Reinforcement learning using a stochastic gradient method with memory based learning
Outline

• Introduction
• Problem parameters
• RL in POMDPs with Stochastic Gradient Ascent
• SGA combined with Memory-based learning
• Evaluation for algorithms
• Conclusions
Motivation & Challenges in the real world

• Environment in the real world

• POMDPs is a suitable model for hidden state problems in the partially observable environment

• Function approximation could be a proposed solution for the large state-action space in the real world
Motivation & Challenges in the real world, Cont’d

- Design RL
  - Memory-based method needs sufficient memory for the size of the environment
  - Memory-less method acquires low average rewards in some certain environments

- Improve RL
  - How to suitably combine Memory-based method and Memory-less method to make use of the advantages in both methods
POMDPs environment for RL

- Transition Probability from state $s$ to $s'$

$$P^\pi(s, s') = \sum_{a \in \mathcal{A}} \sum_{X \in \mathcal{X}} P(X | s) \pi(a, W, X) P^a(s, s')$$

- Expected value of immediate received reward

$$R^\pi(s) = \sum_{a \in \mathcal{A}} \sum_{X \in \mathcal{X}} P(X | s) \pi(a, W, X) R^a(s)$$
Expected reinforcement performance criterion

- Expected value of average reward

\[ \Lambda = \lim_{N \to \infty} E \left\{ \frac{\sum_{t=0}^{N} r_t}{N + 1} \right\} \]

In the POMDPs environment, the optimal policy is defined as the one that maximizes the expected average reward.
A stochastic policy is used to explore the unknown environment to avoid sub-optimal result which is not the maximum value of average reward.
Function Approximation for Stochastic Policy, Cont'd

- The probability of selecting behavior $a$ when observing $X$ under policy $\pi$ is as follows:

$$Pr(a|\pi, X) = \pi(a, W, X)$$

Where $W$ is the internal variable of the policy.

- By modifying $W$, the agent will improve the policy $\pi$ closer to the optimal policy to maximize the average reward.
1. Observe $X_t$ in the environment.
2. Execute action $a_t$ with probability $\pi(a_t, W, X_t)$.
3. Receive the immediate reward $r_t$.
4. Calculate $e_i(t)$ and $D_i(t)$ as

\[
e_i(t) = \frac{\partial}{\partial w_i} \ln \left( \pi(a_t, W, X_t) \right),
\]
\[
D_i(t) = e_i(t) + \gamma D_i(t-1),
\]
where $\gamma (0 \leq \gamma < 1)$ denotes the discount factor, and $w_i$ does the $i^{th}$ component of $W$.
5. Calculate $\Delta w_i(t)$ as

\[
\Delta w_i(t) = (r_t - b) D_i(t),
\]
where $b$ denotes the reinforcement baseline.
6. Policy Improvement: update $W$ as

\[
\Delta W(t) = (\Delta w_1(t), \Delta w_2(t), \ldots, \Delta w_i(t), \ldots),
\]
\[
W \leftarrow W + \alpha(1 - \gamma) \Delta W(t),
\]
where $\alpha$ is a nonnegative learning rate factor.
7. Move to the time step $t + 1$, and go to step 1.
General Form of SGA algorithm, Cont'd

Update Weights
\[ \Delta W_i = (r - b) D_i \]
\[ W_i \leftarrow W_{i-1} + \alpha (1 - \gamma) \Delta W_i \]

Reward
\[ r \]

Sense
\[ o_i \]

Stochastic Policy
\[ P(a | o_i) \]

Action
\[ a \]

Accumulate Eligibility
\[ e_i = \nabla \ln P(a | o_i) \]
\[ D_i \leftarrow e_i + \gamma D_{i-1} \]
Theory for SGA algorithm

• Episodic REINFORCE algorithms

\[ \Delta w_{ij} = \alpha_{ij} (r - b_{ij}) \sum_{t=1}^{k} e_{ij}(t) \]

