Reinforcement Learning Applied in Natural Language Processing

Guanlin Li

March 26, 2018

Harbin Institute of Technology, Tecent AI Lab

Table of contents

Value Function Applications

Value-based Reinforcement Learning

Value-based reinforcement learning is to learn a value function with the form $V(s)$ or $Q(s, a)$, where s, a are state representation and action.

Intuition. Within finite (episode) or infinite (discounted reward) RL setting, think of value function as a map from current state (with action) to the expected future reward. Thus, in terms of planning, the predictive model augmented with value estimation has the ability for planning (foresight).

List of use cases

- Decoding with value networks for Neural Machine Translation [2]

Case Study 1: (He et al. 2017 NIPS)

Motivation.

1. Neural Machine Translation (NMT) models use **beam search** during inference, by predicting one word a step, i.e. $p(y_t|y_{<t}, x)$
2. Beam search is greedy and only history-aware (myopic).
3. How about incorporate the **future-aware** step-wise predictive model?
4. EASY! Use value network $Q(y_{<t}, x, y_t)$.

Case Study 1: (He et al. 2017 NIPS)

Algorithm: Naive beam search

input: x, $p(y_t|x, y_{<t})$, and *beam*

$solution = \emptyset$;

while $beam > 0$ **do**

  $p_t = p(y_t|\cdot)$;

  $y_t = TOP_{beam}(p_t)$;

  if $EOS \in y_t$ **then**

    $beam = beam - 1$;

  end

  $solution = solution \cup y_t$

end

return *solution*;

Algorithm: Value-based beam search

input: x, $p(y_t|x, y_{<t})$, $v(x, y_{<t}, y)$, and *beam*

$solution = \emptyset$;

while $beam > 0$ **do**

  $p_t = \alpha \frac{1}{t} p(y_t|\cdot) + (1-\alpha) \log v(y_t, \cdot)$;

  $y_t = TOP_{beam}(p_t)$;

  if $EOS \in y_t$ **then**

    $beam = beam - 1$;

  end

  $solution = solution \cup y_t$

end

return *solution*;

Notice that the only difference lies in an interpolated step-wise **decisioin function**. So we should learn a value function V parameterized by a value network.

Case Study 1: (He et al. 2017 NIPS)

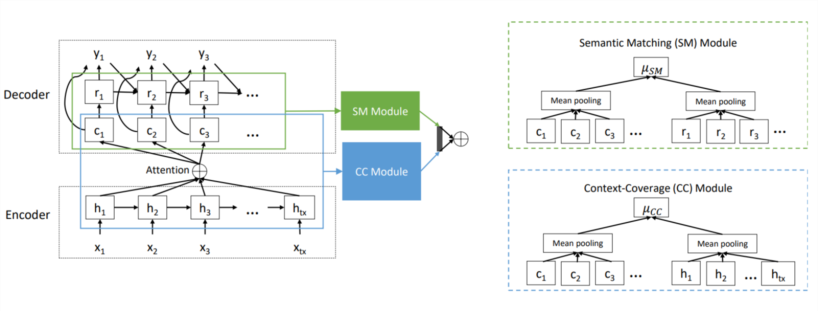

Figure 1: Parameterization of the value function $V_\theta(y_{<t}, x, y_t)$.

In the figure, h_ts are the source side hiddens, c_ts are the context vectors while r_ts are the target side hiddens. (e.g. **RNNSearch**)

- Semantic Matching $\mu_{SM} = f_{SM}(\bar{r}_t, \bar{c}_t)$
- Context Coverage $\mu_{CC} = f_{CC}(\bar{c}_t, \bar{h})$

Case Study 1: (He et al. 2017 NIPS)

Value network is estimated based on an already learned policy $p(y_t|y_{<t}, x)$ through **maximum likelihood**.

The exact value computation at state $(y_{\leq t}, x)$ is:

$$V(y_{\leq t}, x) = \mathbb{E}_{y_{t+1},\ldots,y_T \sim p(y|y_{\leq t}, x)} BLEU(y^*, y') \tag{1}$$

where y' is an action trajectory sampled from policy $p(y|y_{<t}, x)$ and rewarded at the end of an episode (end of sentence).

