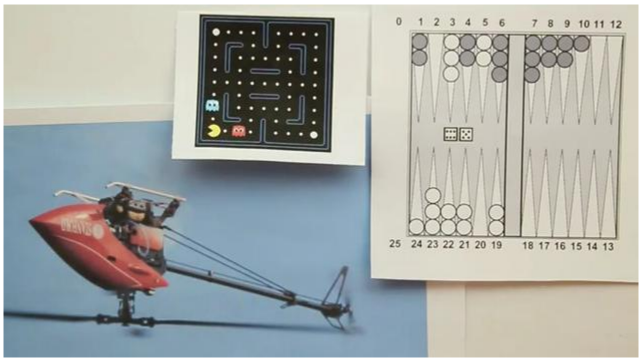
Computational Intelligence

Unit # 17

(Guest Lecture by Ms. Saleha Raza)

Reinforcement Learning



Acknowledgement

- Several examples of this lecture have been taken from Stanford AI class and Stanford Machine Learning class.

Different types of learning

- Supervised learning
- Unsupervised Learning
- Learning via evolution
- Reinforcement Learning

Reinforcement Learning(RL)

- An RL agent learns by interacting with its environment and observing the results of these interactions. This mimics the fundamental way in which humans (and animals alike) learn. As humans, we have a direct sensory-motor connection to our environment, meaning we can perform actions and witness the results of these actions on the environment.
- The key idea can be translated into the following steps for an RL agent:
  - The agent observes an input state
  - An action is determined by a decision making function (policy)
  - The action is performed
  - The agent receives a scalar reward or reinforcement from the environment
  - Information about the reward given for that state / action pair is recorded
- By performing actions, and observing the resulting reward, the policy used to determine the best action for a state can be fine-tuned.
- Eventually, if enough states are observed an optimal decision policy will be generated and we will have an agent that performs perfectly in that particular environment.

Reinforcement Learning

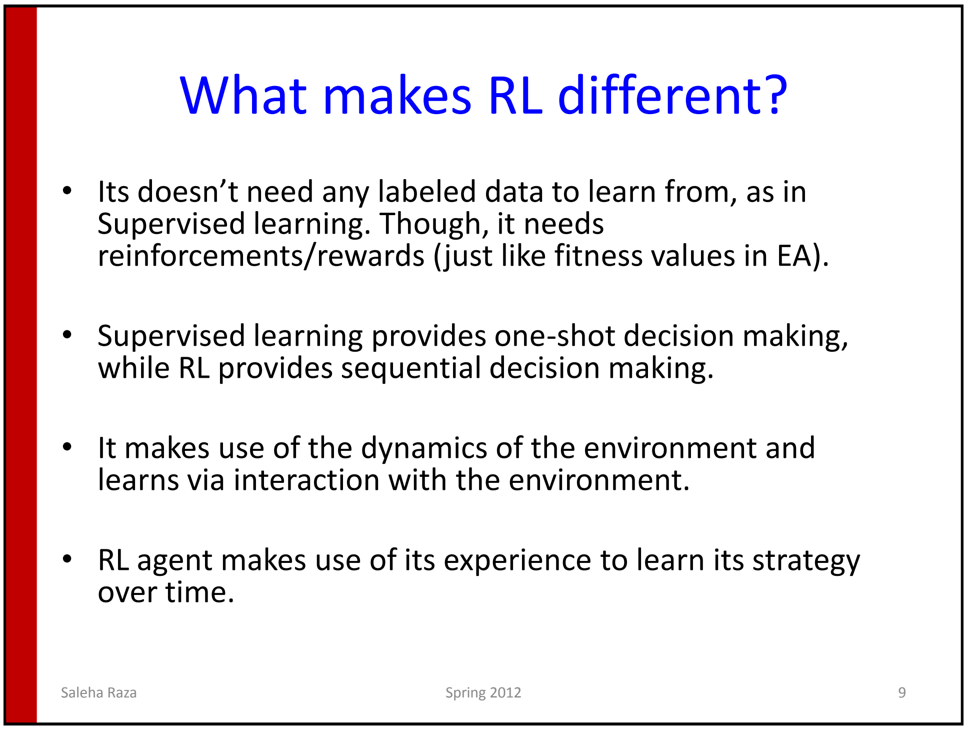
Reinforcement Learning

- RL is distinguished from other computational approaches by its emphasis on learning by the individual from direct interaction with its environment, without relying on exemplary supervision or complete models of the environment.

