An Actor-Critic Algorithm for Sequence Prediction

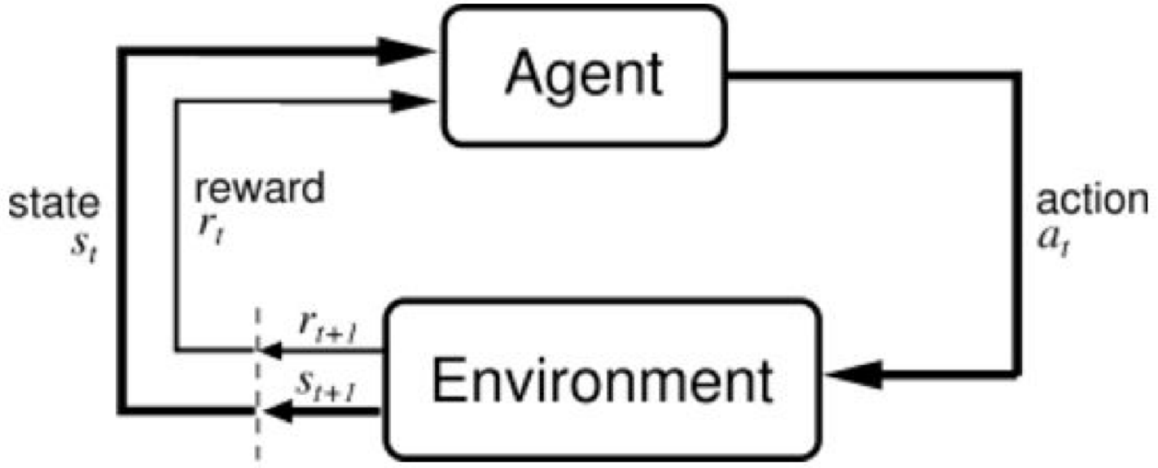
Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Ryan Lowe, Joelle Pineau, Aaron Courville, Yoshua Bengio

Presented By Payam Nikdel

RL Background

- Agent learning sequence of actions(a_t)
- Observes outcomes (state s_{t+1}, rewards r_{t+1}) of those actions
- **Goal**: find a Policy π that maximizes the return R

$\pi = p(a|s)$
R = $\sum_{t=0}^{T} \gamma^t r_{t+1}$ γ: Discount rate
V(s_t): Value function = $E_{a\sim\pi}[R|S_t]$



Policy Evaluation

Method for calculating the value function for a policy

- **Monte Carlo Learning:**

  Wait until end of episode to observe R

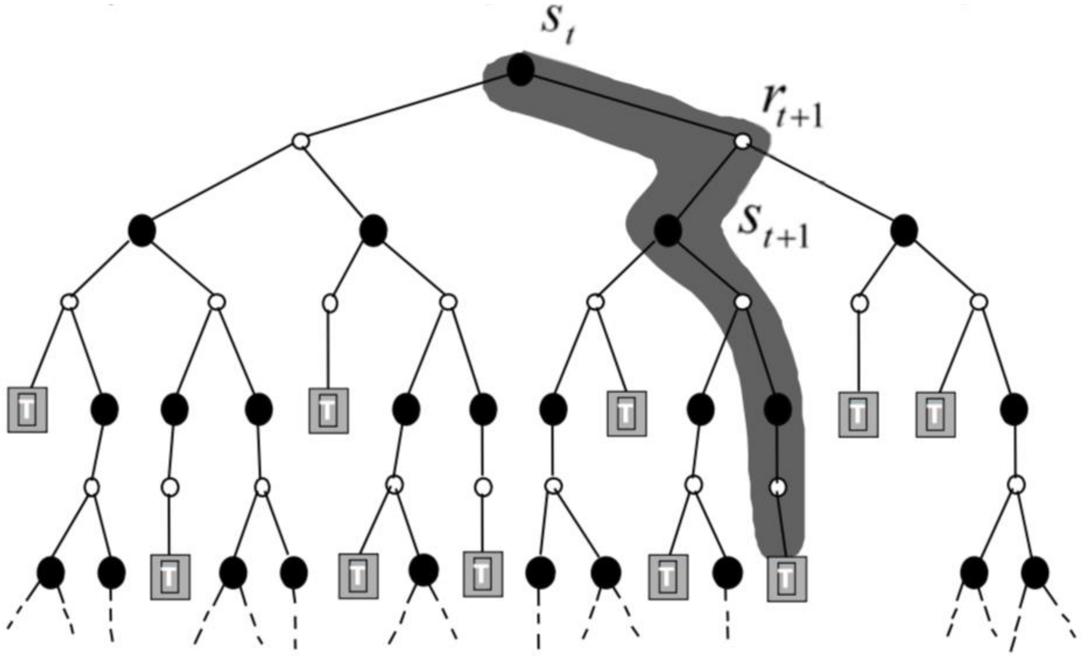
  $$V(s_t) = V(s_t) + \alpha[R - V(s_t)]$$

