An Actor-Critic Algorithm for Sequence Prediction

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Ryan Lowe,
Joelle Pineau, Aaron Courville, Yoshua Bengio

RL Background

- 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: $V(s_t) = \mathrm{E}_{a\sim\pi}[R|s_t]$
- Action-value function: $Q(s_t, a_t) = \mathrm{E}_{a\sim\pi}[R|s_t, a_t = a]$

TD learning

- Methods for policy evaluation (i.e. calculating the value function for a policy)
- <u>Monte Carlo learning</u>: wait until end of the episode to observe the return R

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

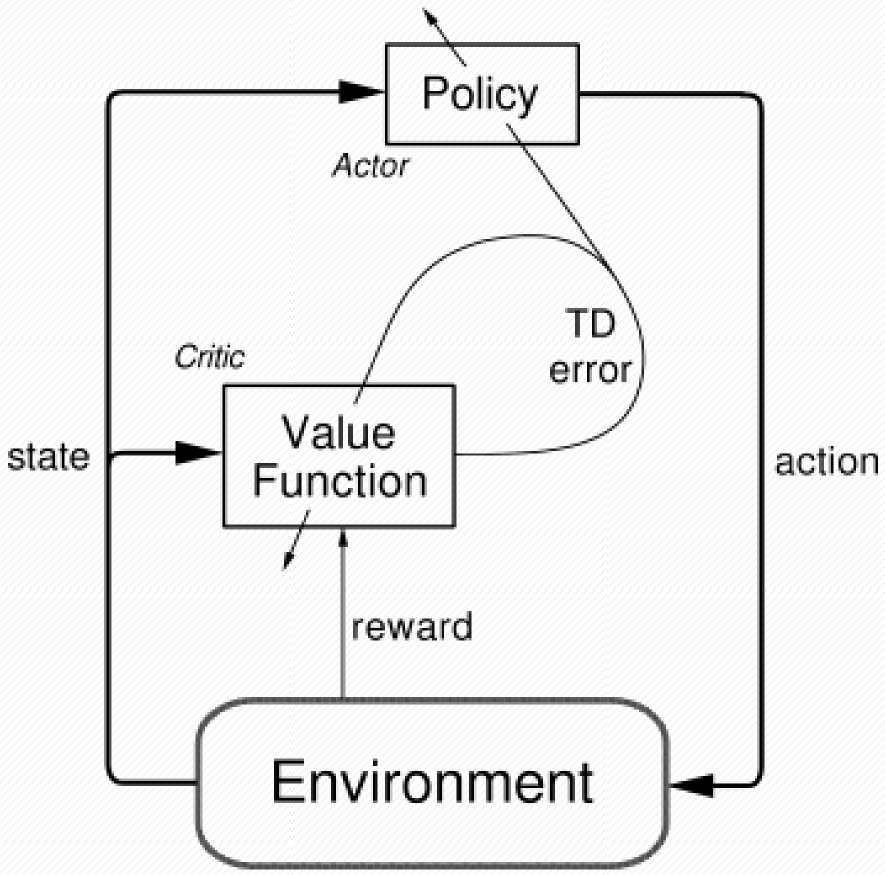
- <u>TD(0) learning:</u> bootstrap off your previous estimate of V

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

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

Actor-Critic

- Have a parametrized value function V (the critic) and policy π (the actor)
- Actor takes actions according to π, critic 'criticizes' them with TD error
- TD error drives learning of both actor and critic



