[Music]

Welcome to part two of the lecture on reinforcement learning. I want to start with discussing seeing a few important distinctions within this area, so the first distinction is passive versus active learning, the second distinction between on policy and off policy and the third aspect is exploitation versus exploration strategies and the fourth aspect is model based versus model free, reinforcement learning methods. The two first distinctions are very much related, so let's start with the first which is distinction between passive learning and active learning. So of course all agents in all these scenarios perform actions and gets reward and so on, so it's not that passive means doing nothing, so actually this original scenario that we talked about where reinforcement learning still holds, with the cycle of actions, reward, etc. The difference here is that a passive agent executes a fixed policy, which means that you can say this is an agent will no free will or can consider it that this policy can have been given to the agent and the agent is not allowed to change it, so the only thing that the agent can do is to act but always according to a predefined fixed policy, but the alien (02:24) can do, can evaluate what's happened, it can look at what rewards it get, it can look at the returns etc., and it can also learn about the utilities of being in its various states. But this is previous circumscribed existence. In active learning the agent can update its policy as it learns and of course there are different modes of that, and it can do that also while acting in the world and of course an active agent must consider what actions is takes, what the outcomes maybe, how it affects the rewards etc etc. So the active learning scenario is where there is much more dynamic handling of the policy and its consequences and of course active learning is not more typical I will say for the true reinforcement learning case then and then the passive learning.

So what is very much related is this distinction between On policy versus Off policy, still one can say that both this case is more related to an active agent because still in both cases we assume that the agent is interested in changing its policy however in the Off policy mode, you at least for a period stick to an invariant policy and during that period while still obeying that policy the agent can learn in order to define a new policy that can be used in a later stage, so sometimes you call the actual policy that you follow for the behavior policy and you call the one which you want to establish the estimation policy. The alternative is an On policy situation where you actually follow a policy but incrementally and dynamically change that based on experience. So one can say that the planning and learning parts are interwoven while in the off policy case the planning and learning parts are separated.

The third distinction is between exploration and exploitation, so by that is meant that an agent when it takes actions can follow two different approaches. On one hand it could prioritize to follow paths already taken, whether already exist experience from those paths from those episodes, especially by priorities those episodes, those paths that gave more return. The alternative is exploration where you could kind of go into deep water, if I can use that analogy, so instead of just doing what you used to do choosing what was good, not doing what was bad and there of course a lot of avenues you never tested, so in an exploration oriented policy, there is a strong ingredients of trying new paths of course with the ambition and hope that those new path should be more rewarding than the traditionally tried out once. And of course then very solve an issue for the agent to decide it could on the trade-off between these two, because obviously it can make sense to do both at various points in time and under various circumstances. So kind of a little more static way of managing this balance is to actually be more explorative when you have a rather weak knowledge or rather uncomplete view of the environment but when you feel or you can experience that you have a more substantial knowledge about the situation, you could rather fall back on the tried out paths, so but this is more like doing this in one period and doing the other in another period. A more dynamic method which is example is called this one called epsilon greedy really do this judgment on with the final granularity, so actually what one do is on the define a little window so if you have an action which satisfies, which has a probability very close to one then you keep to that, but if you cannot find actions with such a high probabilities then you turn to exploration in general of course this distinction makes more sense with reference to the earlier distinctions for the active case rather than the passive case, I mean of course somebody can give you a predefined policy where you should explore, but as you have known fairly well especially the trade of discussion is irrelevant, and of course exploration versus exploitation may make sense both in a off policy situation and an on policy situation but right one can assume that this way of dynamically balance between these two makes more sense in the online active case.

Finally we have the distinction between modern free and model-based reinforcement learning. So we still talk about the situation where our knowledge about the environment is not complete which means that the complete picture in terms of transition function and reward functions are likely but actually there are then two approaches here what's called model-based is the following then we realize we have an incomplete model of the environment but our approach will be to first to complete that model and using certain schemes when we think we have completed the model to the point where we find acceptable,

we regard it as complete and fall back on the let's say the dynamic programming approach which we already realized works well for the complete knowledge case. So that's the model based approach. The model free approach is that we give up the goal to complete the knowledge, so actually we give up the quest for completing the transition relation and the reward relation, rather we work directly with estimates of both the utilities. One example of that is Q-learning that we will talk little more about and when you set up a special measure Q value which is a particular kind of utility value, which you don't explicitly have to place on the transition and the reward functions. What we will do now is that we will look various approaches to solve reinforcement learning problem in the case of no complete information, and we will talk a little about one particular model based approach which is called adaptive dynamic programming and then we will mostly talk about model free approaches. We will talk about an approach to doing an Direct Estimate, direct estimate of the value function. We will look at Monte Carlo something called Monte Carlo situation, we will talk about temporal difference learning and finally we will accept them exemplify that with a method called Q learning.