- **TD(0) Learning:**

  bootstrap off previous estimate of v

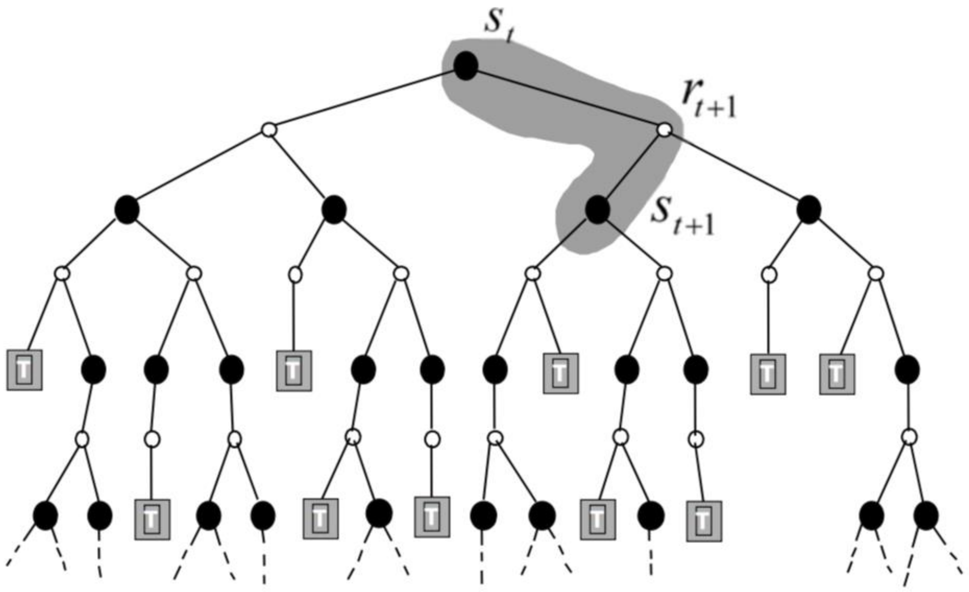
  $$V(s_t) = V(s_t) + \alpha\underbrace{\left[\left(r_t + \gamma V(s_{t+1})\right) - V(s_t)\right]}_{\text{TD-error}}$$

Monte Carlo



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

TD(0)

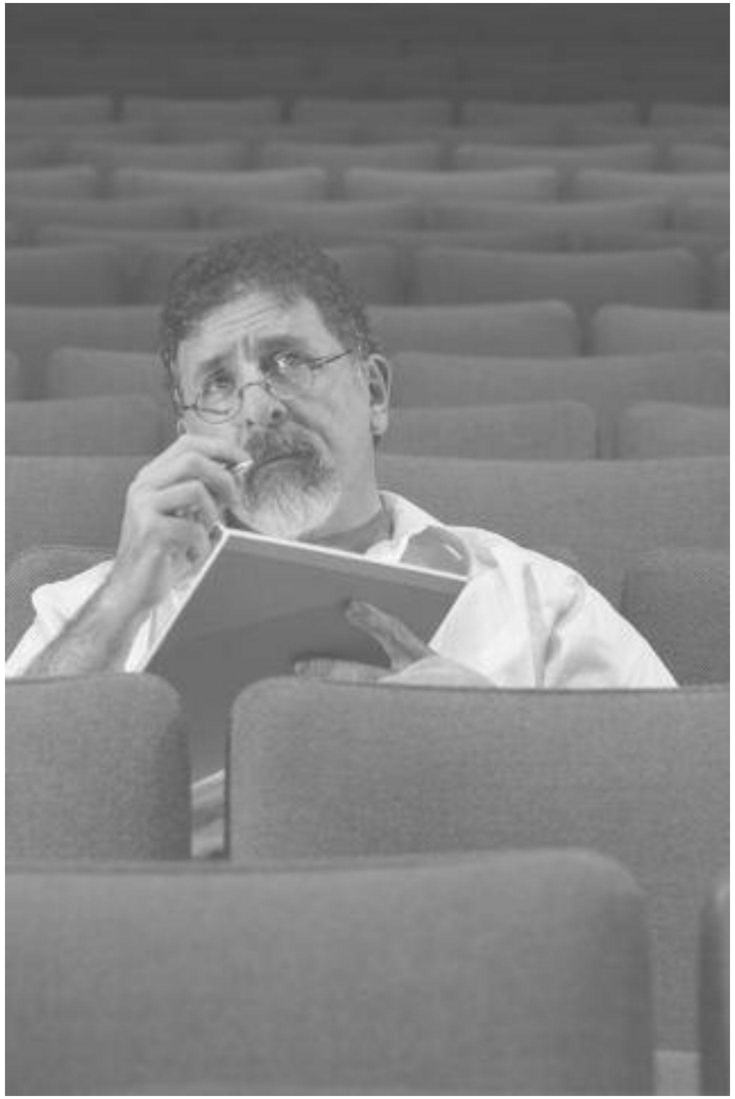


$$V(s_t) = V(s_t) + \alpha[(r_t + \gamma V(s_{t+1})) - V(s_t)]$$

TD learning, adapted from Sutton and Barto (2017)

