Decipering AlphaStar on StarCraft II

Yekun Chai

Institute of Automation, Chinese Academy of Sciences

yekun.chai@ia.ac.cn

July 23, 2019

Overview

1. Introduction
   - An introduction to StarCraft II
   - The challenge of StarCraft II
2. How AlphaStar is trained (expected)
   - Toss things together
   - Training

An Introduction to StarCraft II

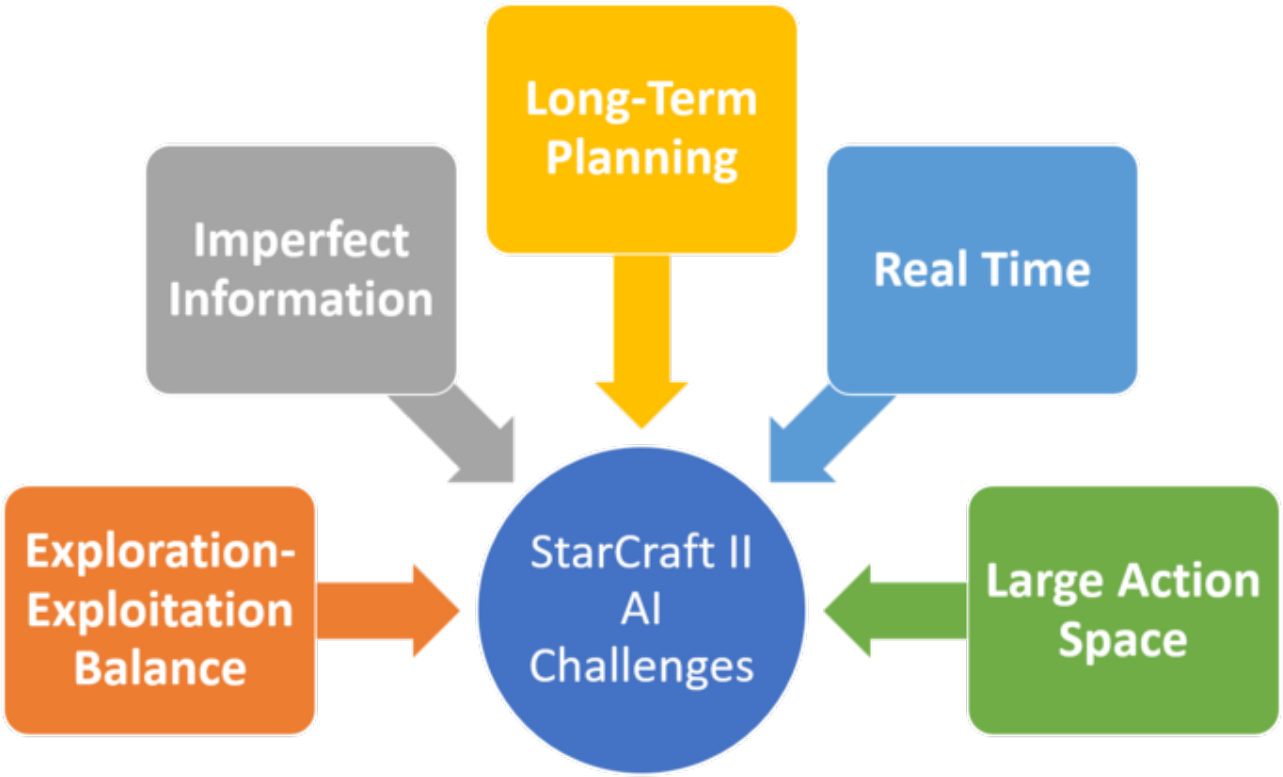
- As a Real-Time Strategy(RTS) games, StarCraft II requires selecting one of three different alien races (i.e. Zerg, Protoss or Terran) with different characteristics and abilities.
- Each player starts with a number of worker units, which gather basic resources to build more units and structures and create new technologies.
- To win,a player must carefully balance
  Macro - *big-picture management of their economy*
  Micro - *low-level control of individual units/agents*

The challenge of StarCraft II

Balance the short and long-term goals and *adapt to unexpected situations* requires:

- **Game theory** - continual exploration and exploitation
- **Imperfect information** - partially observed environments require "actively scouting"
- **Long-term planning** - credit assignment problems
- **Real time** - continually perform actions for each player
- **Large action spaces** - a myriad of units and buildings to be controlled simultaneously; a mass of combinatorial and hierarchical space for decision-making.