• Theorem for episodic REINFORCE algorithms

**Theorem 2.** For any episodic REINFORCE algorithm, the inner product of \( E\{\Delta W \mid W\} \) and \( \nabla_w E\{r \mid W\} \) is nonnegative. Furthermore, if \( \alpha_{ij} > 0 \) for all \( i \) and \( j \), then this inner product is zero only when \( \nabla_w E\{r \mid W\} = 0 \). Also, if \( \alpha_{ij} = \alpha \) is independent of \( i \) and \( j \), then \( E\{\Delta W \mid W\} = \alpha \nabla E\{r \mid W\} \)
Theory for SGA algorithm, Cont’d

• Theorem for Episodic SGA in POMDPs

\[
E \left\{ \sum_{t=0}^{N} (r_t - b)D_i(t) \right\} = \frac{\partial}{\partial w_i(t)} V_N^\pi(s_0)
\]

the agent can use SGA to change the policy in the direction for increasing discounted reward.
Disadvantages for SGA algorithm

• The best policy that appears in the previous trials may not be used in the next trial because of stochastic action selection.

• The route to get the target in some particular environment may not be stable, which would cause expected average reward is low.
Disadvantages for SGA algorithm, Cont'd

- The agent can move up, down, right and left.
- The agent can recognize the presence of a surrounding obstacle.
- S shows the start state, and G shows the goal state.
- When the agent reaches the goal, it obtains reward 100
- The same number of the cell is recognized as the same state by the agent.
Main idea of SGA with memory-based learning

- Firstly, SGA learning is applied to improve the policy. A set of observation-action sequences that accomplish the task is stored.

- Secondly, memory-based learning is applied by calling the actions obtained in the first stage to speed up the learning efficiency.
Difficulties of designing the proposed algorithm

• The observation and action cannot be placed as one-to-one correspondence in the second stage, when there are many incomplete perceptions in the environment.

• It is possible that a selected action in the second stage makes the agent get a new observation that is not stored, or an action stored in memory is lost.
Proposed algorithm

- Observed number $n_{X_t}$ of $X_t$ is introduced to create multiple-action correspondences for a incomplete observation to solve the first difficulty.
Proposed algorithm, Cont'd

• The second difficulty happens because SGA learning may be insufficient. Therefore, learning returns to the first stage until the SGA policy converges again.
Proposed algorithm, Cont'd
Experimental Environment—Maze World

The agent’s eight sensors indicate whether or not there is an obstacle in the surrounding eight areas

\[ X_i = \begin{cases} 0: \text{no} & \text{Xi}=0 \\ 1: \text{yes} & \text{Xi}=1 \end{cases} \quad (i=1,2,\ldots,8) \]
Implementation of SGA

\[ \pi (a, W, X) \]

Neural Network

Roulette Selection

Evaluation for algorithms
Implementation of SGA, Cont'd

- Neural Network

\[ f_j = \frac{1}{1 + \exp\left(-\sum_{k=1}^{8} x_k w_{kj}\right)} \]

- Roulette Selection

\[ \pi(a_j, W, X) = \frac{f_j}{\sum_{k=1}^{4} f_k} \]

- Characteristic Eligibility

\[ e_{ij}(t) = \frac{(\sum_{k=1}^{4} f_k) - f_j}{f_j (\sum_{k=1}^{4} f_k)} \frac{\partial}{\partial w_{ij}} f_j \]
Conclusions

• The proposed method can stabilize routes because route selection in the second stage is deterministic to use the approximate-optimal policy in the first stage to finish the task.

• Comparison experiments are performed with the conventional methods to prove the effectiveness of the proposed method.
Discussion

The SGA Learner actually uses input-output mapping:

1) a parameterized input-controlled distribution function
   (Neural Network) ---- Algorithm Core

2) outputs are randomly generated
   (Roulette Selection)

3) modify the learner's distribution function by performance feedback (Gradient of the policy)

Any other functions?
Reference


Thank you!