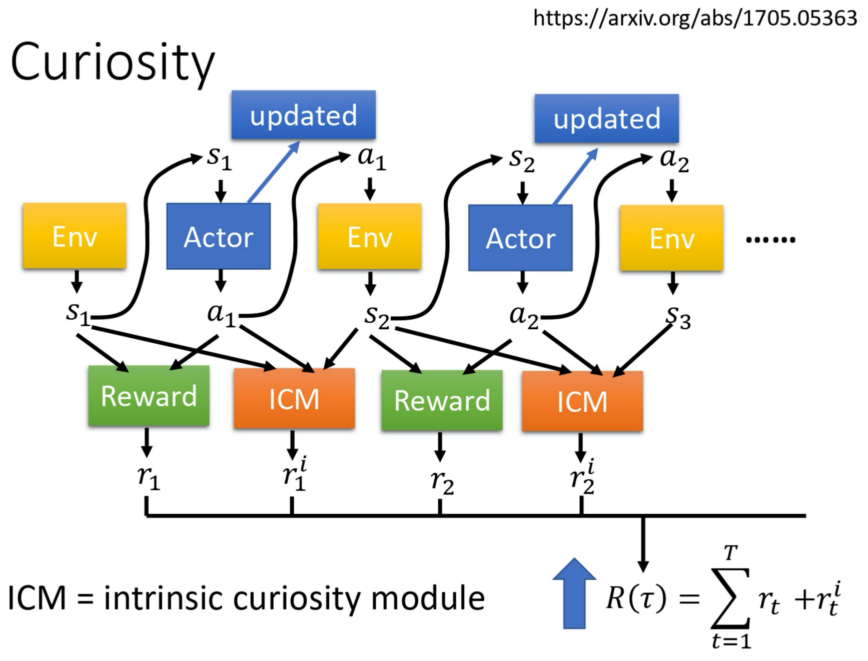
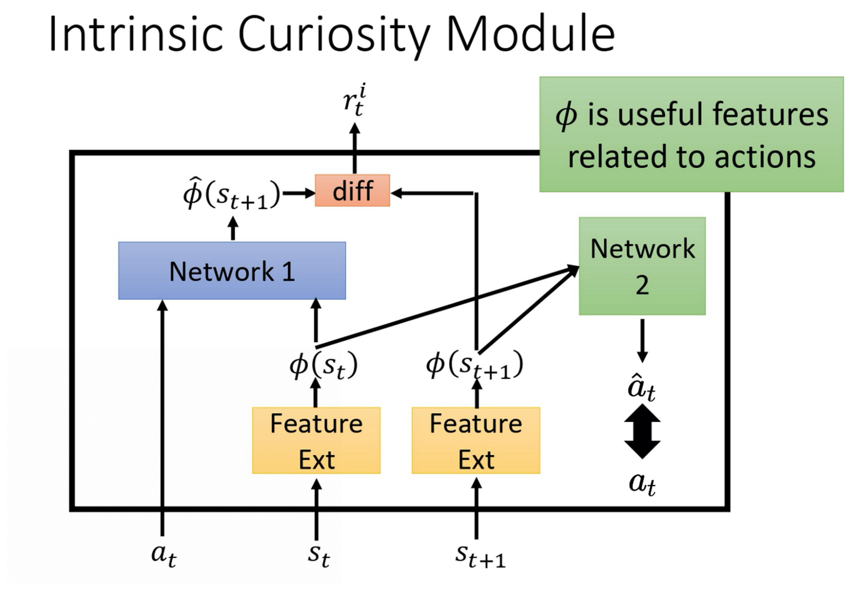
Curiosity

2021年2月18日 10:01

- Curiosity driven reward


https://arxiv.org/abs/1705.05363
Curiosity
updated
updated
Env
Actor
Env
Actor
Env
......
Reward
ICM
Reward
ICM
ICM = intrinsic curiosity module
$R(\tau)=\sum_{t=1}^{T} r_t + r_t^i$

- ICM



In curiosity driven, we will add a new reward function, which is ICM(intrinsic curiosity module). The Input of ICM are state S1, action A1 and state S2. According to the S1, A1, S2, the machine will generate extra output r_1^i. The total reward equals r plus r_1^i. Therefore, when it interacts with the environment, it does not only hope that r is better, it also hopes that r_1^i is better.

There is a network in ICM, which will take S_t *and* A_t and generate the output$\widehat{\emptyset}(S_{t+1})$. In other words,this is the network according to S_t *and* A_t *, to* predict $\widehat{\emptyset}(S_{t+1})$. Next, let's compare similarity of the prediction reslut $\widehat{\emptyset}(S_{t+1})$ and real situation $\emptyset(S_{t+1})$. The less similarity, the bigger reward will be. So this reward r_t^i means that **if the future state is harder to predict, then the reward will be greater**. **This is to encourage the machine to take risks**. **Taking this action now, the less predictable what will happen in the future, the greater the reward of this action**. So if there is an ICM , the machine will tend to take some riskier actions. It wants to explore the unknown world. It wants to see and say, assuming that a certain state cannot be predicted, it will Especially want to adopt that state, this can increase the ability of machine exploration.

The problem in this method is that some state is hard to predict doesn't mean it's good.For example, the outcome of Russian roulette is impossible to predict, which does not mean that agent should play it all the time. So it's not enough just to be curious, It's also to know what's really important.

How to let Machine know what things are really important? we have to add another module, we want to learn a feature extractor, yellow grid on behalf of the feature extractor, it is input a state, and then output a feature vector to represent the state, then we expect this feature extractor can filter out meaningless things, For example, the movement of the wind and grass, the movement of the clouds, and the movement of the leaves.

Assuming that the feature extractor can really filter out the irrelevant stuff, what network 1 actually does is give it an actor and give it feature representation of a state S_t, and let it predict state S_{t+1}. Then compare the prediction result with real state S_{t+1}. The less the representation is, the bigger the reward is. How to learn this feature extractor? Let the feature extractor can filter out unimportant things? The learn method over here is to learn another network 2. This network 2 takes $\widehat{\emptyset}(S_{t+1})$ *and* $\widehat{\emptyset}(S_t)$ as input, then, it predicts what action a is and expects the action a to be as close as possible to the actual action A. So this network 2 is going to output an action, which will achieve that from state S_t to S_{t+1} . It is hoped that this action is as close to the real action as possible. The advantage of this network 2 is because we want to use $\widehat{\emptyset}(S_{t+1})$ *,* $\widehat{\emptyset}(S_t)$ to forecast Action.Therefore, the feature we extract today is related to the action prediction. So things that are not related to the action the machine is trying to take, such as the wind, are filtered out and are not placed in the extracted vector representation.