Fundamentals of Artificial Intelligence

Reinforcement Learning

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

Reinforcement Learning

In Reinforcement learning, the information available for training is intermediate between supervised and unsupervised learning. Instead of training examples that indicate the correct output for a given input, the training data are assumed to provide only an indication as to whether an action is correct or not.

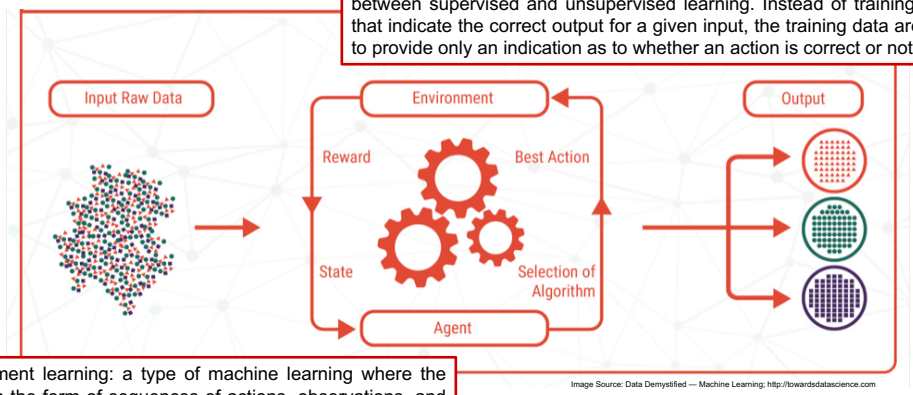


Image Source: Data Demystified — Machine Learning: http://towardsdatascience.com

Reinforcement learning: a type of machine learning where the data are in the form of sequences of actions, observations, and rewards, and the learner learns how to take actions to interact in a specific environment so as to maximise the specified rewards.

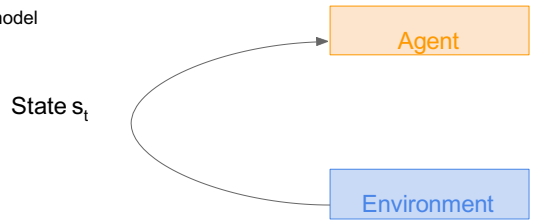
Learn **action to maximize payoff.**

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

In the standard reinforcement learning model an agent interacts with its environment.

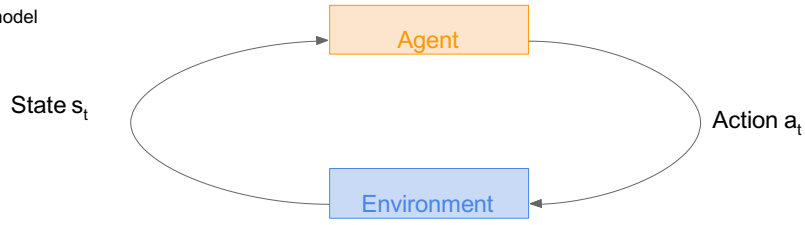


© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

In the standard reinforcement learning model an agent interacts with its environment.



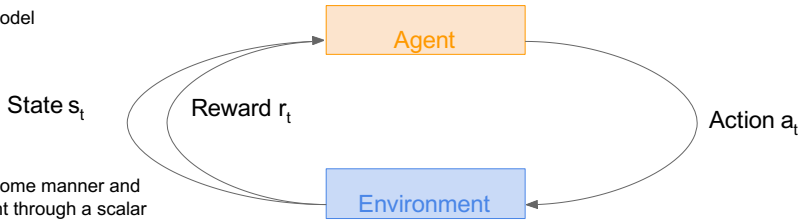
This interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment.

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

In the standard reinforcement learning model an agent interacts with its environment.



The action changes the environment in some manner and this change is communicated to the agent through a scalar reinforcement signal.

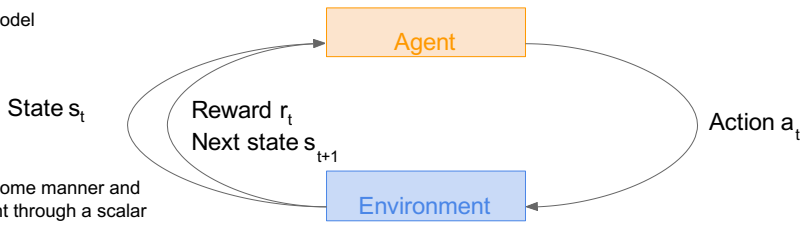
This interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment.

