time step using weights $\alpha$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$
Deep implementation

• For critic use the same architecture, except conditioned on $Y$ instead of $X$

• Input: the sequence generated so far $\hat{Y}_{1\ldots t}$ and the ground-truth sequence $Y$

• Output: Q-value prediction

Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$. 
Tricks: target network

• Similarly to DQN, use a target network

• In particular, have both delayed actor $p'$ and a delayed critic $Q'$, with params $\theta'$ and $\phi'$, respectively

• Use this delayed values to compute target for critic:

$$q_t = r_t(\hat{y}_t; \hat{Y}_{1..t-1}, Y) + \sum_{a \in A} p'(a|\hat{Y}_{1..t}, X) \hat{Q}'(a; \hat{Y}_{1..t}, Y)$$
Tricks: target network

- After updating actor and critic, update delayed actor and critic using a linear interpolation:

\[
\begin{align*}
\theta' &= \tau \theta + (1 - \tau) \theta' \\
\phi' &= \tau \phi + (1 - \tau) \phi'
\end{align*}
\]
Tricks: variance penalty

- Problem: critic can have **high variance** for words that are rarely sampled

- Solution: artificially reduce values of rare actions by introducing a **variance regularization** term:

\[
C = \sum_a \left( \hat{Q}(a; \hat{Y}_{1\ldots t-1}) - \frac{1}{|A|} \sum_b \hat{Q}(b; \hat{Y}_{1\ldots t-1}) \right)^2,
\]
Tricks: reward decomposition

• Could train critic using all the score at the last step, but this signal is sparse
• Want to improve learning of critic (and thus the actor) by providing rewards at each time step

• If final reward is $R(Y)$, decompose the reward into scores for all prefixes: $R(\hat{Y}_{1,...,1}), R(\hat{Y}_{1,...,2}), ..., R(\hat{Y}_{1,...,T})$
• Then the reward at time step $t$ is:

$$r_t(\hat{y}_t) = R(\hat{Y}_{1...t}) - R(\hat{Y}_{1...t-1})$$
Tricks: pre-training

• If you start off with a random actor and critic, it will take forever to learn, since the training signals would be terrible

• Instead, use pre-training: first train actor to maximize log-likelihood of correct answer

• Then, train critic by feeding samples from the (fixed) actor

• Similar to pre-training used in AlphaGo
Experiments

• First test on a synthetic spelling correction task

• Consider very large natural language corpus, and randomly replace characters with a random character.

• Desired output: sentences spelled correctly

• Use One Billion Word dataset (no chance of overfitting)

• Use character error rate (CER) as reward
Experiments

• Also test on real-world German-English machine translation task

• 153,000 aligned sentence pairs in training set

• Use convolutional encoder rather than bi-directional GRU (for comparison to other works)

• Use BLEU score as reward
## Experiments

<table>
<thead>
<tr>
<th>Setup</th>
<th>Character Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Log-likelihood</td>
</tr>
<tr>
<td>$L = 10, \eta = 0.3$</td>
<td>18.6</td>
</tr>
<tr>
<td>$L = 30, \eta = 0.3$</td>
<td>18.5</td>
</tr>
<tr>
<td>$L = 10, \eta = 0.5$</td>
<td>38.2</td>
</tr>
<tr>
<td>$L = 30, \eta = 0.5$</td>
<td>41.3</td>
</tr>
</tbody>
</table>

**Table 1:** Character error rate of different models on the spelling correction task. In the four setups described, $L$ is the length of input strings, $\eta$ is the probability of replacing a character with a random one.
Experiments

<table>
<thead>
<tr>
<th>Paper</th>
<th>BLEU Log-likelihood</th>
<th>RL training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranzato et al.</td>
<td>17.74 (≤ 20.3)</td>
<td>20.73 (≤ 21.9)</td>
</tr>
<tr>
<td>This work</td>
<td>19.23 (21.33)</td>
<td>21.59 (22.34)</td>
</tr>
</tbody>
</table>

**Table 2:** Our machine translation results compared to the previous work by Ranzato et al. “RL training” stands for the MIXER approach for Ranzato et al. and actor-critic training for this paper. The results with the beam search are reported in the parentheses.
Experiments

Figure 1: Progress of log-likelihood (LL) and actor-critic (AC) training in terms of BLEU score. Behaviour is reported for training (train) and validation (valid) datasets. The curves start from the epoch of log-likelihood pretraining from which the parameters were initialized.
## Experiments

<table>
<thead>
<tr>
<th>Word</th>
<th>Words with largest $\hat{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td>and(6.623) there(6.200) but(5.967)</td>
</tr>
<tr>
<td>of</td>
<td>that(6.197) one(5.668) 's(5.467)</td>
</tr>
<tr>
<td>them</td>
<td>that(5.408) one(5.118) i(5.002)</td>
</tr>
<tr>
<td>i</td>
<td>that(4.796) i(4.629) ,(4.139)</td>
</tr>
<tr>
<td>want</td>
<td>want(5.008) i(4.160) 't(3.361)</td>
</tr>
<tr>
<td>to</td>
<td>to(4.729) want(3.497) going(3.396)</td>
</tr>
<tr>
<td>tell</td>
<td>talk(3.717) you(2.407) to(2.133)</td>
</tr>
<tr>
<td>you</td>
<td>about(1.209) that(0.989) talk(0.924)</td>
</tr>
<tr>
<td>about</td>
<td>about(0.706) .(0.660) right(0.653)</td>
</tr>
<tr>
<td>here</td>
<td>.(0.498) ?(0.291) -(0.285)</td>
</tr>
<tr>
<td>.</td>
<td>.(0.195) there(0.175) know(0.087)</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>.(0.168) $\emptyset$ (-0.093) ?(-0.173)</td>
</tr>
</tbody>
</table>

**Table 3:** The best 3 words according to the critic at intermediate steps of generating a translation. The numbers in parentheses are the value predictions $\hat{Q}$. The German original is “über eine davon will ich hier erzählen.” The reference translation is “and there’s one I want to talk about”.
Questions?