- Their use of value functions distinguishes reinforcement learning methods from evolutionary methods that search directly in policy space guided by scalar evaluations of entire policies.

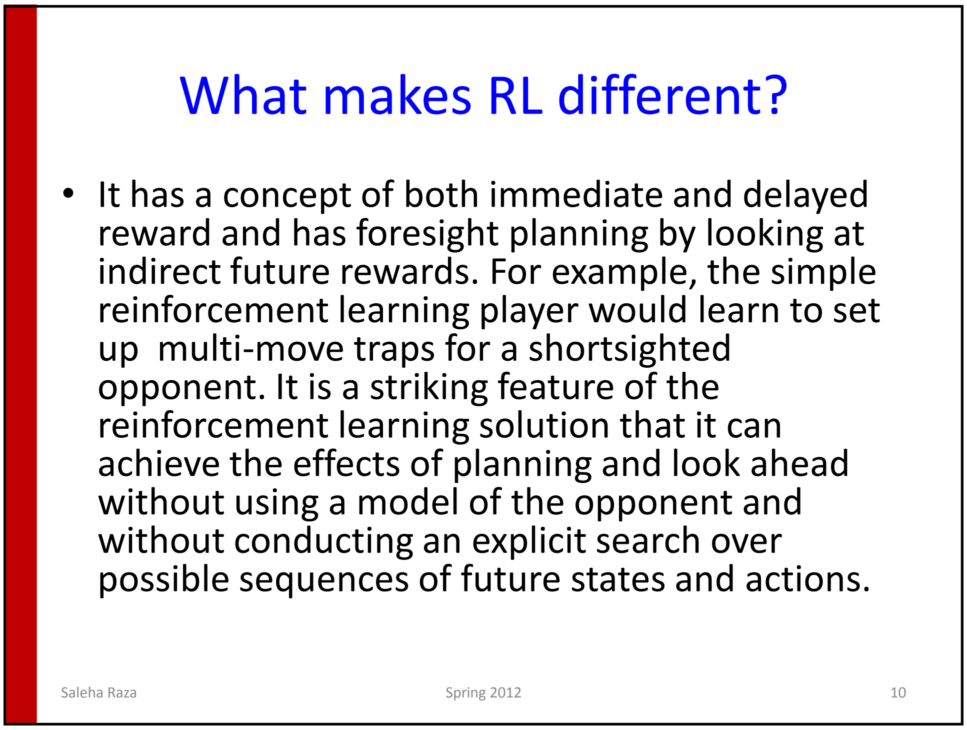
Reinforcement Learning

- Clearly, such an agent:
  - must be able to sense the state of the environment to some extent and
  - must be able to take actions that affect the state. The agent also must
  - have a goal or goals relating to the state of the environment.
- The formulation is intended to include just these three aspects--sensation, action, and goal--in their simplest possible forms without trivializing any of them.



What makes RL different?

- Its doesn’t need any labeled data to learn from, as in Supervised learning. Though, it needs reinforcements/rewards (just like fitness values in EA).
- Supervised learning provides one-shot decision making, while RL provides sequential decision making.
- It makes use of the dynamics of the environment and learns via interaction with the environment.
- RL agent makes use of its experience to learn its strategy over time.



What makes RL different?