So as you can see on the pictures here, you can get a flavour for very rough difference here, so adaptive dynamic programming is model-based it falls back onto the build of a complete model and then applying dynamic program, I hope you remember now the dynamic programming does an exhaustive walkthrough of a breadth-first character, so obviously this fits well with the picture see here, so it actually tests old roots up to a certain point. In contrast to that among the Monte Carlo situation is a method where you make samples, so you will select samples of episodes and this considers the rest and you base your estimates just on those episodes, so one can say that amount of Carlo simulation take few roots but take them all to the end to the terminal point. In contrast to that, the temporal difference methods actually still follow specific paths but do that in a more shallow fashion not always following up the whole episode, rather looking at pairs of states and the differences between pairs of states.

So the simplest method and is actually model freely because it's in the sense that is not actually needs the complete the transition functional or relation and doesn't need a complete reward function, so actually this is a very straightforward method where you look at a sample of epochs or episodes, and then you accumulate the averages along the way for each episode, so here they are in this method they introduced another measure the call ‘reward-to-go’ and actually ‘reward-to-go’ it's just some of may possibly the discounted rewards for each step in the episode. This method keeps a running average over all these ‘rewards-to-go’ for every

episode it looks at. So and then it sets the value function to that average of all these, I mean 'reward-to-go' is not exactly the same as the return as we already defined it, so it's always a slightly different thing, anyway it's somehow ever accumulated to reward and it's for all episodes and then you repeat, you take more example depending on how many trials you take when the trials come to infinity theoretically and the sample average should convert to the true utility for the state, the only problem is this is very time consuming and the convergences is very slow, but otherwise it's very straightforward.

The model-based approach we are going to discuss is a called adaptive dynamic programming. Actually this is an approach, this is very close to what how I generally describe the model-based approach as such, so actually what you do is to first to complete the partially known ADP model which is to complete the full picture of the transition function and the reward function, and then after that you apply dynamic programming as in the simple full knowledge case. So therefore you have some strategy to learn this function and mostly it's a straight forward approach there, so actually you collect examples of rewards for this kind of state action triples and you collect examples also of the transition triples and you take average of collected rewards and you calculate the fractional time such an action leads to a specific other states, so it's basically making a lot of observations of these kinds of triples and making some statistics based on that, so then you easily define these two functions based on that material.

Now we will turn to a model free learning approach called Monte Carlo (MC) simulation. So Monte Carlo methods are computational algorithms that are based on repeated random samples to estimate some property, target property of the model. And actually Monte Carlo simulations simply apply Monte Carlo methods in the context of simulations, so then Monte Carlo reinforcement learning methods learns directly from samples of complete episodes of experience. So this is a limitation this method is built on the analysis of complete one complete episode starting with one state and ending in the terminal state. So Monte Carlo method is model free, no explicit model of the transitions and rewards are built up, so MC takes mean returns for states in all the sampled episodes. The value of expectation of a state is set to the iterative mean of all the empirical returns of the episode. The two variants here slightly different, so you know in one method you only consider the first time you come to a new state because there may be episodes that passes state many times before it enters the terminal but in the other approach you just select or measure data, make observations for every state doesn't matter how often you visit them.

Here you can see an algorithm for first visit Monte Carlo every visit Monte Carlo version is very similar it just modifies step four and in the middle of the algorithm, so it considered States every time there you pass them not only at the first occurrence, for the rest the general idea with the algorithm is you generate episodes using the current policy P and for each generated episodes you go through the episode for each state, you conditionally add a return item to the returns list, so in this returns list you will get as many items that you have that you have States and then during all the iterations you successively calculate the iterative average for all the returns in the list, and then of course in every round you set the value of the value function of the state to that computed average or on the return, when you've done that for an episode you will generate a new episode and you repeat it all over again until some convergence occurs. And actually on the next slide you can see how we calculate the incremental mean, but I don't have any more comments to that.

So here you get a simple example and that shows how the Monte Carlo simulation algorithm actually works. It is just two states and you look at two episodes with a certain structure for the rewards and the formalism is explained in this slide. So then you can see pretty detailed calculation how things are done for the first visit Monte Carlo method, so that you can see that you calculate the return for the first ten state in the first episode and then you calculate to return for A in the second episodes and then you can see that the interesting item here is the average of those, so then the volume function of our all becomes the average and they do the same for the second stage and the similar calculation will happen for the every visit case, so this is all very straightforward.

The last thing we will do in this part is to look at another formal model free learning which called the temporal difference learning or TD. It's actually class of model free reinforcement learning, which learn by bootstrapping from the current estimate of the value function, so actually what this kind of algorithms does is that it's to take a sample from the environment like the Monte Carlo situations and then perform a test based on current estimates like what the window adaptive dynamic programming methods. So while Monte Carlo methods only just estimate once the final outcome is known which means that you consider the whole episode, so here the temporal difference method adjust predictions before the final outcome is known. So actually adjusts the estimated utility value of the current state based on its immediate reward and the estimated value of the next state, so actually it's in principle update based on the relation to the next neighbour. So I assume the word term ‘Temporal’ is motivated by there's normally a temporal relation when you move from a state to another and you at the end you can see the updating equation, where you actually say that the value state

the new value state is the old value state, plus a certain parameter times the reward that you get when you either take the step plus lambda which is this discount which always normally have an in this situation, times the estimate for the next step Minus the state you are, so and of course by choosing different values of alpha and lambda you can get slightly different functionality here. Here so this lecture will be continued soon in part in the part 3 video. Thank you!