Mapping instructions and visual observations to actions with reinforcement learning

Dipendra Misra, John Langford, and Yoav Artzi
Task

Features:
- One single model
- Limited data
Task

• Training inputs:

\( \{(x^i, s^i_1, e^i)_{i=1}^N \} \)

• \( x^i \) is an instruction is a sequence \( < x_1, x_2, ... x_n > \) where \( x_i \) is a token
• \( s^i_1 \) is a start state
• \( e^i \) is an execution demonstration of \( x \) starting at \( s_1 \), which is an m-length sequence \( <(s_1, a_1), ..., (s_m, a_m)> \), where \( s_j \in S, a_j \in A \) and \( a_m = \text{STOP} \)
Task

• Testing inputs:

\[ \{(x^i, s_1^i, s_g^i)^M_i = 1 \} \]

• \(x^i\) is an instruction is a sequence \(<x_1, x_2, \ldots, x_n>\) where \(x_i\) is a token
• \(s_1^i\) is a start state
• \(s_g^i\) is a goal state
Architecture

\[
\tilde{s} = [v, x, \psi(a_{j-1})]
\]

State context
Actions

Action is decomposed into direction $a^D$ and block $a^B$. We compute the feed forward network:

$$
\begin{align*}
    h^1 &= \max(W^{(1)} \bar{s}_j + b^{(1)}), 0 \\
    h^D &= W^{(D)} h^1 + b^{(D)} \\
    h^B &= W^{(B)} h^1 + b^{(B)},
\end{align*}
$$

$$
\begin{align*}
    P(a^D_j = d \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(h^D_d) \\
    P(a^B_j = b \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(h^B_b).
\end{align*}
$$

$$
\begin{align*}
    P(a \mid x, s, a_{j-1}) = P(a^D_j = d \mid x, s, a_{j-1}) \cdot P(a^B_j = d \mid x, s, a_{j-1})
\end{align*}
$$
Reward function

The final shaped reward is the sum of reward shaping and the problem reward

\[ R^{(i)}(s, a) = \begin{cases} 
1.0 & \text{if } s = s_{m(i)} \text{ and } a = \text{STOP} \\
-1.0 & \text{if } s \neq s_{m(i)} \text{ and } a = \text{STOP} \\
-1.0 & \text{if } a \text{ fails to execute} \\
-\delta & \text{else} 
\end{cases} \]

where \( m^{(i)} \) is the length of \( e^{(i)} \).

Reward shaping:

- Distance-based shaping \( (F_1) \)
- Trajectory-based shaping \( (F_2) \)
Reward shaping

• Distance-based shaping (if the agent moved closer to the goal state)

\[ F_1^{(i)}(s_j, a_j, s_{j+1}) = \phi_1^{(i)}(s_{j+1}) - \phi_1^{(i)}(s_j) . \]

The potential \( \phi_1^{(i)} \) is proportional to the negative distance from the goal state:

\[ \phi_1^{(i)}(s) = -\eta \| s - s_g^{(i)} \| \]

• Trajectory-based shaping (considering the previous state and action)

\[ F_2^{(i)}(s_{j-1}, a_{j-1}, s_j, a_j) = \phi_2^{(i)}(s_j, a_j) - \phi_2^{(i)}(s_{j-1}, a_{j-1}) \]

Encourage the agent to take action close to the execution demonstration state
Policy gradient objective

• Contextual Bandit

  Suitable for the few-sample regime common in natural language problem

  Policy is learned from agent context rather than the world state

\[ \nabla_\theta J = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\nabla_\theta \log \pi(s, a) R^{(i)}(s, a)] \]
Policy gradient objective

- Contextual Bandit

\[
\nabla_\theta J = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\nabla_\theta \log \pi(s, a) R^{(i)}(s, a)]
\]

\[
\theta \leftarrow \theta + \mu \text{ADAM}(\frac{1}{j} \sum_{j'=1}^{j} \Delta_{j'})
\]

Input: a differentiable policy parameterization \( \pi(a|s, \theta), \forall a \in \mathcal{A}, s \in \mathcal{S}, \theta \in \mathbb{R}^n \)
Initialize policy weights \( \theta \)
Repeat forever:
Generate an episode \( S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T \), following \( \pi(\cdot|\cdot, \theta) \)
For each step of the episode \( t = 0, \ldots, T - 1 \):
\( G_t \leftarrow \text{return from step } t \)
\( \theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi(A_t|S_t, \theta) \)

Immediate reward
Average for this episode
Total reward for an episode
Policy gradient objective

- Entropy Penalty

To avoid falling into negative reward and rarely completing the task in the early training:

\[
\nabla_\theta J = - \frac{1}{N} \sum_{i=1}^{N} E[\nabla_\theta \log \pi(\tilde{s}, a) R^{(i)}(s, a) + \lambda \nabla_\theta H(\pi(\tilde{s}, \cdot))] \n\]
Algorithm 1 Policy gradient learning

**Input:** Training set \( \{(\bar{x}^{(i)}, s^{(i)}, e^{(i)})\}_{i=1}^{N} \), learning rate \( \mu \), epochs \( T \), horizon \( J \), and entropy regularization term \( \lambda \).

**Definitions:** \( \text{IMG}(s) \) is a camera sensor that reports an RGB image of state \( s \). \( \pi \) is a probabilistic neural network policy parameterized by \( \theta \), as described in Section 4. \( \text{EXECUTE}(s, a) \) executes the action \( a \) at the state \( s \), and returns the new state. \( R^{(i)} \) is the reward function for example \( i \). \( \text{ADAM}(\Delta) \) applies a per-feature learning rate to the gradient \( \Delta \) (Kingma and Ba, 2014).

**Output:** Policy parameters \( \theta \).

1: \( \Rightarrow \) Iterate over the training data.
2: \( \textbf{for} \ t = 1 \text{ to } T, \ i = 1 \text{ to } N \textbf{ do} \)
3: \( I_{1-K}, \ldots, I_{0} = \emptyset \)
4: \( a_{0} = \text{NONE}, s_{1} = s_{1}^{(i)} \)
5: \( j = 1 \)
6: \( \Rightarrow \) Rollout up to episode limit.
7: \( \textbf{while} \ j \leq J \text{ and } a_{j} \neq \text{STOP} \textbf{ do} \)
8: \( \Rightarrow \) Observe world and construct agent context.
9: \( I_{j} = \text{IMG}(s_{j}) \)
10: \( s_{j} = (\bar{x}^{(i)}, I_{j}, I_{j-1}, \ldots, I_{1-K}, a_{j}^{d-1}) \)
11: \( \Rightarrow \) Sample an action from the policy.
12: \( a_{j} \sim \pi(\tilde{s}_{j}, a) \)
13: \( s_{j+1} = \text{EXECUTE}(s_{j}, a_{j}) \)
14: \( \Rightarrow \) Compute the approximate gradient.
15: \( \Delta_{j} \leftarrow \nabla_{\theta} \log \pi(\tilde{s}_{j}, a_{j}) R^{(i)}(s_{j}, a_{j}) + \lambda \nabla_{\theta} H(\pi(s_{j}, \cdot)) \)
16: \( j+ = 1 \)
17: \( \theta \leftarrow \theta + \mu \text{ADAM}(\frac{1}{J} \sum_{j=1}^{J} \Delta_{j}) \)
18: \( \text{return} \ \theta \)
Results

**Distance error:**
The sum of Euclidean distances for each block between its position at the end of the execution and in the gold goal state.

F1 is better than F2.

Reflects problem with limited data.