The challenge of StarCraft II



Toss things together

AlphaStar aggregates a couple of sota approaches in terms of the *long-term sequence modeling* and *large output dimension*, e.g. machine translation, language modeling, visual representations.

- Transformer
- Relational inductive biases
- LSTM core
- Auto-regressive policy head
- Ptr Net
- Centralized value baseline - COMA policy gradient

Training

Imitation learning (supervised)

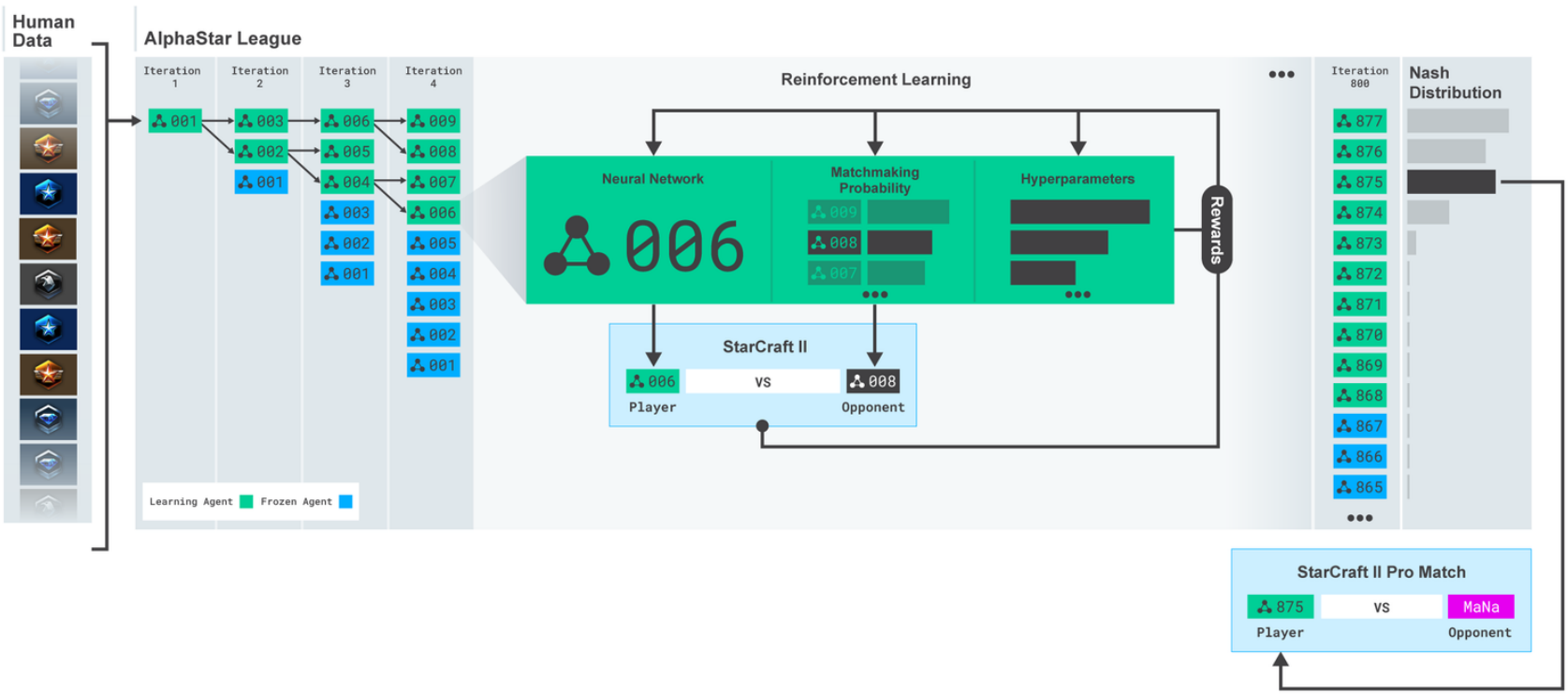
- AlphaStar is initially trained with **imitation learning** with anonymous game replays from human experts
- This offers a good initialization for neural networks
- This initial agent beat the built-in "Elite" level AI ($\approx$ human golden level)