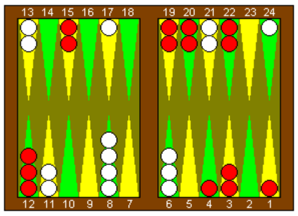
- It has a concept of both immediate and delayed reward and has foresight planning by looking at indirect future rewards. For example, the simple reinforcement learning player would learn to set up multi-move traps for a shortsighted opponent. It is a striking feature of the reinforcement learning solution that it can achieve the effects of planning and look ahead without using a model of the opponent and without conducting an explicit search over possible sequences of future states and actions.

Some RL Examples

- A master chess player makes a move. The choice is informed both by planning—anticipating possible replies and counter replies--and by immediate, intuitive judgments of the desirability of particular positions and moves.
- An adaptive controller adjusts parameters of a petroleum refinery's operation in real time. The controller optimizes the yield/cost/quality trade-off on the basis of specified marginal costs without sticking strictly to the set points originally suggested by engineers.
- A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour.
- A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past.

Some popular implementation of RL

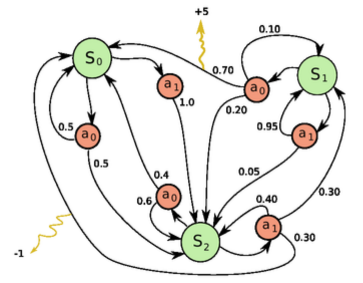
Stanford autonomous helicopter
http://ai.stanford.edu/~pabbeel/RL-videos.html



TD Gammon – A RL agent of Backgammon
http://www.research.ibm.com/massive/tdl.html

Markov Decision Process (MDP)

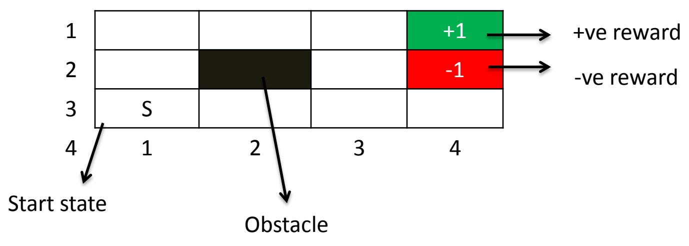
- RL problems models the world using Markov Decision Processes (MDPs).
- MDP provides a mathematical framework for modeling decision-making in stochastic situations.
- At each time step, the MDP is in some **state** , and the decision maker may choose any **action** that is available in state . The process responds at the next time step by randomly moving into a new state , and giving the decision maker a corresponding **reward** . The probability that the process moves into its new state is influenced by the chosen action. Specifically, it is given by the **state transition function** .



Markov Decision Process (MDP)

- A Markov decision process is a 4-tuple (S, A, P(.,.),R(.,.)) , where
  - S is a finite set of states,
  - A is a finite set of actions (alternatively, is the finite set of actions available from state),
  - $P_a(s,s')$ is the probability that action 'a' in state 's' at time 't' will lead to state s' at time t+1
  - R(s') is the expected immediate reward received after transition to state s'.
- **Objective**: The core problem of MDPs is to find a *policy* for the decision maker: a function that specifies the action that the decision maker will choose when in state .

GridWorld Example



The objective is to find a policy that navigates the agent from the start state to the Goal state while resulting in maximum +ve reward.

Any idea, how to model this problem using Evolution or Swarm Intelligence?

Elements of RL

- Beyond the agent and the environment, one can identify four main sub elements of a reinforcement learning system:
  - a policy,
  - a reward function,
  - a value function,
  - and, optionally, a model of the environment.

Reward Function

- A reward function defines the goal in a reinforcement learning problem. Roughly speaking, it maps each perceived state (or state-action pair) of the environment to a single number, a reward, indicating the intrinsic desirability of that state.
- A reinforcement learning agent's sole objective is to maximize the total reward it receives in the long run. The reward function defines what are the good and bad events for the agent. In a biological system, it would not be inappropriate to identify rewards with pleasure and pain. They are the immediate and defining features of the problem faced by the agent.