We use Monte Carlo method to approximate the above equation.

1. **Roll-in** stochastically sample to an intermediate sate $(y_{\leq t}, x)$.
2. **Roll-out** from the intermediate state, use K-beam search to sample K **whole sentence** $y'_{(k)}$.

$\hat{V}(y_{\leq t}, x) \approx \frac{1}{K} \sum_k BLEU(y^*, y'_{(k)})$

Case Study 1: (He et al. 2017 NIPS)

Since, during inference, we do not have access to the ground truth y^*, so we use a parameterized value network $V_\theta(y_{\leq t}, x)$ to approximate it, There are many ways to do so:

1. through mean-square regression.
    - $loss = \left(V_\theta(y_{\leq t}, x) - \hat{V}(y_{\leq t}, x)\right)^2$
2. through Margin Infused Relaxed Algorithm or MIRA [1].
    - $loss = \exp^{V_\theta(y_{\leq t_2}, x) - V_\theta(y_{\leq t_2}, x)}$ if $\hat{V}(y_{\leq t_1}, x) > \hat{V}(y_{\leq t_2}, x)$ else 0
    - which means **rank coherence**

QUESTION: why not use $V_\theta(y_{\leq t_2}, x) - V_\theta(y_{\leq t_2}, x)$? (Hint: the p_t in beam search.)

Case Study 1: (He et al. 2017 NIPS)

Performance.

Table 1: Overall Performance

	En→Fr	En→De	Zh→En NIST06	Zh→En NIST08	En→Fr Deep
NMT-BS	30.51	15.67	36.2	29.4	37.86
NMT-BSO	31.23	16.64	36.59	30.5	–
NMT-VNN	**31.54**	**16.97**	**37.6**	**31.22**	**38.19**

Case Study 1: (He et al. 2017 NIPS)

Stability to beam size and insensible to $\alpha > 0.5$

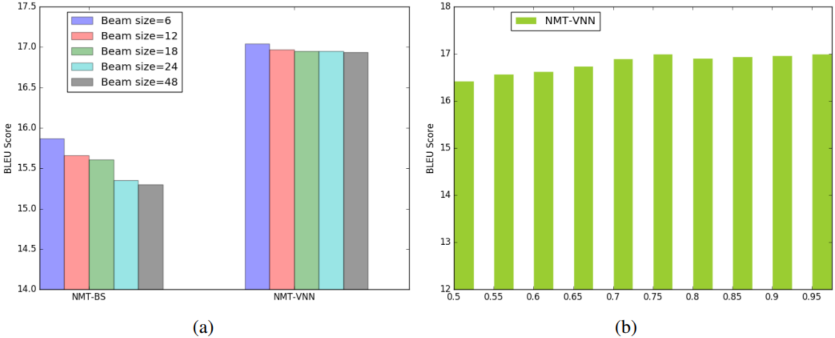

Figure 3: (a). BLEU scores of En→De task w.r.t different beam size. (b). BLEU scores of En→De task w.r.t different hyperparameter α.

Case Study 1: (He et al. 2017 NIPS)

- What if only use value function for decoding (not mixed with policy)?
- What if continue (iteratively) training the policy and the value network? Equal to AC?

Forecast

To be continue... (more RL applications)[1]

A notes page: RL tutorials and applications on NLP.

[1] maybe later in April or May.

Questions?

Backup slides

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

metropolis will automatically turn off slide numbering and progress bars for slides in the appendix.

References I

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer.
Online passive-aggressive algorithms.
Journal of Machine Learning Research, 7(Mar):551–585, 2006.

Y. Xia, F. Tian, L. Wu, J. Lin, T. Qin, N. Yu, and T.-Y. Liu.
Deliberation networks: Sequence generation beyond one-pass decoding.
In *Advances in Neural Information Processing Systems*, pages 1782–1792, 2017.