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

In the standard reinforcement learning model an agent interacts with its environment.



The action changes the environment in some manner and this change is communicated to the agent through a scalar reinforcement signal.

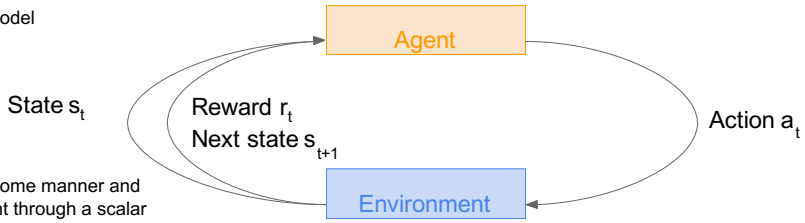
This interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment.

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

In the standard reinforcement learning model an agent interacts with its environment.



The action changes the environment in some manner and this change is communicated to the agent through a scalar reinforcement signal.

This interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment.

Learning is based on the reward hypothesis – All goals can be described by maximization of expected cumulative rewards.

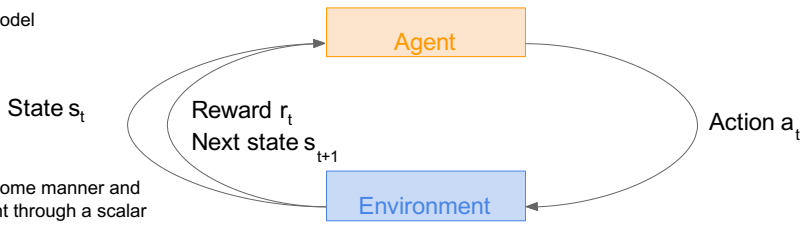
© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

More general than supervised or unsupervised learning. Learn from interaction with the environment to achieve a goal.

Reinforcement Learning: Concerned with the **problem of finding suitable actions** to take in a given situation **in order to maximize a reward.**

In the standard reinforcement learning model an agent interacts with its environment.



The action changes the environment in some manner and this change is communicated to the agent through a scalar reinforcement signal.

This interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment.

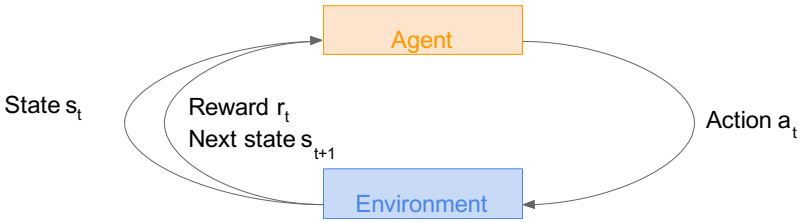
Learning is based on the reward hypothesis – All goals can be described by maximization of expected cumulative rewards.

© Shyamanta M Hazarika, ME, IIT Guwahati

Components of a RL Agent

An RL agent may include one or more of these components:

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment



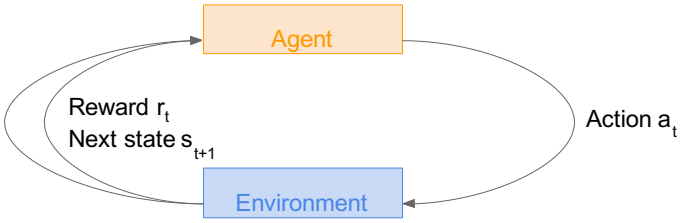
© Shyamanta M Hazarika, ME, IIT Guwahati

Components of a RL Agent

An RL agent may include one or more of these components:

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$



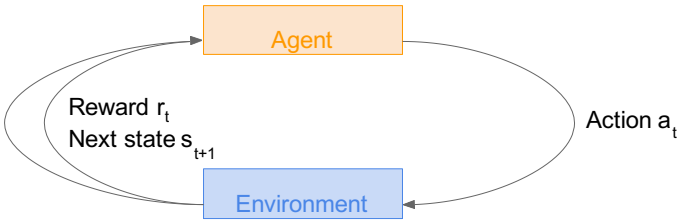
© Shyamanta M Hazarika, ME, IIT Guwahati