Input: the policy π to be evaluated
Output: value function V
initialize V arbitrarily, e.g., to 0 for all states
for *each* *episode* **do:**
    initialize state s
    for *each* *step of episode, state* s *is not terminal* **do:**
        a ← action given by π for s
        take action a, observe r, s'
        V (s) ← V (s) + α[r + γ V (s') − V (s)]
        s ← s'
    end
end

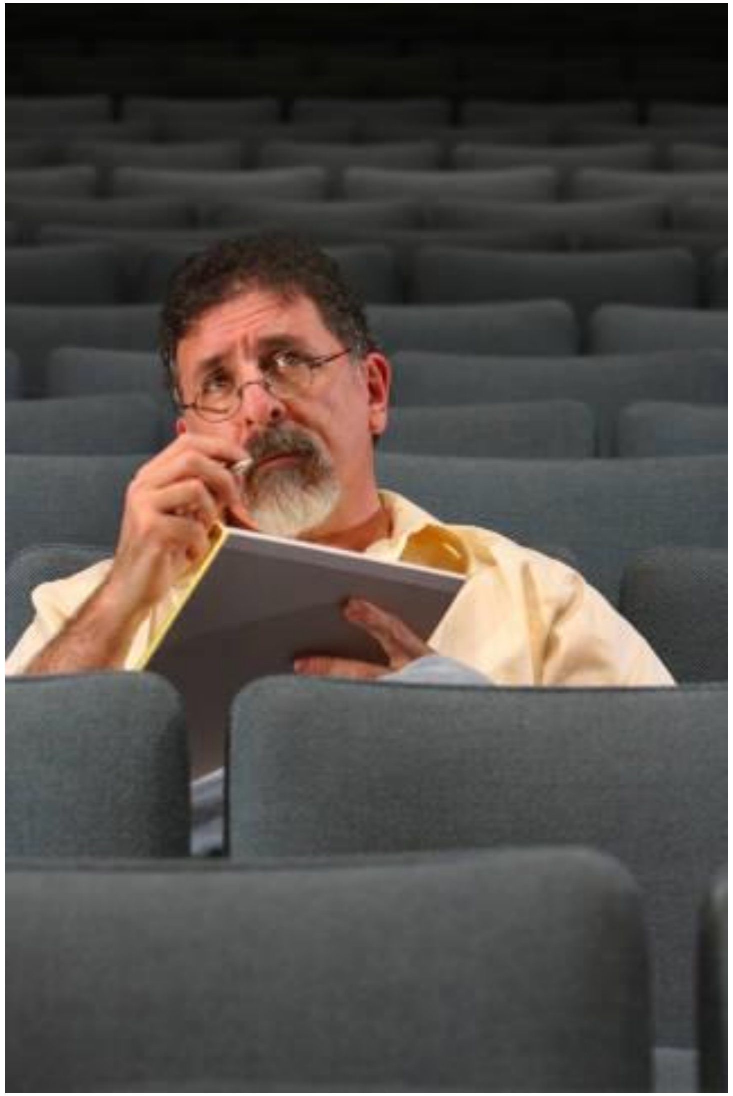
Actor-Critic



