Types of Reinforcement learning algorithm – adaptive ϵ methods

Ameet Rahane

April 11, 2019

1 Preface

All of the following methods are ways I'm considering changing explore/exploit behavior within models in order to fit it to a biological agent. Something of notes is that most of these models are built to optimize for reward gained. This is not necessarily our goal. We want to find the best fit to a mouse.

2 ϵ Greedy

With this method, the amount of exploration is globally controlled by a parameter $\epsilon \in [0, 1]$ that denotes the probability of action selection. You choose actions based on the following equation:

$$a_t = \begin{cases} a_t^* & p = 1 - \epsilon \\ \text{random} & p = \epsilon \end{cases} \tag{1}$$

This basically stochastically chooses between two choices – pure randomness or pure maximization (this is probably a bit idealistic and won't fit mice that well).

3 Softmax

- Bias exploration towards promising actions
- Softmax action selection methods grade action probabilities by estimated values
- Most Common is Boltzmann Distribution:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{e^{\sum_{a' \in A} \frac{Q(s,a')}{\tau}}} \tag{2}$$

- Here τ is a temperature constant. That is $\tau \to \infty$ implies that $P = \frac{1}{|A|}$ and $T \to 0$ implies that it's greedy

4 Value Difference Based Exploration

This method extends $\epsilon-$greedy by adapting a state dependent exploration probability $\epsilon(s)$ instead of the classical hand tuning. The key idea is to consider the TD-error observed from value-function backups as a measure of the agent's uncertainty about the environment which directly affects the exploration probability.

Consider the RL framework where an agent interacts with a Markovian decision process. At each time step $t \in \mathbb{N}$, the agent is in a sertain state $s_t \in S$. After the selection of the action,$a_t \in A(s_t)$, the agent

receives a reward signal from the environment $r_{t+1} \in \mathbb{R}$ and is passed into a successor state. The decision which action a is chosen in a certain state is characterized by a policy $\pi(s) = a$, which could also be stochastic $\pi(a|s) = p(a_t = a|s_t = s$. A policy that maximizes the cumulative reward is denoted as π^*.

A state action value denotes the expected cumulative reward R_t for following π by starting in state s and selection action a:

$$Q^\pi(s,a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\} \tag{3}$$

γ is a discount factor in $(0, 1]$ for episodic learning tasks and $0 < \gamma < 1$ for continuous learning tasks.

The basic idea of VDBE is to extend ϵ greedy by controlling a state dependent explorations probability $\epsilon(s)$ in dependence of the value function error instead of manual tuning. The following equatins adapt such desired behavior according to a softmax Boltzmann distribution of the value fuction estimates. This is performed after eadch learning step by:

$$f(s,a,\sigma) = \frac{1 - e^{\frac{-|a\text{TD-Error}|}{\sigma}}}{1 + e^{\frac{-|a\text{TD-Error}|}{\sigma}}} \tag{4}$$

Then, we update ϵ as follows:

$$\epsilon_{t+1}(s) = \delta f(s_t, a_t, \sigma) + (1 - \delta)\epsilon_t(s) \tag{5}$$

σ is a positive constant called inverse sensitivity and $\delta \in [0, 1)$, a parameter determining the influence of the selection action on the exploration rate. An obvious setting for δ may be the inverse of the number of actions in the current state: $\delta = \frac{1}{|\mathcal{A}(s)|}$. Low inverse sensitivities cause full exploration even at small value changes. On the other hand, shigh inverse sensitivities cause a high level of exploration only at large value changes. In the limit, the exploration rate converges to zero as the Q-function converges, which results in pure greedy action selection. One drawback is the exploration actions are chosen uniformly distributed among all possible actions in the current state.

