Page 4

As a quick refresher, Actor-Critics idea is to estimate the gradient of our policy as a $\nabla J = \nabla_\theta \log \pi_\theta(a|s)(R - V_\theta(s))$. The policy π_θ is supposed to provide us the probability distribution of actions given the observed state. The quantity $V_\theta(s)$ called a critic and equals to the value of the state and is trained using the mean square loss between the critic return and the value estimated by the Bellman equation. To improve exploration, the entropy bonus $L_H = \pi_\theta(s) \log \pi_\theta(s)$ is usually added to the loss.

For N actions, it will be two vectors of size N, the first is the mean values μ and the second vector will contain variances σ^2.

By definition, the probability density function of the Gaussian distribution is $f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. We could directly use this formula to get the probabilities, but to improve numerical stability it worth to do some math and simplify the expression for $\log \pi_\theta(a|s)$. The final result will be: $\log \pi_\theta(a|s) = -\frac{(x-\mu)^2}{2\sigma^2} - \log \sqrt{2\pi\sigma^2}$. The entropy of the Gaussian distribution could be obtained using the differential entropy definition and will be $\ln \sqrt{2\pi e\sigma^2}$. Now we have everything we need to implement the Actor-Critic method. Lets do it.

Page 9

We have two functions, one is actor, lets call it $\mu(s)$which converts the state into the action and another is critic, by the state and the action giving us the Q-value: $Q(s, a)$. We can substitute the actor function into the critic and get the expression with only one input parameter of our state: $Q(s, \mu(s))$. At the end, neural networks are just functions.

Now, the next step. The output of the critic gives us the approximation of the thing were interested to maximize in the first place – the discounted total reward. This value depends not only on the input state, but also on parameters of the actor θ_μ and the critic networks θ_Q. At every step of our optimisation we want to change the actors weights to improve the total reward we want to get. In mathematical terms, we want the gradient of our policy.

In his Deterministic Policy Gradient theorem, David Silver has proved that stochastic policy gradient is equivalent to the deterministic policy gradient, in other words, to improve the policy, we just need to calculate the gradient of the function $Q(s, \mu(s))$. By applying the chain rule, we get the gradient: $\nabla_a Q(s, a) \nabla_{\theta_\mu} \mu(s)$.

Page 10

The price we have to pay for all this goodness is that our policy is now deterministic, so, we have to explore the environment somehow. The answer will be to add the noise to the actions returned by the actor before we pass them to the environment. There are several options here. The simplest method is just to add the random noise to the actions $\mu(s) + \epsilon\mathcal{N}$. Well use this way of exploration in the next method well consider in the chapter. More fancy approach to the exploration will be to use the stochastical model, very popular in financial world and other domains dealing with stochastic processes: Ornstein-Uhlenbeck processes.

This process models the velocity of the massive Brownian particle under the influence of the friction and is defined by this stochastic differential equation: $\partial x_t = \theta(\mu - x_t)\partial t + \sigma \partial W$, where θ, μ, σ are parameters of the process and W_t is the Wiener process. In discrete-time case, the Ornstein-Uhlenbeck process could be written as $x_{t+1} = x_t + \theta(\mu - x) + \sigma \mathcal{N}$. In our exploration, well add the value of the Ornstein-Uhlenbeck process to the action returned by the actor.

Page 19

The Bellman operator has a form of $Z(x, a) \overset{D}{=} R(x, a) + \gamma Z(x', a')$, and supposed to transform the probability distribution as shown on the image