Reinforcement learning

- Seed a multi-agent reinforcement learning process with a continuous league
- Adopt *population-based training* (PBT) and *multi-agent* RL
- The league training can be treated as a bootstrapping DAgger process

AlphaStar league training

- Form a continuous league, with agents compete with each other
- New agents were dynamically supplemented to the league, by branching from existing competitors; each agent then learns from games against other competitors.
- Each agent would play against the strongest strategies and does not forget how to defeat the earlier version of agents.



Transformer

Background

- End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence. However, sequence-aligned RNNs preclude parallelization.

Solution

Transformer applies these approaches in MT tasks:

- stacked self-attention
- point-wise fully-connected layers
- Residual connection
- Layer normalization

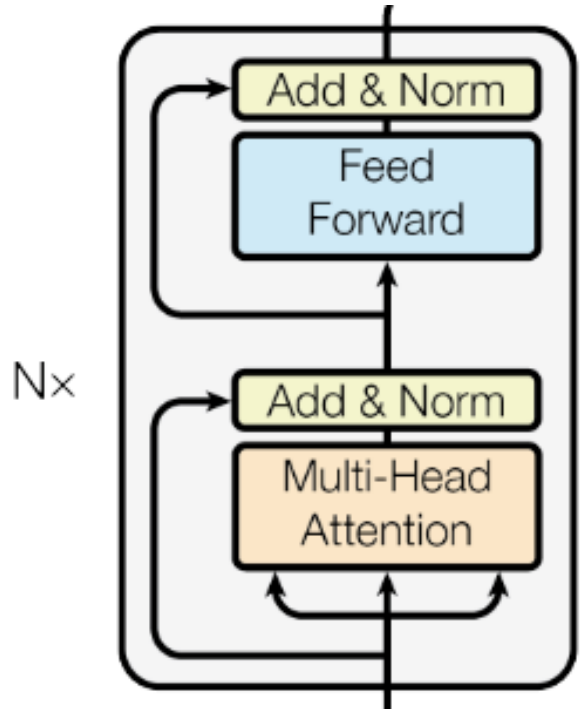
Transformer Encoder

Encoder

1. The transformer encoder applies one multi-head attention followed by one fully-connected feed-forward layer
2. Then pass to a layer normalization:

$$\textsc{LayerNorm}\big(x+\textsc{sublayer}(x)\big)$$

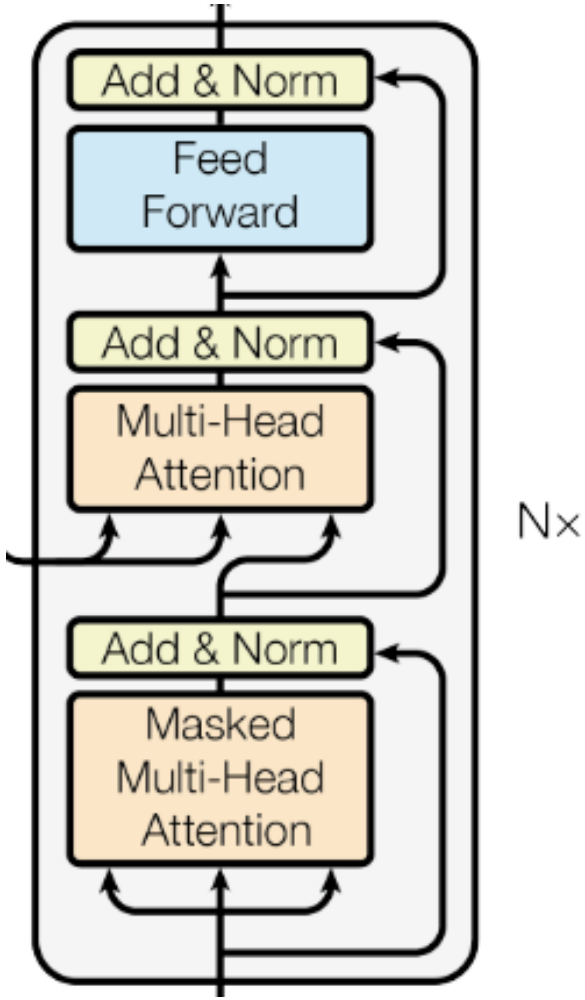
3. $N = 6$ stack transformer layers; the output dimension $d_{\text{model}} = 512$