Value Function

- Whereas a reward function indicates what is good in an immediate sense, a value function specifies what is good in the long run.
- Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.
- For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true.
- To make a human analogy, rewards are like pleasure (if high) and pain (if low), whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state. Expressed this way, we hope it is clear that value functions formalize a basic and familiar idea.

Reward vs Values

- Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary.
- Without rewards there could be no values, and the only purpose of estimating values is to achieve more reward. Nevertheless, it is values with which we are most concerned when making and evaluating decisions. Action choices are made based on value judgments. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run. In decision-making and planning, the derived quantity called value is the one with which we are most concerned.
- Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and reestimated from the sequences of observations an agent makes over its entire lifetime.
- In fact, the most important component of almost all reinforcement learning algorithms is a method for efficiently estimating values. The central role of value estimation is arguably the most important thing we have learned about reinforcement learning over the last few decades.

Value Iteration

- Let the path taken by an RL agent to reach the goal state is:

  $S_0 \text{->} S_1 \text{->} S_2 \text{->} S_3 \text{->}\ldots..$

  $\text{Value}(S_0) = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \gamma^3 R(S_3) + \ldots..$

  $\text{Value}(S_0) = R(S_0) + \gamma V(S_1)$

  Since state transitions are stochastic, we have

  $\text{Value}(S_0) = R(S_0) + \gamma E[\, V(S_1)\,]$

  Where E represents 'expected value' and γ is discount factor.

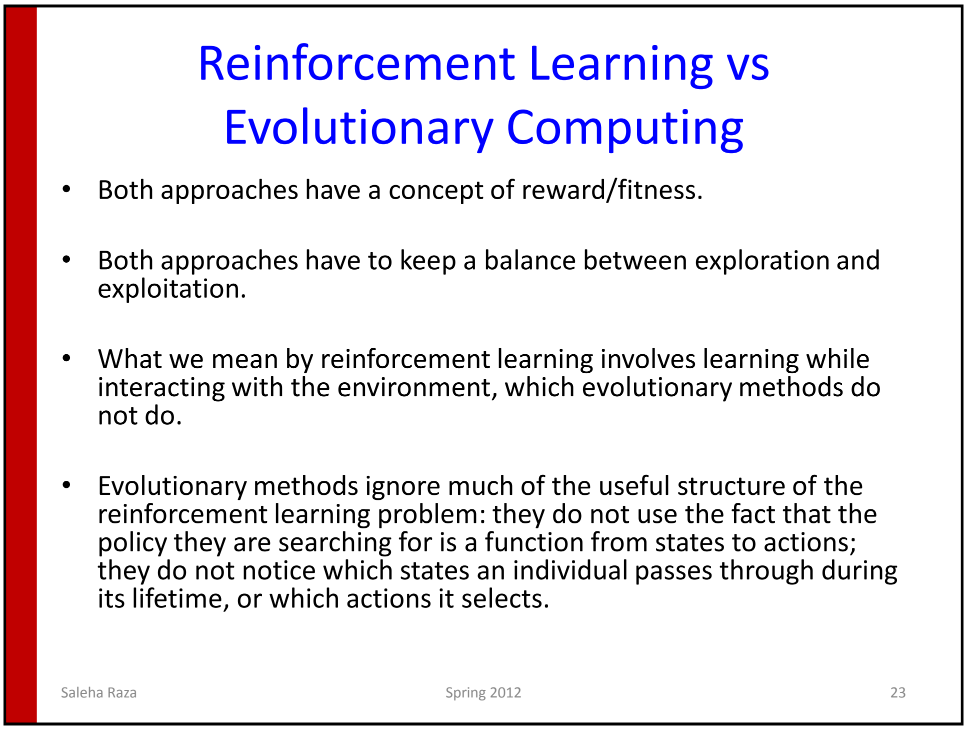
  Discount factor: The discount factor determines the importance of future rewards. A factor of 0 will make the agent "opportunistic" by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

Policy

- A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states.
- It corresponds to what in psychology would be called a set of stimulus-response rules or associations.
- In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic.