Initially: act out some policy

Actor-Critic

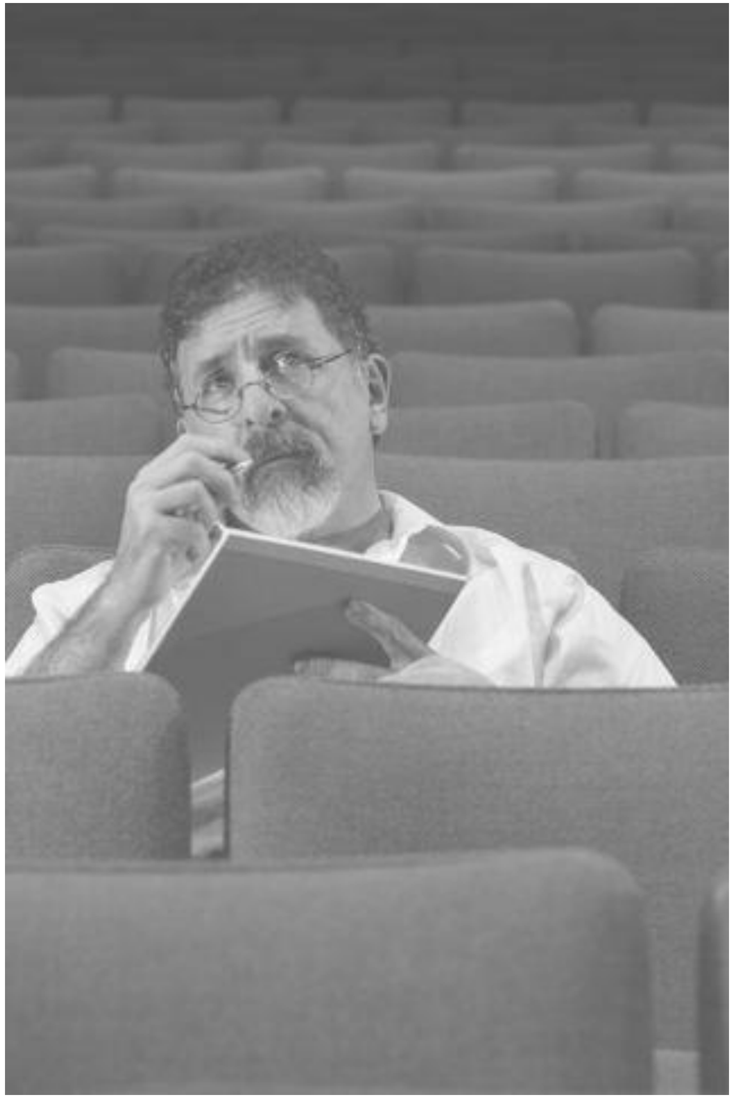
Initially: act out some policy



Critic ≡ policy evaluation

Actor-Critic

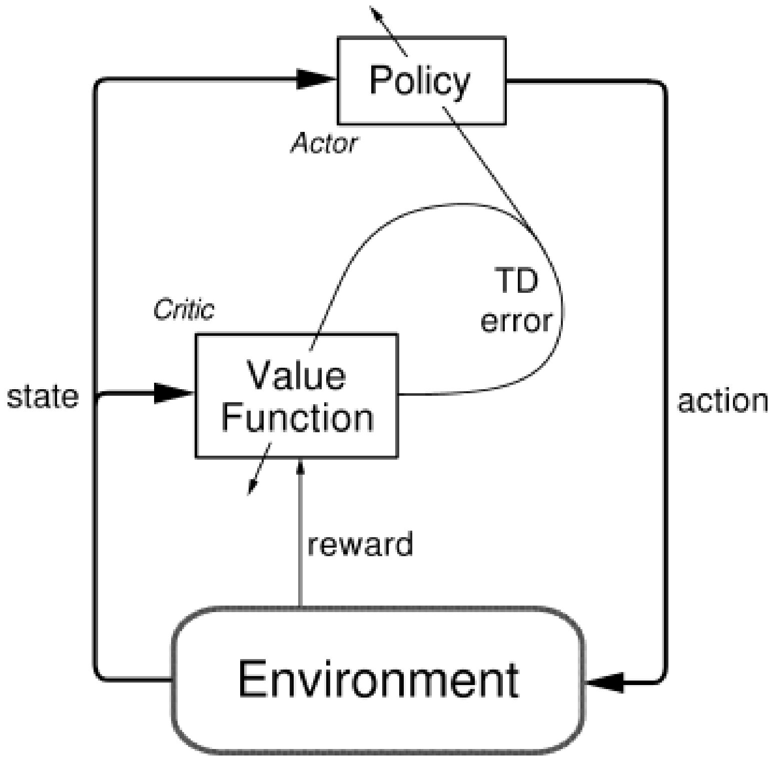
Actor ≡ policy improvement



Critic ≡ policy evaluation

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

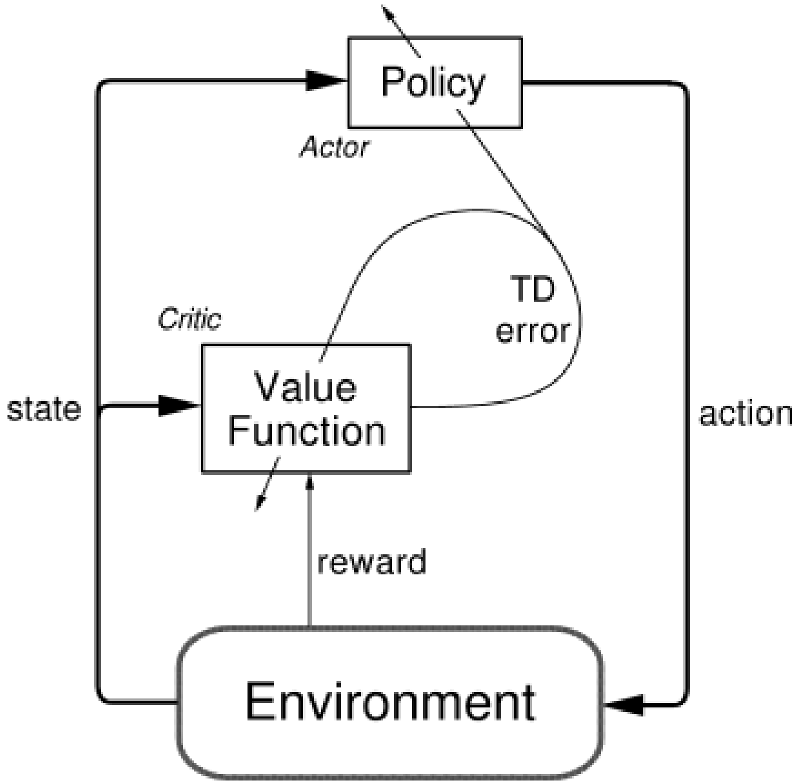


Sutton & Barto, 1998

Actor-Critic

- Critic learns with usual TD learning, or with LSTD