5 Adaptive control between ϵ Greedy and Softmax

5.1 Learning the Q function by on and off policy methods

Value functions are learned by sampling observations of the interactions between the agent and its environment. For this, the branch of temporal difference learing offers two commonly used alogrithm which are named Sarsa for on-policy control:

$$\Delta_{\text{Sarsa}} = [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{6}$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Delta_{\text{Sarsa}} \tag{7}$$

and Q learning for off policy control:

$$b^* = \text{argmax}_{b \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, b) \tag{8}$$

$$\Delta_{\text{Qlearning}} = [r_{t+1} + \gamma Q(s_{t+1}, b^*) - Q(s_t, a_t)] \tag{9}$$

$$Q(s_t, a_t = Q(s_t, a_t) + \alpha\Delta_{\text{Qlearning}} \tag{10}$$

α is a stepsize parameter. The only difference in two algorithms is the inclusion of successor state information used for the evaluation of action a_t taken in state s_t while learning the value function.

5.2 VDBE-Softmax

One way we can fix the issue with VDBE is to do a softmax step instead of sampling uniformly for the explore. This is as follows:

$$\pi(s) = \begin{cases} \text{Softmax: } \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,b)/\tau}} & \zeta < \epsilon \\ \text{argmax}_{a \in \mathcal{A}(s)} Q(s,a) & \zeta \geq \epsilon \end{cases} \tag{11}$$

This adapts the state dependent exploration rate according to VDBE but selects random actions according to Softmax in the case of exploration.

6 Creating a probability distribution over the possible q value and sampling that

7 Apply the Softmax at each step

8 Binary stochastic switching between greedy behavior and probabilistic sampling

9 Binary stochastic switching between greedy behavior and softmax

10 Adaptive Exploration using stochastic neurons

THIS NAME IS A MISNOMER – I STRONGLY DISAGREE WITH THIS

Finding a near optimal exploration parameter by trail and error can be a very time consuming task and it si desired to have algorithms adapt this parameters based on current operating conditions. The proposed idea is adapting the parameter a_e of an action selection policy $\pi(a_e, \cdots)$ towards improving the future outcome of π with regard to a performance meausre ρ. For maximizing ρ in the future, we use Williams' REINFORCE with multiparameter distributions algorithm using a stochastic neuron model. The input to such neuron is a weighted parameter vector θ from which the neuron determines an adaptable stochastic scalar as its output. In example, if the policy is an epsilon greedy strategy, then a_e is ϵ.

If we assume one starting state, the exploration parameter a_e is drawn at the beginning of an episode from a Gaussian, that is: $a_e \sim \mathcal{N}(\mu, \sigma)$ with the following density function:

$$g(a_e, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(a_e-\mu)^2/2\sigma^2} \tag{12}$$

Let θ denote the vector of adaptable parameters consisting of

$$\theta = \begin{pmatrix} \mu \\ \sigma \end{pmatrix} \tag{13}$$

At the end of episode i, the components of θ are adapted towards the gradient with regard to the outcome ρ of the current episode

$$\theta_{i+1} \approx \theta_i + \alpha \Delta_\theta \rho \tag{14}$$

For improving the future performance of $\pi(a_e, \cdots)$, the policies outcome is measured as the cumulative rewaqrd in the current episode.

$$\rho = E\{r_1 + r_2 + \cdots r_T | \pi(a_e, \cdot, \cdot)\} \tag{15}$$

The characteristic eligibility of each component of θ is estimated by

$$\frac{\partial \ln g(a_e, \mu, \sigma)}{\partial \mu} = \frac{a_e - \mu}{\sigma^2} \tag{16}$$

$$\frac{\partial \ln g(a_e, \mu, \sigma)}{\partial \sigma} = \frac{(a_e - \mu)^2 - \sigma^2}{\sigma^3} \tag{17}$$

A reasonable algorithm for adapting μ and σ has the following form:

$$\Delta\mu = \alpha_R(\rho - \overline{\rho})\frac{a_e - \mu}{\sigma^2} \tag{18}$$

$$\Delta\sigma = \alpha_R(\rho - \overline{\rho})\frac{(a_e - \mu)^2 - \sigma^2}{\sigma^3} \tag{19}$$