Transformer Decoder

Decoder

1. Same as previous encoder, $N = 6$ identical stacked layers
2. DIFFERENCE: the 1st multi-head attention is *masked to prevent from attending to subsequent positions*, ensuring that the prediction output at position i only depends on the kown outputs at positions less than i, despite of the future.

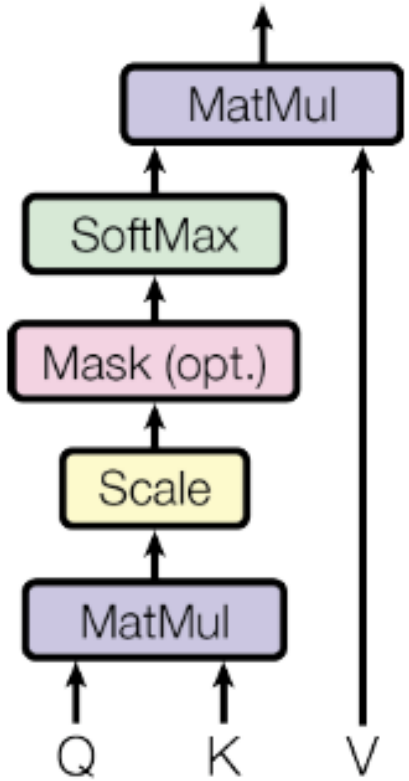


Scaled dot-product

- Consider the encoded representation of input sequences as a set of **key-value** pairs (K,V) with dimension of sequence length n.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V}$$

- **Benefits**: faster and space-efficient, in comparison with additive attention in practice.



Multi-head attention

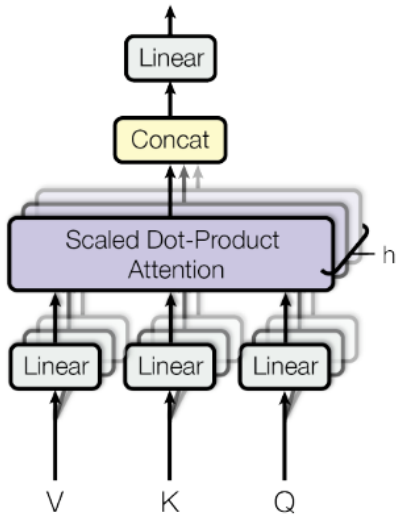
linear project the Q, K and V h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively.

Encoder

$K = V = Q$, i.e. the output of previous layer. Each position in the encoder can attend to all positions in the previous layer of the encoder.

Decoder

allow each position in the decoder to attend to all positions in the decoder up to and including that position.



Transformer

1. Point-wise feed-forward nets

$$\text{FFNN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

2. Positional encoding

$$PE_{(pos,2i)} = sine(\frac{pos}{10000^{2i/d_{model}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}})$$

Relational inductive biases

- Vinicius *et. al*(2019) augmented model-free DRL with a mechanism for relational reasoning over structured representations via **self-attention mechanisms**
- Incorporate the relational inductive biases for entity- and relation-centric state representation
- Based on distribute A2C algorithms
- Improved the performance, learning efficiency, generalization and interpretability.