- Actor learns according to the policy gradient theorem

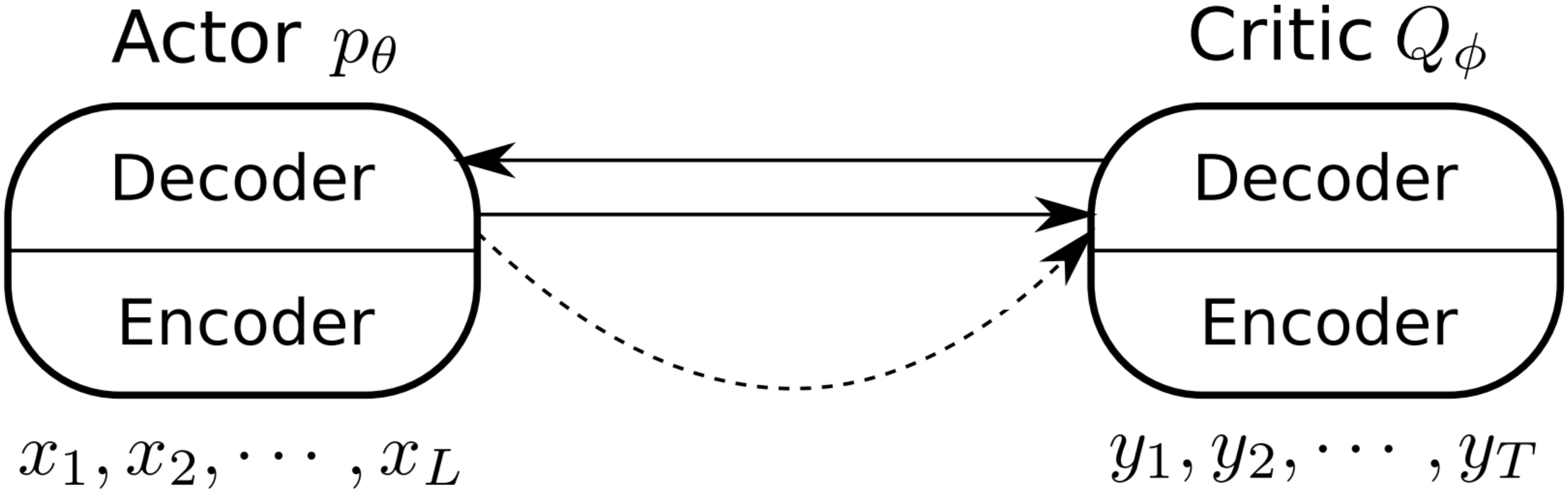


Sutton & Barto, 1998

Actor-Critic for Sequence Prediction

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

- **Critic:** some function with parameters ϕ that computes the TD-error of decisions made by actor, which is used for learning



Actor p_θ
Critic Q_ϕ
Decoder
Encoder
Decoder
Encoder
$x_1, x_2, \cdots, x_L$
$y_1, y_2, \cdots, y_T$

Why Actor-Critic?