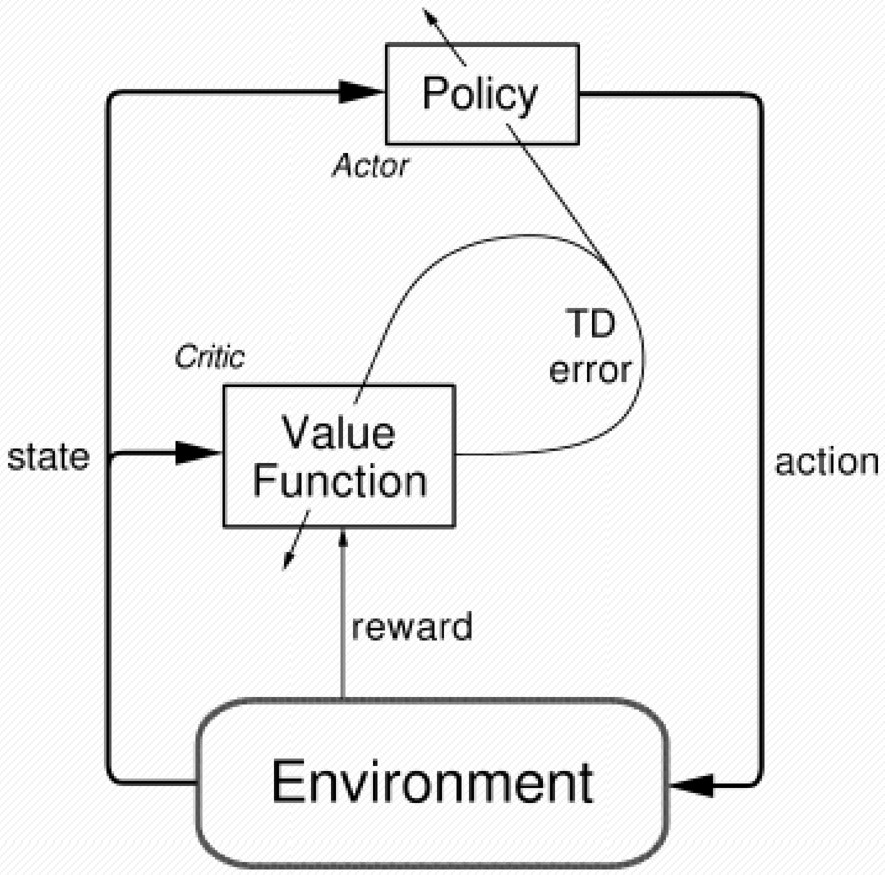
(Sutton & Barto, 1998)

Actor-Critic

- Critic learns with usual TD learning, or with LSTD

- Actor learns according to the policy gradient theorem:

$$\frac{dR}{d\theta} = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$$



Actor-Critic for Sequence Prediction

- Actor will be some function with parameters θ that predicts sequence one token at a time (i.e. generates 1 word at a time)

- Critic will be some function with parameters ϕ 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, $Y = (y_1, \ldots, y_T)$ be the target output sequence
- Let $\widehat{Y}_{1,\ldots,t} = (\widehat{y}_1, \ldots, \widehat{y}_t)$ be the sequence generated so far
- Our critic $\widehat{Q}(a; \widehat{Y}_{1,\ldots,t}, Y)$ is conditioned on outputs so far $\widehat{Y}_{1,\ldots,t}$, and ground-truth output Y
- Our actor $p(a; \widehat{Y}_{1,\ldots,t}, X)$ is conditioned on outputs so far $\widehat{Y}_{1,\ldots,t}$, and the input X

Policy Gradient for Sequence Prediction

- Denote V as the expected reward under π_θ

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

$$\frac{dV}{d\theta} = \mathbb{E}_{\hat{Y} \sim p(\hat{Y})} \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} \frac{dp(a|\hat{Y}_{1 \ldots t-1})}{d\theta} Q(a; \hat{Y}_{1 \ldots t-1})$$

Algorithm

while Not Converged **do**

Receive a random example (X, Y).

Generate a sequence of actions $\hat{Y}$ from p'.