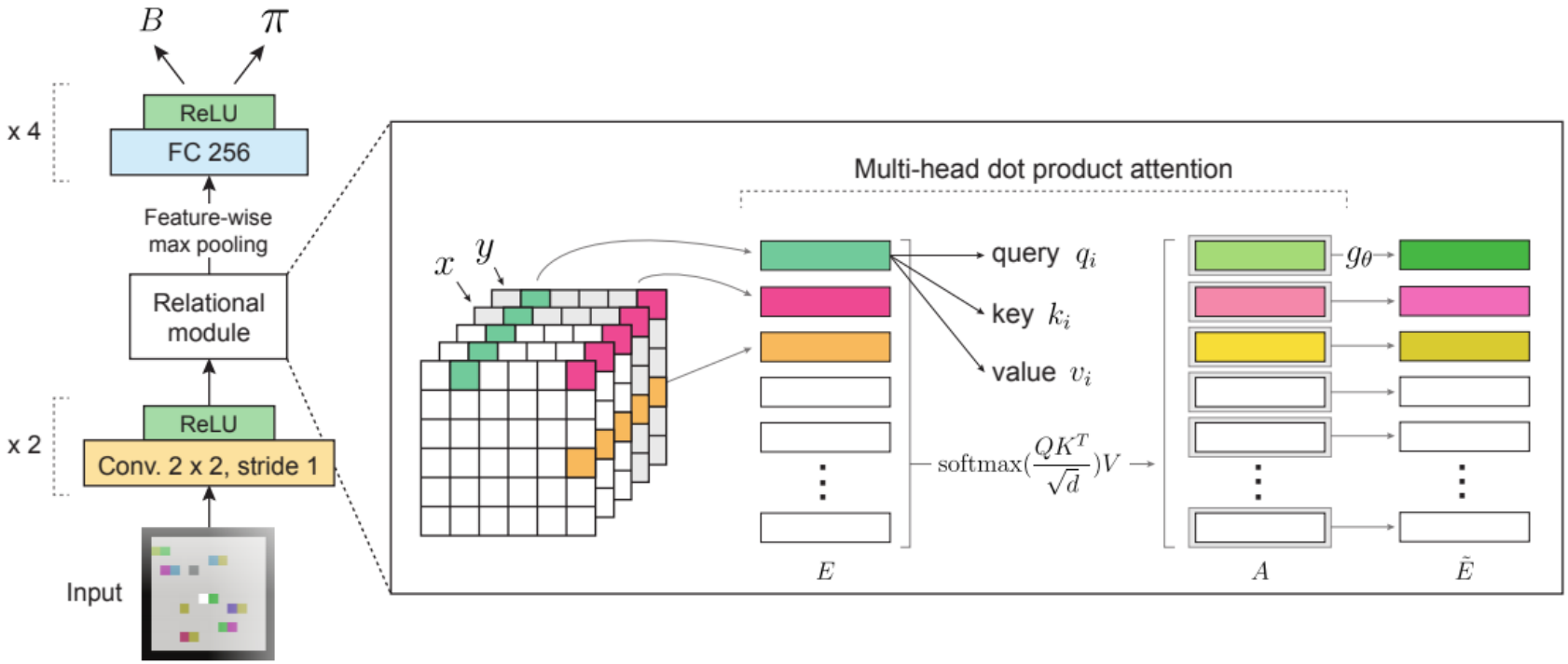
Relational inductive biases

- **Input**: receives raw visual input pixels with multi-layer ConvNets.
- **Output**: $(c+1)$-dimensional vector. The vector contains c-dimensional vector of π's logits where c is the number of discrete actions, plus a scalar baseline value estimate B.

Relational inductive biases

- *Feature-to-entity transformation*- **reshape** the feature map $S \in \mathbb{R}^{m \times n \times f}$ to entity vectors $E \in \mathbb{R}^{N \times f}$ (where $N = m \cdot n$)
- self-attention mechanism
- pass to an MLP with a residual connection

$$\tilde{\boldsymbol{e}}_i = g_\theta(\boldsymbol{a}_i^{h=1:H})$$



Auto-regressive policy head

Vinyals *et. al* (2017) represented the policy with the **auto-regressive** manner, i.e. predict each action conditioned on previous actions:

$$\pi_\theta(a|s) = \prod_{l=0}^{L} \pi_\theta(a^l|a^{<l}, s) \tag{1}$$

The auto-regressive policy transforms the problem of choosing a full action a to **a sequence of decisions** for each argument a^l.

