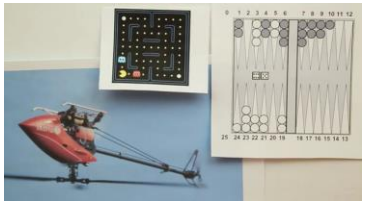
Computational Intelligence
Unit # 14

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.

Reinforcement Learning(RL)

- 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

- 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.***

Reinforcement Learning

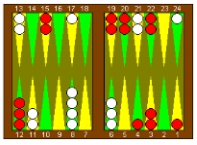
- Clearly, such an agent:
  - must be able to sense the state of the environment to some extent and
  - must be able to take actions that affect the state.
  - The agent also must have a goal or goals relating to the state of the environment.

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

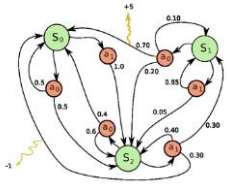
Stanford autonomous helicopter
http://ai.stanford.edu/~pabbeel/RL-videos.html



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

Markov Decision Process (MDP)

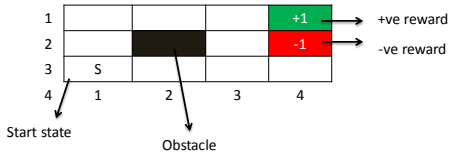
- RL problems models the world using Markov Decision Processes (MDPs).
- MDP provides a mathematical framework for modeling decision-making in stochastic situations.
- At each time step, the MDP is in some **state** , and the decision maker may choose any **action** that is available in state . The process 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.
- 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.

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.

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.

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.
- EA looks for the immediate fitness of a given state and does not accommodate delayed reward.
- 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.

Value Iteration

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

  S_0 -> S_1 -> S_2 -> S_3 ->.....

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

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

Since state transitions are stochastic, we have

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

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.

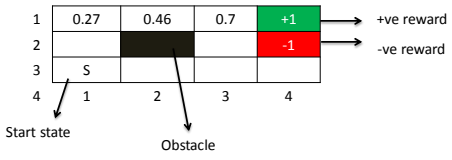
Value Iteration

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

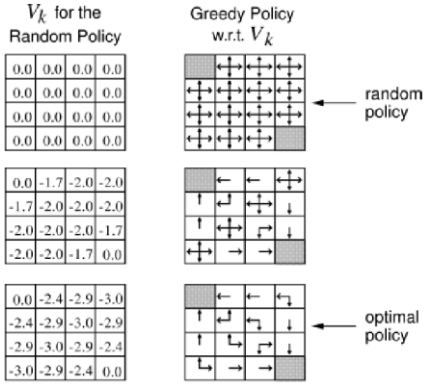
Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Value Iteration



$$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$$

Value Iteration



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

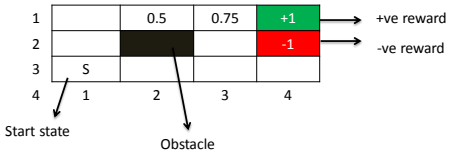
Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by π for s
    Take action a; observe reward, r, and next state, s'
    $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$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

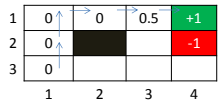


$$V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$$

Temporal Difference Learning

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

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



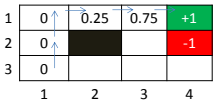
- The updated values as our RL agent moves are as follows:

  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.

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



- In next iteration, 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

Driving Home Example

- Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of week, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6 o'clock, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. Traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes. Unfortunately, at this point you get stuck behind a slow truck, and the road is too narrow to pass. You end up having to follow the truck until you turn onto the side street where you live at 6:40. Three minutes later you are home.

Driving Home Example

- The sequence of states, times, and predictions is thus as follows:

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Driving Home Example

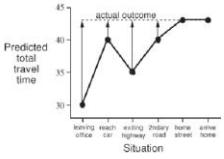
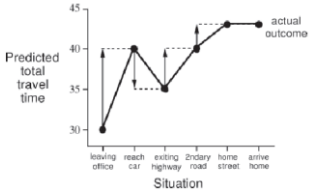


Figure 6.3:Changes recommended by Monte Carlo methods in the driving home example.



Action Selection Policies

- **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.

Action Selection - Softmax

- **Softmax -** One drawback of ε-greedy is that when it explores it chooses equally among all actions. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action. In tasks where the worst actions are very bad, this may be unsatisfactory. The obvious solution is to vary the action probabilities as a graded function of estimated value. The greedy action is still given the highest selection probability, but all the others are ranked and weighted according to their value estimates. These are called *softmax* action selection rules.

Softmax

- The most common softmax method uses a Gibbs, or Boltzmann, distribution. It chooses action on the th play with probability

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}},$$

- where τ is a positive parameter called the *temperature*. High temperatures cause the actions to be all (nearly) equiprobable. Low temperatures cause a greater difference in selection probability for actions that differ in their value estimates. In the limit as $\tau \to 0$, softmax action selection becomes the same as greedy action selection.