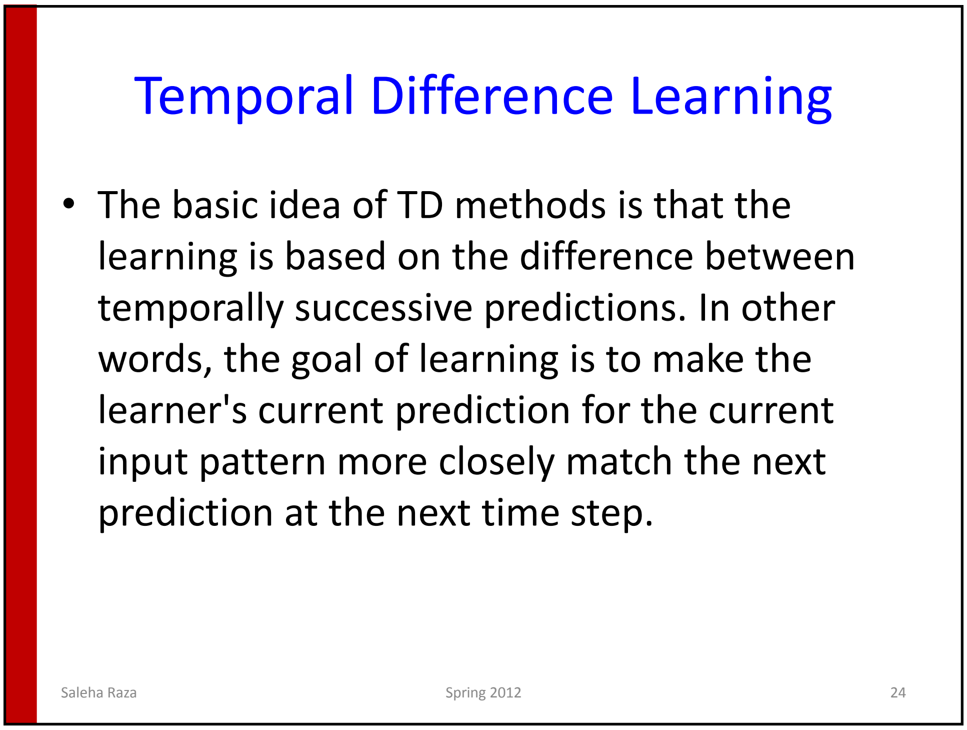
Model

- The fourth and final element of some reinforcement learning systems is a *model* of the environment.
- This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for *planning*, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced.
- RL models the environment in the form of MDPs.



Reinforcement Learning vs Evolutionary Computing

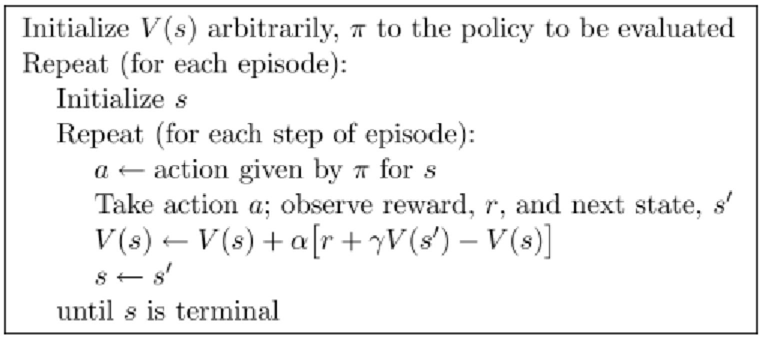
- Both approaches have a concept of reward/fitness.
- Both approaches have to keep a balance between exploration and exploitation.
- What we mean by reinforcement learning involves learning while interacting with the environment, which evolutionary methods do not do.
- Evolutionary methods ignore much of the useful structure of the reinforcement learning problem: they do not use the fact that the policy they are searching for is a function from states to actions; they do not notice which states an individual passes through during its lifetime, or which actions it selects.