Ptr Nets

Problems

Conventional sequence-to-sequence architecture can only applies softmax distribution over a fixed-sized output dictionary:

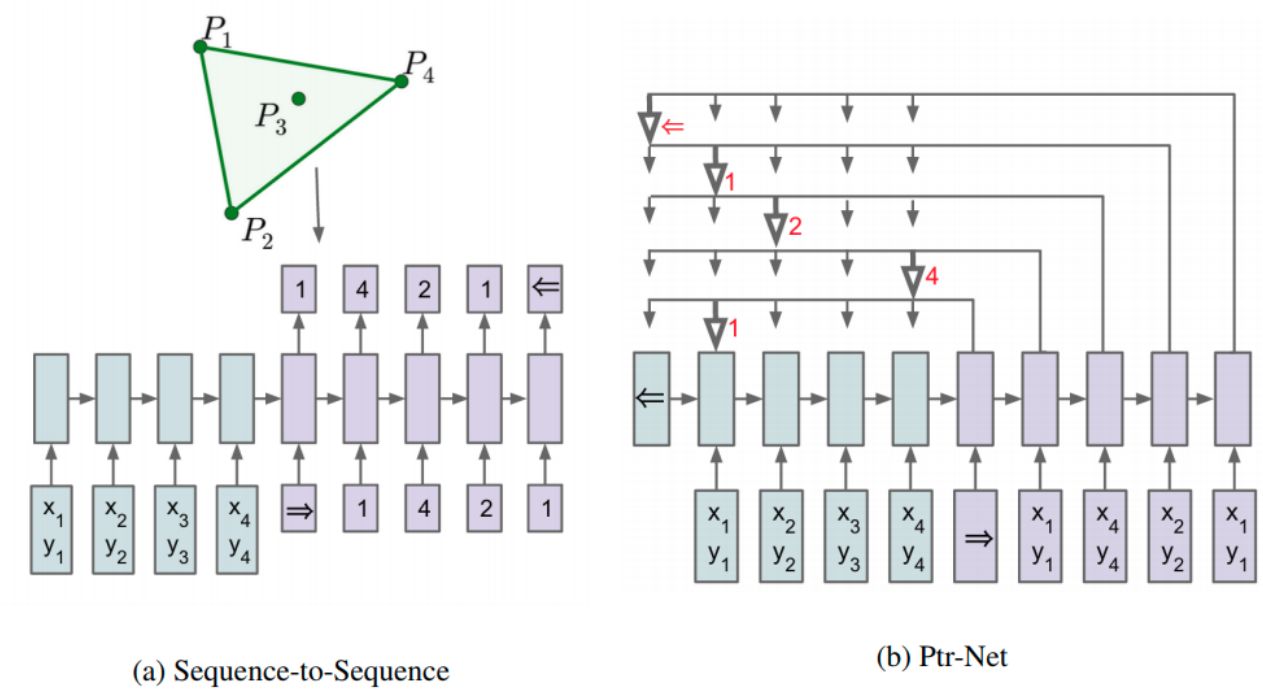
$$p(\mathcal{C}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p(C_i|C_1,\cdots,c_{i-1},\mathcal{P};\theta)$$

where $\mathcal{P} = \{P_1,\cdots,P_n\}$ is a sequence of n vectors and $\mathcal{C}^{\mathcal{P}} = \{C_1,\cdots,C_{m(\mathcal{P})}\}$ is a sequence of $m(\mathcal{P})$ indices.

Solution

Pointer Net is applied to handle this.