- Teacher forcing (like maximum log-likelihood)[1]:
  - During training, tokens come from training data
  - At generation time, tokens are previous generated y
  - ➢ leads to **discrepancies between train and test time**.
  - actor-critic, can condition on actor's previous outputs

- Allows for the direct optimization of a task-specific score, e.g. BLEU, rather than log-likelihood

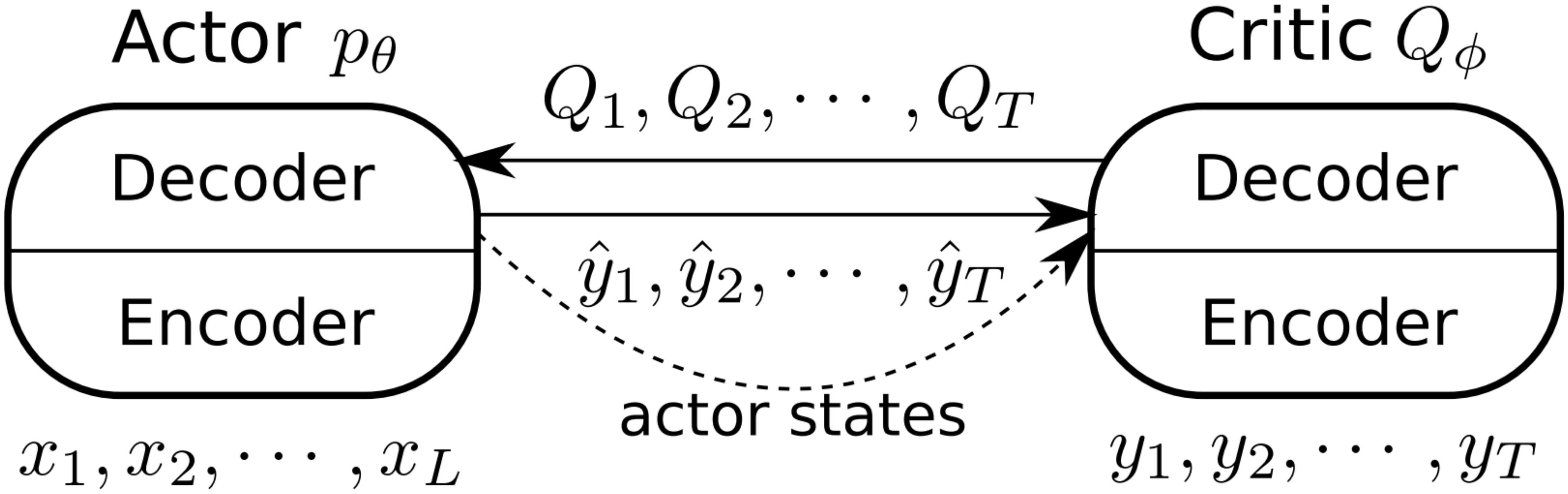
[1]Maximum likelihood training can be suboptimal. (Bengio et al 2015)

Actor & Critic Notation

- Critic:
  - Separate RNN parametrized by ϕ
  - Conditioned on actor generated sequence $\hat{Y}_{1,\dots,t} = (\hat{y}_1, \dots, \hat{y}_t)$ and ground-truth output Y
  - Produces the estimates $Q\hat{}(a; \hat{Y}1\dots_t)$ for all $a \in A$
  - *Critic Q-values are only required during training! No critic in test time*

- Actor:
  - Actor policy $p(a;\hat{Y}_{1,\dots,t},X)$ is conditioned on outputs so far $\hat{Y}_{1\dots t}$ and the input X

Actor & Critic Notation



Algorithm: Policy gradients 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: Policy gradients for sequence prediction

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: Policy gradients for sequence prediction

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: Policy gradients for sequence prediction

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: **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

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

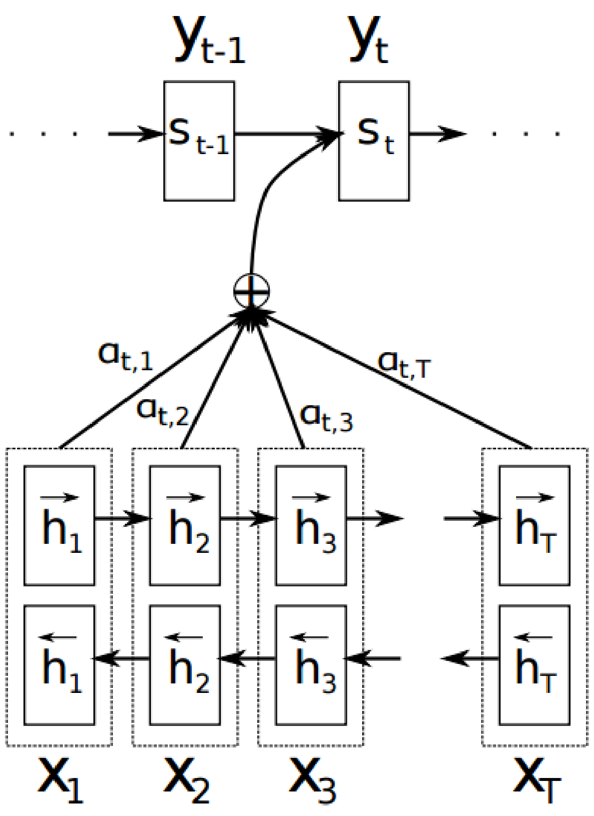


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Deep implementation: **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