Components of a RL Agent

An RL agent may include one or more of these components:

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$



- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

11

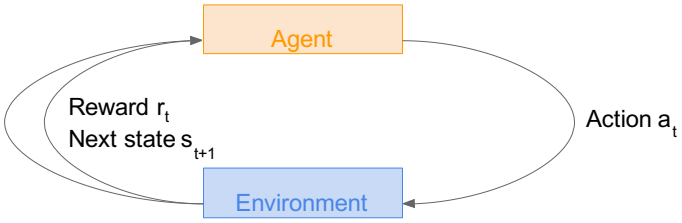
© Shyamanta M Hazarika, ME, IIT Guwahati

Components of a RL Agent

An RL agent may include one or more of these components:

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$



- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

12

© Shyamanta M Hazarika, ME, IIT Guwahati

Elements of RL Problem

1. The Environment

- Every RL system learns a mapping from situations to actions by trial-and-error interactions with a dynamic environment.
- This environment must at least be partially observable; the observations may come in the form of sensor readings, symbolic descriptions, or possibly "mental" situations.
- If the RL system can observe perfectly all the information in the environment that might influence the choice of action to perform, then the RL system chooses actions based on true "states" of the environment.
- This ideal case is the best possible basis for reinforcement learning and, in fact, is a necessary condition for much of the associated theory.

© Shyamanta M Hazarika, ME, IIT Guwahati

Elements of RL Problem

2. The Reinforcement Function

RL systems learn a mapping from situations to actions by trial-and-error interactions with a dynamic environment.

- The "goal" of the RL system is defined using the concept of a reinforcement function, which is the exact function of future reinforcements the agent seeks to maximize.
- There exists a mapping from state/action pairs to reinforcements; after performing an action in a given state the RL agent will receive some reinforcement (reward) in the form of a scalar value.
- The RL agent learns to perform actions that will maximize the sum of the reinforcements received when starting from some initial state and proceeding to a terminal state.

© Shyamanta M Hazarika, ME, IIT Guwahati

Elements of RL Problem

3. The Value Function

Having the environment and the reinforcement function defined; the question now is how the agent learns to choose "good" actions.

- A *policy* determines which action should be performed in each state; a policy is a mapping from states to actions.
- The *value* of a state is defined as the sum of the reinforcements received when starting in that state and following some fixed policy to a terminal state.
- The *optimal policy* would be the mapping from states to actions that maximizes the sum of the reinforcements. The value of a state is dependent upon the policy.
- The value function is a mapping from states to state values and can be approximated using any type of function approximator (e.g., multilayered perceptron, memory based system, radial basis functions, look-up table, etc.).

© Shyamanta M Hazarika, ME, IIT Guwahati

The Reinforcement Function

- There are at least three noteworthy classes often used to construct reinforcement functions that properly define the desired goals.

1. Pure Delayed Reward and Avoidance Problems

   - In Pure Delayed Reward class of functions the reinforcements are all zero except at the terminal state.
   - The sign of the scalar reinforcement at the terminal state indicates whether the terminal state is a goal state (a reward) or a state that should be avoided (a penalty).

© Shyamanta M Hazarika, ME, IIT Guwahati

The Reinforcement Function

- There are at least three noteworthy classes often used to construct reinforcement functions that properly define the desired goals.

  1. Pure Delayed Reward and Avoidance Problems

Example of a pure delayed reward reinforcement function: Standard cart-pole or inverted pendulum problem. A cart supporting a hinged, inverted pendulum is placed on a finite track. The goal of the RL agent is to learn to balance the pendulum in an upright position without hitting the end of the track. The situation (state) is the dynamic state of the cart pole system. Two actions are available to the agent in each state: move the cart left, or move the cart right. The reinforcement function is zero everywhere except for the states in which the pole falls or the cart hits the end of the track, in which case the agent receives a -1 reinforcement.

© Shyamanta M Hazarika, ME, IIT Guwahati

The Reinforcement Function

- There are at least three noteworthy classes often used to construct reinforcement functions that properly define the desired goals.

