Routing Paper

Wyatt Jones

University of Iowa

August 30, 2018

1 Introduction

Talk about how RL is very general method to solve DP problems, is really cool, and can do amazing things (GO, ATARI) Talk about application to OR with TSP and how it is a big problem, talk about how feas constraint is annoying, and large cont state space, very applicable to a lot of problems Why isnt it applied more? Discuss the problems that arise when doing RL research initial policy parameterization matters, many hyperparamters, hard to select network architecture, hard to evaluate if the architecture is capable of learning the policy (SR vs RL), local min, sensitive to random seed, many different RL algorithms with new advances happening frequently, training is computationally intensive and so cant try everything, people dont write about what didnt work.

A major complication when trying to select the appropriate architecture for your neural network that will be trained using RL is that there is often not the necessary feedback to iteratively improve the architecture until the network is training optimally. This is due to the subset of architectures that will effectively train on a given problem is very small compared to the number of potential architectures and that when the architecture is not close enough the feedback that the researcher observes is that the network simply doesn't improve while training is taking place. This makes it difficult to evaluate the value of different changes to NNA when the NN is just not learning.

The researcher can use SL to evaluate the value of a given NNA with the hope that if a given NNA can be trained using SL on a smaller problem then it will work using RL on the larger problem that is of importance. In order to study this I used SL to train several different NNA and then used RL to try and train the same NNA and compared the results.

2 RL Review

intro to RNN, REINFORCE, Actor-Critic, A3C, GA3C

3 Experimental Design

add subsections cite bill cooks book for describing TSP

$$p(y_1, \dots, y_T | x_1, \dots, x_T) = \prod_{t=1}^{T} p(y_t | y_1, \dots, y_{t-1}, c)$$

Let $x = (x_1, \dots, x_T)$ be a sequence where x_i is the x-y coordinates for point i, y_t is the id of the point traveled to at time t and c is a context vector.

The encoder maps the input sequence x into a context vector c. The approach used in this paper is to use a RNN composed of LSTM cells such that

$$h_t = f(x_t, h_{t-1})$$

where f is an LSTM and $c = h_T$.

The decoder does is trained to predict the probability of choosing the next location

$$p(y_t | y_1, \dots, y_{t-1}, c) = g(y_{t-1}, s_t, c)$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

Bidirectional consists of forward and backward RNN's. The forward RNN reads the input sequence in order from $t = 1$ to T and the backward reads the input sequence from T to $t = 1$. The hidden state h_j is then constructed by concatenating the two vectors $h_j = [\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top]$

The final architecture is from cite Pointer Networks, where there is no longer an encoder and the probability of moving to the next point is simplified to the following

$$p(y_t|y_1,\ldots,y_{t-1},c) = \text{softmax}(c)$$

where h_t is the LSTM's hidden state at step t and c is the context vector. The context vector is calculated using the following formulas.

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

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

where a is an attention mechanism. For this paper I study the affect that two different attention mechanisms have on the learning process. The first attention mechanism is the Bahanadu Attention Mechanism from cite https://arxiv.org/pdf/1409.0473.pdf

$$e_{ij} = v_\alpha^\top \tanh(W_\alpha s_{i-1} + U_\alpha h_j)$$

where $v_\alpha \in \mathbb{R}^n, W_\alpha \in \mathbb{R}^{n\times n}, U_alpha \in \mathbb{R}^{n\times 2n}$ are weight matricies.

The second attention mechanism is the Luong Attention Mechanism from cite Effective Approaches to Attention-based Neural Machine Translation.

$$e_{ij} = s_{i-1}^\top W_\alpha h_j$$

The parameterized policy function is optimized using the ADAM optimizer which is common in this literature. To examine the sensitivity to hyperparameter choices I examine the affect that a learning rate of 1e-2, 1e-3, and 1e-4 has on the ability for each architecture to learn the optimal policy. I also examine how the number of LSTM cells affects the training process by training the neural networks with 32, 64, and 128 LSTM cells.

4 Results

(Main Idea is to show how it is necessary to build a specific architecture to solve specific problems) while the method is general it's implementation is anything but general (Sub Ideas SL can be used to narrow subset but not all SL will work for RL)

The first NNA that I studied was a unidirectional encoder/decoder framework for NMT. This method was able to train using SL but did not work when the state was embeded, bahanadu attention was used, or when the policy was not decoded using a greedy decoder.

The second NNA that I studied was a variation of the first except with a bidirectional encoder. This method was able to train using SL but did not work when the state was embeded, bahanadu attention was used, or when the policy was not decoded using a greedy decoder.

The third NNA that I studied was a variation of the second except with a stack bidirectional encoder. This method was able to train using SL but did not work when the state was embeded, bahanadu attention was used, or when the policy was not decoded using a greedy decoder.

The fourth NNA that I studied is the current best for this framework taken from NCO, PN. It uses that attention mechanism as the output and was able to be trained using SL and worked with state embed, both Luong and Bah. Talk about initial policy prediction between NNA 1, 2, 3 and 4. Talk about how state embed changes that. Talk about the difference between Luong and Bah.

Methods that work for SL 1, 2, 3, 5-10 1) working 1, not working 12, 13, 32 doesnt work with state embed, stochastic, (does it work with bah?)

2) working 3, 50 not working 15, 90 doesnt work with stochastic, same batch, (does it work with bah?, state embed?)

3) working 17, 30, not working 5

5-10) working 9, 21, 22, 46, 47, 56, 58, 62, 63, 77, 78, 81, 82, 83 not working 41, 60, 76, 79, 80 check mod=9, does work with state embed, bah doesnt work with rnn 32, lr˙decay˙off, maxgrad 0, time˙input working with bah 46, 47, 56, 62, 63, 81, 82, 83 working with luong 9, 21, 58, 77 (not sure about 9, 21 but maybe only works with state embed)

Methods that work for RL 5-10 5-10) working 22, 34, 35, 37, 51, 52, 53?, 61, 64 (these are all identical architectures except for 64) not working 10, 11, 23, 24, 25, 26, 27, 36, 38, 39, 40, 42, 43, 44, 45, 54, 55, 57, 65, 66, 84, 85, 86, 87, 88, 89, 91, 92, 93, 94, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110 working with bah 64 working with luong 22, 34, 35, 37, 51, 52, 53, 61, 64 not working with bah 44, 45, 57, 65, 94, 99 not working with luong 10, 11, 23, 24, 25, 26, 27, 36, 38, 39, 40, 42, 43, 54, 55, 66, 84, 85, 86, 87, 88, 89, 91, 92, 93, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110

never worked ideas PPO, moving average, sequence cost, use time, direction 4 or 6, beam search, PCA

5 Conclusion

Mnih et al. (2016)

References

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).