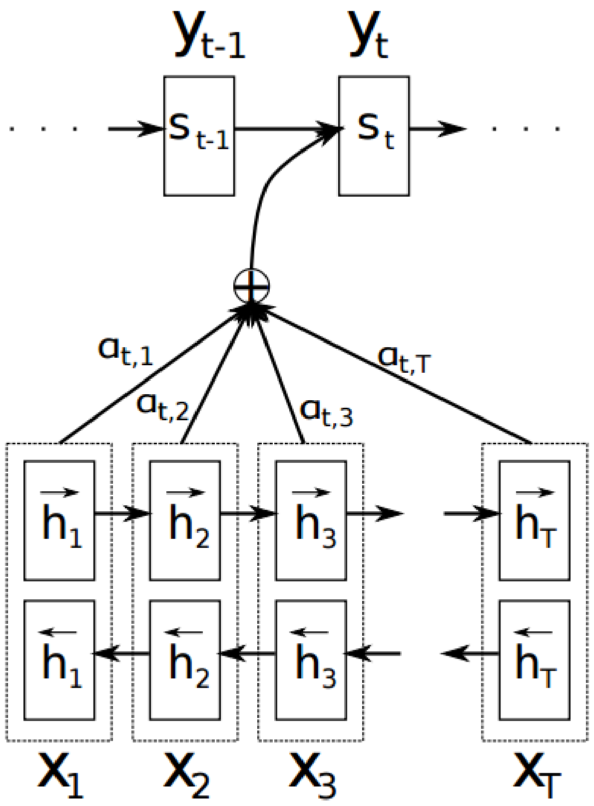


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

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 $R(\acute{Y})$, 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

- It will take forever to learn, if we start off with a random actor and critic
- Use pre-training: first train actor to increase likelihood of correct answer (Maximizing log $p(y_{t+1}|Y_{1\ldots t},X)$)
- Then, train critic by feeding samples from a fixed actor

Experiments

1. Tested on a synthetic spelling correction task
   - Dataset generated by randomly replacing a character using a random character
   - One billion word dataset (no chance for overfitting)
   - Used a character error rate (CER) as reward
   - models:
     - Maximum likelihood
     - Actor-critic
     - Two version of REINFORCE
       I. Exactly as in [1]
       II. Use critic as a base line for REINFORCE

[1] SEQUENCE LEVEL TRAINING WITH RECURRENT NEURAL NETWORKS (Ranzato et al 2015)