  2. Minimum Time to Goal

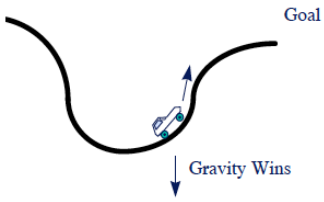
     - Reinforcement functions in this class cause an agent to perform actions that generate the shortest path or trajectory to a goal state.

© Shyamanta M Hazarika, ME, IIT Guwahati

The Reinforcement Function

- There are at least three noteworthy classes often used to construct reinforcement functions that properly define the desired goals.

2. Minimum Time to Goal



An example is one commonly known as the "Car on the hill" problem. The problem is defined as that of a stationary car being positioned between two steep inclines. The goal of the RL agent is to successfully drive up the incline on the right to reach a goal state at the top of the hill. The state of the environment is the car's position and velocity. Three actions are available: forward thrust, backward thrust, or no thrust at all. Agent learn to use momentum to gain velocity to successfully climb the hill. The reinforcement function is -1 for ALL state transitions except the transition to the goal state, in which case a zero reinforcement is returned.

© Shyamanta M Hazarika, ME, IIT Guwahati

The Reinforcement Function

- There are at least three noteworthy classes often used to construct reinforcement functions that properly define the desired goals.

3. Games

  - An alternative reinforcement function would be used in the context of a game environment - wo or more players with opposing goals.
  - RL system can learn to generate optimal behavior for the players; finding the minimax, or saddlepoint of the reinforcement function.
  - Agent would evaluate the state for each player and would choose an action independent of the other players action.
  - Actions are chosen independently and executed simultaneously, the RL agent learns to choose actions for each player that would generate the best outcome for the given player in a "worst case" scenario.

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

- In our discussion of Markov Decision Problems, we assumed that we knew the agent's reward function, R, and a model of how the world works, expressed as the transition probability distribution.
- In reinforcement learning, we would like an agent to learn to behave well in an MDP world, but without knowing anything about reward function or the transition probability distribution when it starts out.

Parameter Estimation - you can estimate the next-state distribution P(s'|s, a) by counting the number of times the agent has taken action a in state s and looking at the proportion of the time that s' has been the next state. Similarly, you can estimate R(s) just by averaging all the rewards you've received when you were in state s.

© Shyamanta M Hazarika, ME, IIT Guwahati

Approximating the Value Function

- Reinforcement learning is a difficult problem because the learning system may perform an action and not be told whether that action was good or bad.
- Initially, the approximation of the optimal value function is poor. In other words, the mapping from states to state values is not valid.
- The primary objective of learning is to find the correct mapping. Once this is completed, the optimal policy can easily be extracted.
  - Q-learning - finds a mapping from state/action pairs to values; one of the most successful approaches to RL.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- Deterministic Markov decision process - the state transitions are deterministic
  - An action performed in state x_t always transitions to the same successor state x_{t+1} .
  - In a nondeterministic Markov decision process, a probability distribution function defines a set of potential successor states for a given action in a given state.
- If the MDP is non-deterministic, then value iteration requires that we find the action that returns the maximum expected value
  - The sum of the reinforcement and the integral over all possible successor states for the given action.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- Theoretically, value iteration is possible in the context of non-deterministic MDPs.
  - Computationally impossible to calculate the necessary integrals without added knowledge or some degree of modification.
- Q-learning solves the problem of having to take the max over a set of integrals.
- Rather than finding a mapping from states to state values (as in value iteration), Q-learning finds a mapping from state/action pairs to values (called Q-values).
  - Q-value is the sum of the reinforcements received when performing the associated action and then following the given policy thereafter.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Function

- *Q*(s,a)* is the expected discounted future reward for starting in state s, taking action a, and continuing optimally thereafter.

Assuming we have some way of choosing actions, now we're going to focus on finding a way to estimate the value function directly.

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} \Pr(s' \mid s,a) \max_{a'} Q^*(s',a')$$

The Q value of being in state s and taking action a is the immediate reward, *R(s)*, plus the discounted expected value of the future. We get the expected value of the future by taking an expectation over all possible next states, *s'*. In each state *s'*, we need to know the value of behaving optimally. We can get that by choosing, in each *s'*, the action *a'* that maximizes *Q*(s',a')*.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Function

- *Q*(s,a)* is the expected discounted future reward for starting in state s, taking action a, and continuing optimally thereafter.

