Reinforcement Learning with Deep Structured Semantic Model

Xugang Ye

To consider how to combine the deep structured semantic model (DSSM) with the reinforcement learning (RL), let’s look at the fundamental components of the RL.

Consider the Markov process shown as the Figure 1. At time $t$, let $s_t$ be the state, $a_t$ be an action and $g_t$ be the long-term gain. Note that $g_t$ is an additive function, that is $g_t = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k})$, where $r(s, a)$ is the immediate reward of taking action $a$ at state $s$ and $\gamma$ is the discount factor.

![Figure 1: The simple Markov process of (state, action, reward).](image)

We consider the expected long-term gain at state $s_t = s$ and calculate it via conditioning:

$$E(g_t|s_t = s) = \sum_a E(g_t|s_t = s, a_t = a)P(a_t = a|s_t = s)$$

$$= \sum_a E(r(s_t, a_t) + \gamma g_{t+1}|s_t = s, a_t = a)P(a_t = a|s_t = s)$$

$$= \sum_a \left(r(s, a) + \gamma E(g_{t+1}|s_t = s, a_t = a)\right)\pi(a|s).$$  

(1)

Here $\pi(a|s) = P(a_t = a|s_t = s)$ is the policy of choosing action $a$ given state $s$. $\gamma \in (0,1)$ is the discount factor.

Let’s consider $E(g_{t+1}|s_t = s, a_t = a)$. We can calculate it via conditioning on $s_{t+1}$, that is

$$E(g_{t+1}|s_t = s, a_t = a) = \sum_{s'} E(g_{t+1}|s_{t+1} = s', s_t = s, a_t = a)P(s_{t+1} = s'|s_t = s, a_t = a)$$

$$= \sum_{s'} E(g_{t+1}|s_{t+1} = s')T(s'|s, a).$$  

(2)

Here $T(s'|s, a) = P(s_{t+1} = s'|s_t = s, a_t = a)$ is the transition mechanism of reaching state $s'$ given action $a$ taken at state $s$.

Plugging (2) into (1) yields

$$E(g_t|s_t = s) = \sum_a \left(r(s, a) + \gamma \sum_{s'} E(g_{t+1}|s_{t+1} = s')T(s'|s, a)\right)\pi(a|s).$$  

(3)

which tells that maximizing $E(g_t|s_t = s)$ is reduced to branches of maximizing $E(g_{t+1}|s_{t+1} = s')$. This falls into the dynamic programming framework.
Let $E^*(g_t|s_t = s), E^*(g_{t+1}|s_{t+1} = s')$ denote the optimal values for time $t, t + 1$, respectively, then we have

\[
E^*(g_t|s_t = s) = \sum_a (r(s, a) + \gamma \sum_{s'} E^*(g_{t+1}|s_{t+1} = s') T(s'|s, a)) \pi^*(a|s),
\]

(4)

where $\pi^*(a|s)$ denote the optimal policy of choosing action $a$ given state $s$.

Let $a_s$ be the best action for state $s$ and assume $\pi^*(a|s) = \text{I}_{a = a_s}$, then

\[
E^*(g_t|s_t = s) = r(s, a_s) + \gamma \sum_{s'} E^*(g_{t+1}|s_{t+1} = s') T(s'|s, a_s),
\]

(5)

which is the Bellman Equation.

Note that $E^*(g_t|s_t = s)$ is actually a function of $s$ and $a_s$, we can simplify the notation and denote it as $Q(s, a_s)$, then

\[
Q(s, a_s) = r(s, a_s) + \gamma \sum_{s'} Q(s', a_s) T(s'|s, a_s).
\]

(6)

Assume that the immediate reward is observable, then there are two basic functions $Q(s, a_s)$ and $T(s'|s, a_s)$. $T(s'|s, a_s)$ can be learned from observing the environment. That is, as long as there are enough $(s, a_s, s')$-triplets collected, a model can be built for $T(s'|s, a_s)$. Given a $T(s'|s, a_s)$, $Q(s, a_s)$ can be learned from value iterations. The key idea arises from this point.

We can use two deep semantic model: one for $T(s'|s, a_s)$, the other for $Q(s, a_s)$. The model for $T(s'|s, a_s)$ is to predict, or anticipate. We can view the pair $(s, a_s)$ as query and $s'$ as document. The model for $Q(s, a_s)$ is to act. Note that both the state space and the action space are infinite, it's worthwhile to have the both models as ranking models.

The model architecture for $T(s'|s, a_s)$ can be illustrated by the following Figure 2.

![Figure 2: The architecture of the latent semantic model for transition mechanism.](image-url)
The model architecture for $Q(s, a)$ can be illustrated by the following Figure 3.

![Figure 3: The architecture of the latent semantic model for action selection.](image)

We know how to train a model for $T(s'|s, a_s)$. In order to train a model for $Q(s, a_s)$, we need to embed the learning of semantic mapping structure into a value iteration procedure. Suppose the parameters of $Q$ is denoted as $\theta$, we can design the following composite learning procedure:

1) Initialize $\theta$ as $\theta^{(0)}$, where the part for $q = s$ is as the same as the part for $d = s'$ in the model for $T(s'|s, a_s)$;
2) Given $\theta^{(k)}$,
   2.1) for each $s$ in the training set, calculate $Q(s, a)$ for a collection of $a$'s and pick up $a_s$ that has the best $Q$-value;
   2.2) for each $(s, a_s)$-pair, simulate a collection of $s'$ by $T(s'|s, a_s)$ and calculate $y(s, a_s) = r(s, a_s) + \gamma \sum_{s'} Q(s', a_{s'})T(s'|s, a_s)$; if $\sum_s(y(s, a_s) - Q(s, a_s))^2$ is small enough, then stop; otherwise, go to 2.3)
   2.3) minimize $\sum_s(y(s, a_s) - Q(s, a_s))^2$ with respect to the part of $\theta^{(k)}$ for $d = a_s$ and update $\theta^{(k)}$ into $\theta^{(k+1)}$.

This learning procedure actually incorporates the long-term reward information and also leverage the experience acquired by the model for $T(s'|s, a_s)$.

Note that this learning procedure is more like exploitation. Let’s consider the exploration. Exploration means an action $a_s$ is generated from a state $s$ by $Q(s, a_s)$, and then wait for the environment to transit to a state $s'$. Once $s'$ is observed, an action $a_{s'}$ is generated again by the Q-function. When doing this, we can collect a new set of $(s, a_s, s')$-triplets to tune the parameters of $T(s'|s, a_s)$. Once we have a new model for $T(s'|s, a_s)$, we can re-train the model for $Q(s, a_s)$ by using the above learning procedure.

Regarding the applications, we can think about the machine-human dialogue. We can treat $Q(s, a)$ as a value function, or ranker, of the machine. It lets the machine to choose the best sentence as it can. We can also treat $T(s'|s, a_s)$ as the machine’s predictor of human’s
response. The remaining question is how to label the episodes. Probably a separate sentiment model is needed to evaluate the sentiment score of each human response.