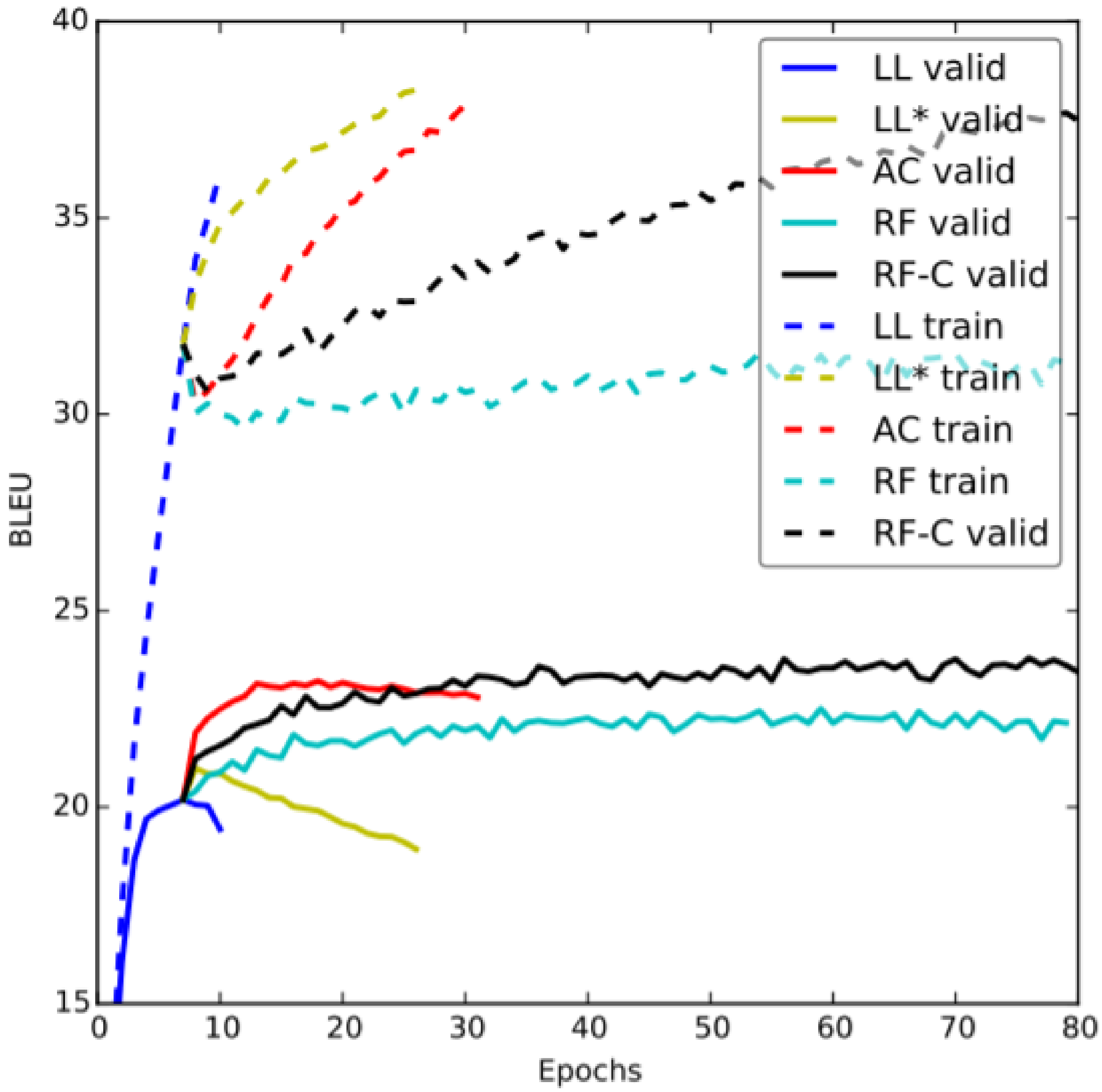


Figure 2: Progress of log-likelihood (LL), REINFORCE (RF) and actor-critic (AC) training in terms of BLEU score on the training (train) and validation (valid) datasets. LL* stands for the annealing phase of log-likelihood training. The curves start from the epoch of log-likelihood pretraining from which the parameters were initialized.

Table 1: Character error rate of different methods on the spelling correction task. In the table L is the length of input strings, η is the probability of replacing a character with a random one. LL stands for the log-likelihood training, AC and RF-C and for the actor-critic and the REINFORCE-critic respectively, AC+LL and RF-C+LL for the combinations of AC and RF-C with LL.

Setup	Character Error Rate				
	LL	AC	RF-C	AC+LL	RF-C+LL
$L = 10, \eta = 0.3$	17.81	17.24	17.82	**16.65**	16.97
$L = 30, \eta = 0.3$	18.4	17.31	18.16	**17.1**	17.47
$L = 10, \eta = 0.5$	38.12	35.89	35.84	**34.6**	35
$L = 30, \eta = 0.5$	40.87	37.0	37.6	**36.36**	36.6

Experiments

2. Tested on a real-world German to English and English to French machine translation
    1. IWSLT 2014 :
        - 153,000 aligned sentence pairs in training set (German to English)
        - Changed the actor to 256 hidden units instead of GRU to compare it with REINFORCE [1] model.
    2. WMT 14 :
        - More
          than 12 million examples

[1] SEQUENCE LEVEL TRAINING WITH RECURRENT NEURAL NETWORKS (Ranzato et al 2015)

Table 3: Our IWSLT 2014 machine translation results with a bidirectional recurrent encoder compared to the previous work. Please see Table 1 for an explanation of abbreviations. The asterisk identifies results from (Wiseman & Rush, 2016).

Decoding method	Model						
	LL*	BSO*	LL	RF-C	RF-C+LL	AC	AC+LL
greedy search	22.53	23.83	25.82	27.42	**27.7**	27.27	27.49
beam search	23.87	25.48	27.56	27.75	28.3	27.75	**28.53**

Table 4: Our WMT 14 machine translation results compared to the previous work. Please see Table 1 for an explanation of abbreviations. The apostrophy and the asterisk identify results from (Bahdanau et al., 2015) and (Shen et al., 2015) respectively.

Decoding method	Model					
	LL'	LL*	MRT *	LL	AC+LL	RF-C+LL
greedy search	n/a	n/a	n/a	29.33	**30.85**	29.83
beam search	28.45	29.88	**31.3**	30.71	31.13	30.37

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)
∅	.(0.168) ∅ (-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?