Compute targets for the critic

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

Algorithm

6: Update the critic weights ϕ using the gradient

$$\frac{d}{d\phi}\left(\sum_{t=1}^{T}\left(\hat{Q}(\hat{y}_t;\hat{Y}_{1\ldots t-1},Y)-q_t\right)^2+\lambda C\right)$$

Algorithm

7: Update actor weights θ using the following gradient estimate

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

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 α

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

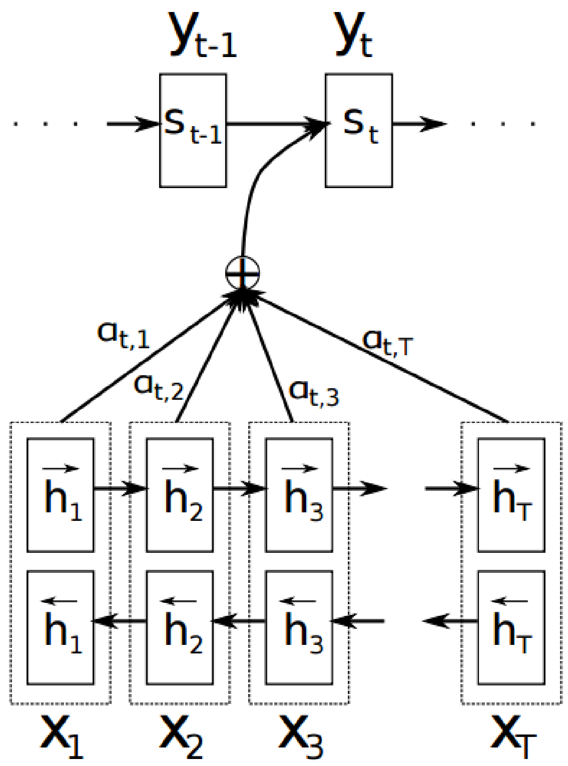


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Deep implementation

- For critic use the same architecture, except conditioned on Y instead of X
- <u>Input</u>: the sequence generated so far $\hat{Y}_{1\ldots t}$, and the ground-truth sequence Y
- <u>Output</u>: Q-value prediction

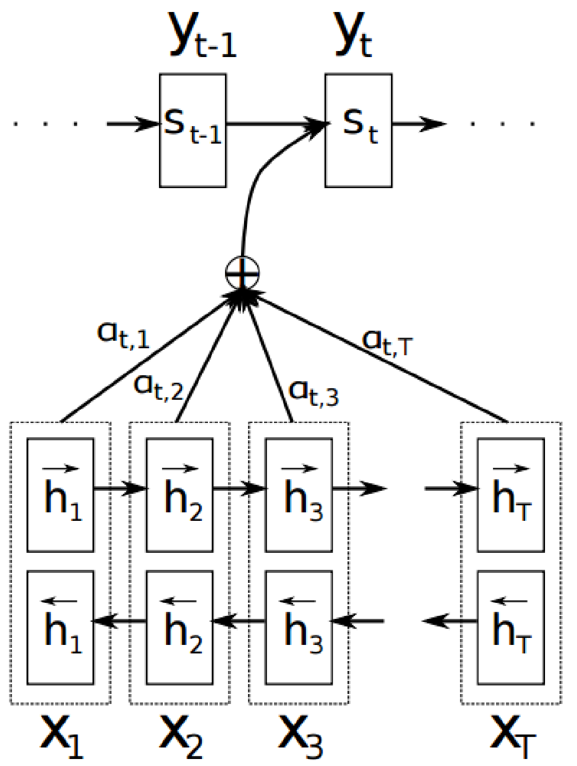


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Tricks: target network

- Similarly to DQN, use a target network
- In particular, have both delayed actor p' and a delayed critic Q', with params θ' and ϕ', respectively
- Use this delayed values to compute target for critic:

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

Tricks: target network

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

8: Update delayed actor and target critic, with a constant $\tau \ll 1$:

$$\theta' = \tau\theta + (1 - \tau)\theta'$$
$$\phi' = \tau\phi + (1 - \tau)\phi'$$

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:

$$C = \sum_a \left(\hat{Q}(a; \hat{Y}_{1 \ldots t-1}) - \frac{1}{|\mathcal{A}|} \sum_b \hat{Q}(b; \hat{Y}_{1 \ldots t-1}) \right)^2,$$