Temporal Difference Learning

- The basic idea of TD methods is that the learning is based on the difference between temporally successive predictions. In other words, the goal of learning is to make the learner's current prediction for the current input pattern more closely match the next prediction at the next time step.

Temporal Difference Learning



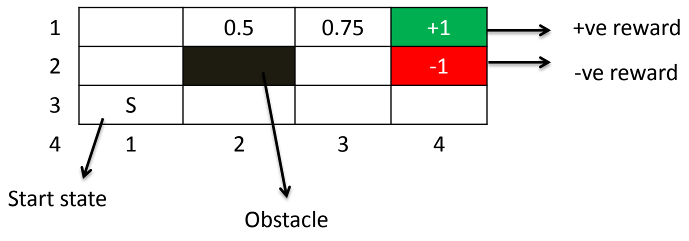
```
Initialize $V(s)$ arbitrarily, $\pi$ to the policy to be evaluated
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        $a \leftarrow$ action given by $\pi$ for $s$
        Take action $a$; observe reward, $r$, and next state, $s'$
        $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$
        $s \leftarrow s'$
    until $s$ is terminal
```

Figure 6.1: Tabular TD(0) for estimating V^{π}.

Temporal Difference Learning

- **Learning rate**
- The learning rate determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information.

Temporal Difference Learning



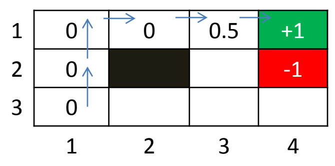
$$V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$$

Temporal Difference Learning

Let alpha = 0.5 and gamma = 1

All initial values and rewards are zero.

$$V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$$



- The updated values as our RL agent moves are as follows:

  V(S21) = 0 + 0.5[0 + 1(0) – 0)] = 0

  Similarly,

  V(11) = V(12) = 0

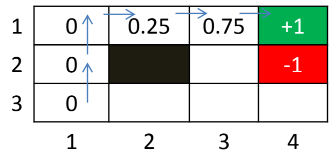
  However,

  V(13) = 0 + 0.5[0 + 1(1) – 0] = 0.5

Temporal Difference Learning

Let alpha = 0.5 and gamma = 1
All initial values and rewards are zero.

$$V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$$



	1	2	3	4
1	0	0.25	0.75	+1
2	0			-1
3	0			

- In next iteration, the following updates, the following values will be updated:

  V(12) = 0 + 0.5[0 + 1(0.5) – 0] = 0.25

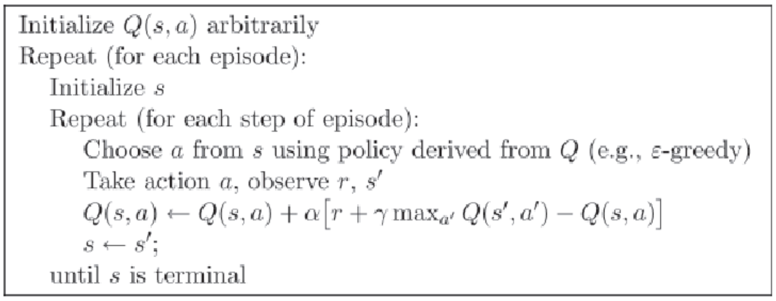
  V(13) = 0.5 + 0.5[0 + 1(1) -0.5] = 0.75

Q-learning

- Q-learning is a reinforcement learning algorithm that does not need a model of its environment and can be used online. Q-learning algorithms work by estimating the values of state-action pairs. The value Q(s, a) is defined to be the expected discounted sum of future payoffs obtained by taking action a from state s and following the current optimal policy thereafter. Once these values have been learned, the optimal action from any state is the one with the highest Q-value.

- The agent can perform adaptively in a world without understanding it. All it tries to do is sort out good actions to perform from bad ones.