Ptr Nets



(a) Sequence-to-Sequence

(b) Ptr-Net

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i), \quad j \in (1, \cdots, n)$$

$$p(C_i | C_1, \cdots, C_{i-1}, \mathcal{P}) = \text{softmax}(u^i)$$

Ptr Nets

Guess

It can be inferred that the Pointer Net is used to *output the action for each unit*, since the StarCraft involves many units in concert and the number of units changes over time.

Centralized value baseline

Counterfactual multi-agent (COMA) policy gradients.

1. Centralized critic
2. Counterfactual baseline
3. Critic representation

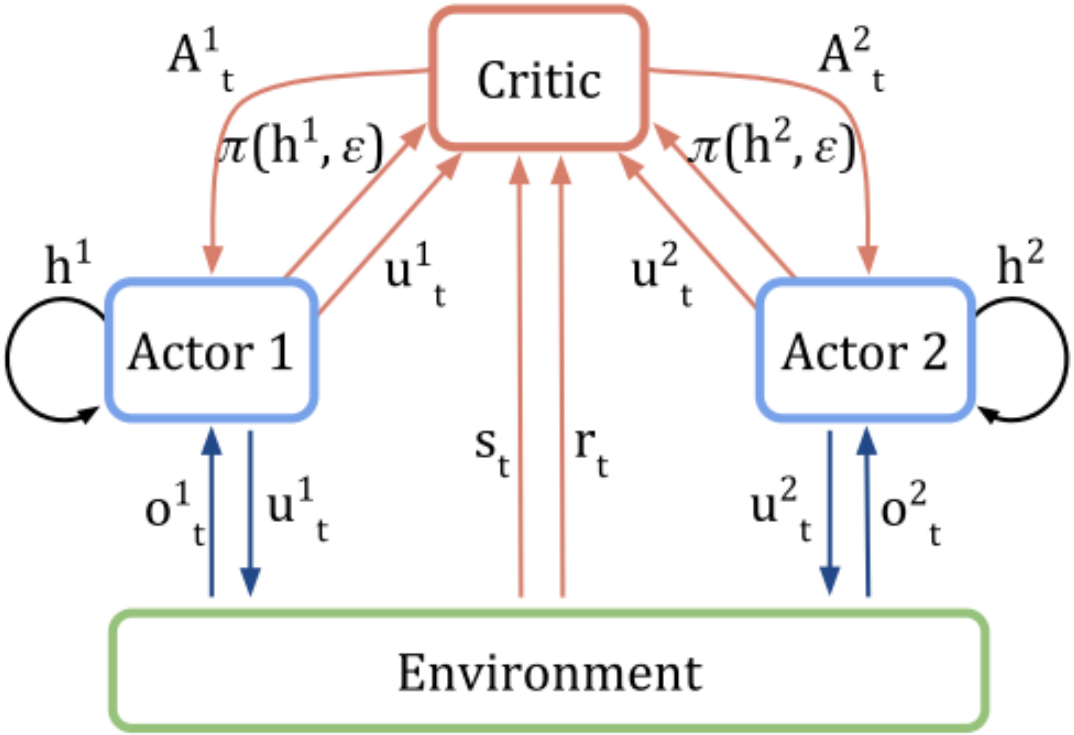
COMA - Centralized critic

Problems

Conventional *independent actor-critic* (IAC) independently trains each agent, but the **lack of information sharing** impedes to learn coordinated strategies that depend on *interactions between multiple agents*, or to *estimate the contribution of single agent's action* to the global rewards.

COMA - Centralized critic

- **Solution**: use a **centralized critic** that conditions on the true global state s, or the joint action-observation histories τ otherwise.
- The critic (red parts in the figure) is used only during *learning* and only the actor is needed during execution.



COMA - Counterfactual baseline

problems

A naive way is to follow the gradient based on the TD error:

$$g = \nabla_{\theta^\pi} \log \pi(\mu|\tau_t^a)(r + \gamma V(s_{t+1}) - V(s_t))$$

- This fails to address the key credit assignment problem
- The TD error only considers global rewards
- The gradient from each actor does not explicitly considered based on their respective contribution

COMA - Counterfactual baseline

difference reward

Compute the change of global reward when the action a of an individual agent is replaced by a default action c^a:

$$D^a = r(s, \boldsymbol{u}) - r(s, (\boldsymbol{u}^{-a}, c^a))$$

But this requires:

- the access to a simulator $r(s, (\boldsymbol{u}^{-a}, c^a))$
- a use-specific default action c^a.