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 $R(\widehat{Y})$, decompose the reward into scores for all prefixes: $(R(\widehat{Y}_{1,\ldots,1}), R(\widehat{Y}_{1,\ldots,2}), \ldots, R(\widehat{Y}_{1,\ldots,T}))$
- Then the reward at time step t is:

$$r_t(\widehat{y}_t) = R(\widehat{Y}_{1\ldots t}) - R(\widehat{Y}_{1\ldots t-1})$$

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

Setup	Character Error Rate	
	Log-likelihood	Actor-Critic
$L = 10, \eta = 0.3$	18.6	17.3
$L = 30, \eta = 0.3$	18.5	17.1
$L = 10, \eta = 0.5$	38.2	35.7
$L = 30, \eta = 0.5$	41.3	37.1

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, η is the probability of replacing a character with a random one.

Experiments

Paper	BLEU	
	Log-likelihood	RL training
Ranzato et al.	17.74 ($\leq$ 20.3)	20.73 ($\leq$ 21.9)
This work	19.23 (21.33)	21.59 (22.34)

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

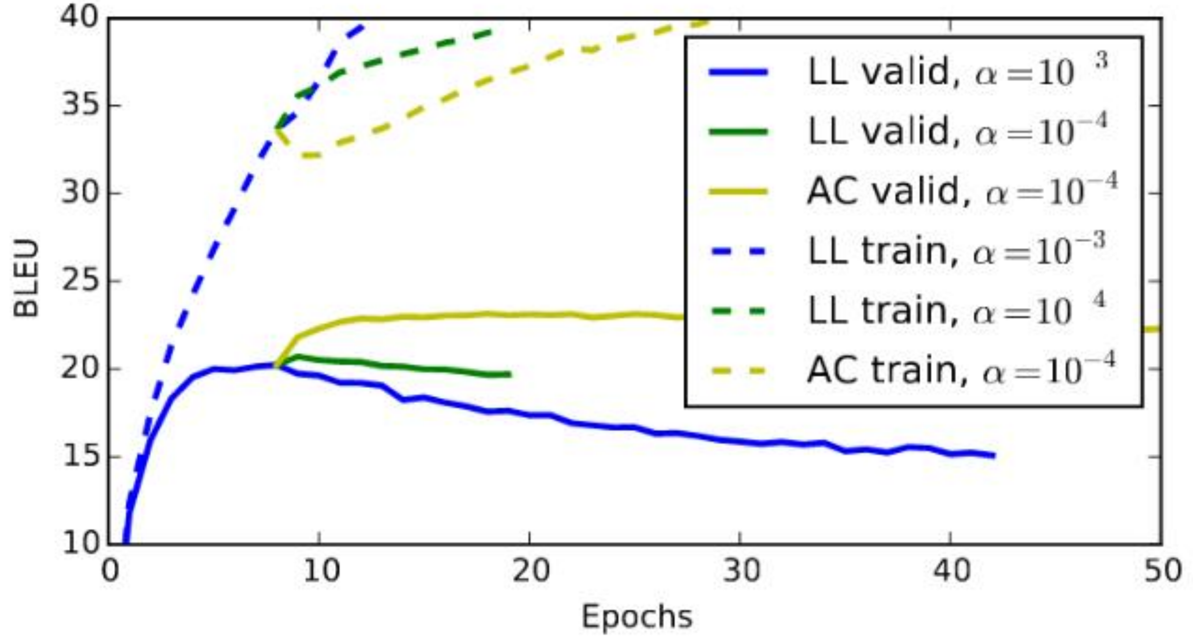


Figure 1: Progress of log-likelihood (LL) and actor-critic (AC) training in terms of BLEU score. Behaviour is reported for training (train) and validation (valid) datasets. The curves start from the epoch of log-likelihood pretraining from which the parameters were initialized.

Experiments

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

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?