The learnign rate has to be chosen appropriately, as a small positive constant: $\alpha_t R = \alpha\sigma^2$. The baseline $\overline{\rho}$ is adapted by a simple reinforcemnt comparison scheme: $\overline{\rho} = \overline{\rho} + \alpha(\rho - \overline{\rho})$. The standard deviation effecrtively constrols exploration in teh space of α_e. Importantly, a proper functioning of the proposed algorithm depends on some requirements. In order to limit the search of reasonable parameter, the exploration parameter, mean and standard deviation must be bounded for obtaining reasonable performance. Furthermore, if the learning problem consists of mroe than one starting state all parameters must be associated to each occuring starting state.

Look at meta-learning of exploration and exploitation parameters with replacing eligibility traces by Tokic and look at page 4 for the algorithm.

11 Meta learning of exporation and exploitation parameters

A drawback of epsilon greedy and softmax is the optimism in the face of uncertainty. For this, it was proposed VDBE with Sofmax, which controls exploration and exploitation in a meta learning fashion. Look above. The general idea of this is that high fluctuations of the action value function should lead to a high degree of exploration because the observation is insufficiently approximated by the prediction. Ont he other hand, when the prediction about future reward siwell approximated, so far learned knowledge should be exploited. In this sense, the corresponding exploration rate in state s is updated after each value function backup. I'm pretty sure this paper is just a survey of different methods, with details and applications to a robot – useful but not a new algorithm

12 Gradient Algorithms for Exploration/Exploitation Trade-offs

12.1 MBE

THis should have been much earlier in this thing but whatever. This is a combination wof Softmax with ϵ greedy into the max boltzmann exploration rule, which selects explroation actions according to softmax instead of being equally distributed.

12.2 Global Episodic Control

The REC algorithm determines at each time step a continuous valued action from a multiparameter distribution, representing the exploration parameter a_e. For this purpose, we use a normal distribution with parameter μ and σ which are given to the neuron as inputs. At the beginning of eacvh learing episode, the exploration parameter, beign valid over the whole episode is determined from the distribution: $a_e \sim \mathcal{N}(\mu, \sigma)$. The rest of this is just recap from the last one, so I'm skipping it.

12.3 Local Step-Wise Control

The results from the previous section can be extended for obtaining a local variant, aiming at producing exploratory behavior only in states with improvement potential. In general, all utilized parameters become local parameters associated to each state. The mean and standard deviation are readapted after evaluation action a performed in state s by Q learning or Sarsa. In prior to an action selection in state s, an expoloration parameter a_e is determined based on $\mu(s)$ and $\sigma(s)$. As before, $a_e \sim \mathcal{N}(\mu(s), \sigma(s))$. For evaluating the utility of a_e, the estimated utility of the actual taken actions $Q_{t+1}(s_t, a_t)$ is considered. Let $\rho = Q_{t+1}(s_t, a_t)$. THe following equaiton now readapt the distribution parameters from state s_t based on the policies outcome using its current parameter a_e.

$$\Delta\mu(s_t) = \alpha_R(\rho - \overline{\rho}(s_t))\frac{a_e - \mu(s_t)}{\sigma(s_t)^2} \tag{20}$$

$$\Delta\sigma(s_t) = \alpha_R(\rho - \overline{\rho}(s_t))\frac{(a_e - \mu(s_t))^2 - (s_t)^2}{\sigma(s_t)^3} \tag{21}$$

The baseline $\overline{\rho}$ is adapted in the same fashion as above:

$$\overline{\rho} = \overline{\rho}(s_t) + \alpha(\rho - \overline{\rho}(s_t)) \tag{22}$$

These presented policies are evaluated using off-policy Q-learning and on-policy Sarsa learning.

13 Chunk based exploration/exploitation???