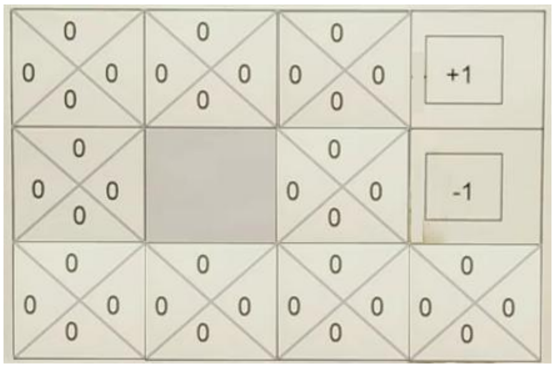
QLearning



```
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $a$, observe $r$, $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
        $s \leftarrow s'$;
    until $s$ is terminal
```

Figure 6.12:Q-learning: An off-policy TD control algorithm.

Q-learning



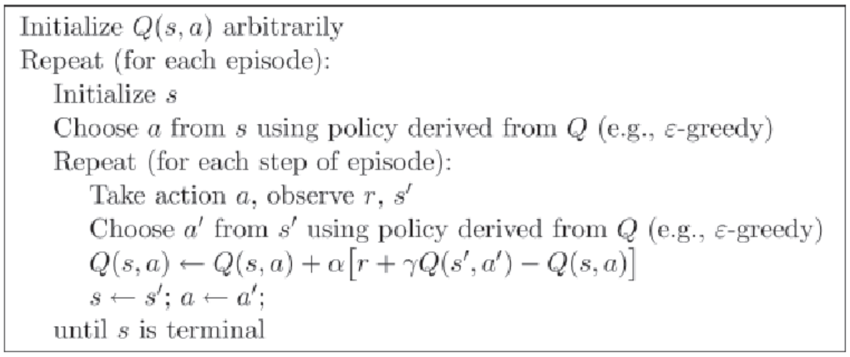
Instead of having 'value' for each state. We now have a value for each state, action (s,a) pair.

SARSA

This name simply reflects the fact that the main function for updating the Q-value depends on the current state of the agent "**S**$_1$", the action the agent chooses "**A**$_1$", the reward "**R**" the agent gets for choosing this action, the state "**S**$_2$" that the agent will now be in after taking that action, and finally the next action "**A**$_2$" the agent will choose in its new state. Taking every letter in the quintuple (s_t , a_t , r_t , s_{t+1} , a_{t+1}) yields the word *SARSA*.

SARSA is similar to Q-learning. The only difference is that SARSA learns next state's action relative to the policy it follows, while Q-Learning does it relative to the greedy policy and selects the action that gives maximum Q value.

SARSA



```
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Repeat (for each step of episode):
        Take action $a$, observe $r$, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
        $s \leftarrow s'$; $a \leftarrow a'$;
    until $s$ is terminal
```

Action Selection Policies

- There are three common policies used for action selection. The aim of these policies is to balance the trade-off between exploitation and exploration, by not always exploiting what has been learnt so far.
- **ε-greedy** - most of the time the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a small probability , an action is selected at random. The action is selected uniformly, independent of the action-value estimates. This method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal actions are discovered.
- **ε -soft** - very similar to -greedy. The best action is selected with probability 1-ε and the rest of the time a random action is chosen uniformly.
- **softmax** - one drawback of ε-greedy and ε-soft is that they select random actions uniformly. The worst possible action is just as likely to be selected as the second best. Softmax remedies this by assigning a rank or weight to each of the actions, according to their action-value estimate. A random action is selected with regards to the weight associated with each action, meaning the worst actions are unlikely to be chosen. This is a good approach to take where the worst actions are very unfavorable.

Some Videos

- http://www.youtube.com/watch?v=tovrpoUkzYU
- http://www.youtube.com/watch?v=Xf_IhCbTQGY&feature=BFa&list=FLTnVTxTVQSBDkVSWPhBpRTQ

How to deal with continuous domains?

- Discretization
- Function Approximation

References

- R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- http://reinforcementlearning.ai-depot.com/Main.html
- http://en.wikipedia.org/wiki/Markov_decision_process