COMA - Counterfactual baseline

Counterfactual baseline

Compute the agent a we can compute the advantage function that compares the Q-value for the current action μ^a to a counterfactual baseline that marginalize out μ^a, while keeping the other agents' actions μ^{-a} fixed:

$$A^a(s,\mu) = Q(s,\mu) - \sum_{\mu'^a} \pi^a(\mu'^a|\tau^a) Q(s,(\mu^{-a},\mu'^a))$$

where $A^a(s,\mu^a)$ measures the *difference* when only a's action changes, learn directly from agents' experiences rather than on extra simulations, a reward model or a use-designed default action

COMA - Critic representation

- The output dimension of networks would be equal to $|U|^n$, where n is the of agents.
- COMA uses critic representation in which it also takes the action of other agents u_t^{-a} as part of the input, and output a Q-value for each of agent a's action, with the of output nodes $|U|$

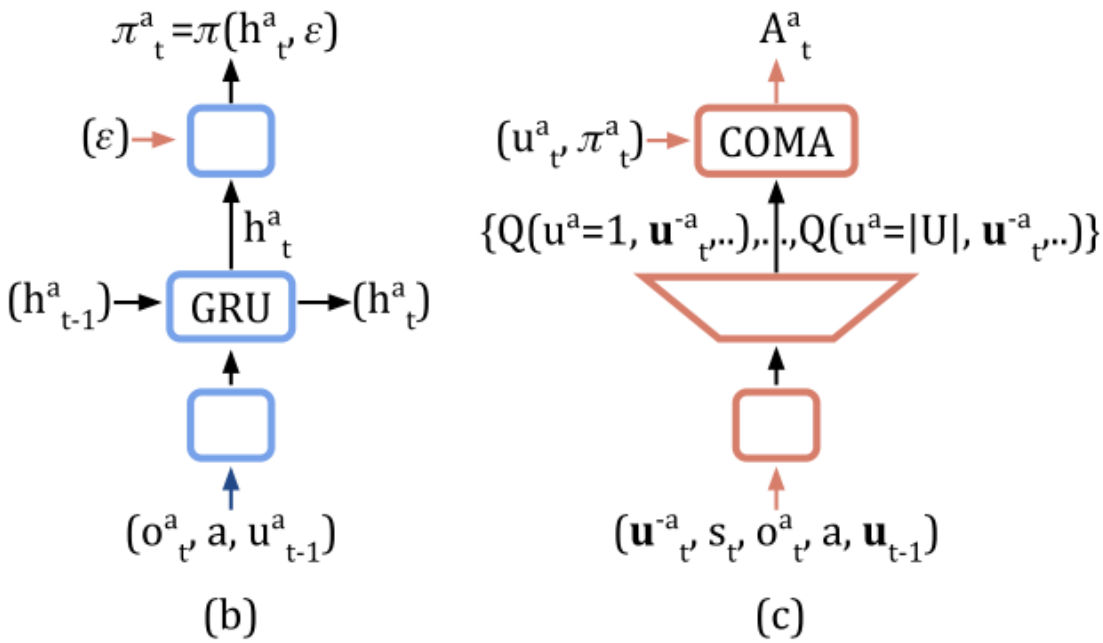


Fig. (b) and (c) are the architectures of the actor and critic.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
2. Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... Shanahan, M. (2018). Deep reinforcement learning with relational inductive biases. ICLR 2019
3. Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... Quan, J. (2017). Starcraft II: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782
4. Vinyals, O., Fortunato, M., Jaitly, N. (2015). Pointer networks. In Advances in Neural Information Processing Systems (pp. 2692-2700).
5. Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S. (2018, April). Counterfactual multi-agent policy gradients. In Thirty-Second AAAI Conference on Artificial Intelligence.

The End