Assuming we have some way of choosing actions, now we're going to focus on finding a way to estimate the value function directly.

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} \Pr(s' \mid s,a) \max_{a'} Q^*(s',a')$$

- If you know Q*, then it's really easy to compute the optimal action in a state.
  - Take the action that gives the largest Q value in that state.
- When using V*, it required knowing the transition probabilities to compute the optimal action, so this is considerably simpler.
- And it will be effective when the model is not explicitly known.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- Q learning, which estimates the *Q** function directly, without estimating the transition probabilities.
  - Once we have Q*, finding the best way to behave is easy.
- The learning algorithm deals with individual "pieces" of experience with the world.
  - One piece of experience is a set of the current state s, the chosen action a, the reward r, and the next state s'.
  - Each piece of experience will be folded into the Q values, and then thrown away!

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- A piece of experience in the world is (s,a,r,s')

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- A piece of experience in the world is (s,a,r,s')
  - Initialize *Q(s,a)* arbitrarily

As in value iteration, by initializing the Q function arbitrarily. Zero is usually a reasonable starting point.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- A piece of experience in the world is (s,a,r,s')
  - Initialize *Q(s,a)* arbitrarily
  - After each experience, update Q:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha q(r,s')$$

As in value iteration, by initializing the Q function arbitrarily. Zero is usually a reasonable starting point.

The basic form of the update looks like this. The parameter alpha is a learning rate; usually it's something like 0.1. So, we're updating our estimate of Q(s,a) to be mostly like our old value of Q(s,a), but adding in a new term that depends on r and the new state.

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- A piece of experience in the world is (s,a,r,s')
  - Initialize *Q(s,a)* arbitrarily
  - After each experience, update Q:

As in value iteration, by initializing the Q function arbitrarily. Zero is usually a reasonable starting point.

The basic form of the update looks like this. The parameter alpha is a learning rate; usually it's something like 0.1. So, we're updating our estimate of Q(s,a) to be mostly like our old value of Q(s,a), but adding in a new term that depends on r and the new state.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha q(r,s')$$

$$q(r,s') = r + \gamma \max_{a'} Q(s',a')$$

q(r,s) is an example of the value of taking action a in state s. The actual reward, r, is a sample of the expected reward R(s). And the actual next state, s', is a sample from the next state distribution. And the value of that state s' is the value of the best action we can take in it, which is the max over a' of Q(s',a').

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- A piece of experience in the world is (s,a,r,s')
  - Initialize *Q(s,a)* arbitrarily
  - After each experience, update Q:

As in value iteration, by initializing the Q function arbitrarily. Zero is usually a reasonable starting point.

The basic form of the update looks like this. The parameter alpha is a learning rate; usually it's something like 0.1. So, we're updating our estimate of Q(s,a) to be mostly like our old value of Q(s,a), but adding in a new term that depends on r and the new state.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha q(r,s')$$

$$q(r,s') = r + \gamma \max_{a'} Q(s',a')$$

q(r,s) is an example of the value of taking action a in state s. The actual reward, r, is a sample of the expected reward R(s). And the actual next state, s', is a sample from the next state distribution. And the value of that state s' is the value of the best action we can take in it, which is the max over a' of Q(s',a').

- Guaranteed to converge to optimal Q if the world is really an MDP

© Shyamanta M Hazarika, ME, IIT Guwahati

Q-Learning

- Requires that the states and actions be drawn from a small enough set that we can store the Q function in a table.
  - Large or even continuous state spaces, make the direct representation approach impossible.
  - We can try to use a function approximator, such as a neural network, to store the Q function, rather than a table.
- Q-Learning can sometimes be very slow to converge. More advanced techniques in reinforcement learning are aimed at addressing this problem.

© Shyamanta M Hazarika, ME, IIT Guwahati

Reinforcement Learning

- Reinforcement learning is appealing because of its generality.
  - Any problem domain that can be cast as a Markov decision process can potentially benefit from this technique
- Reinforcement learning is a promising technology; but possible refinements that will have to be made before it has truly widespread application.
  - Reinforcement learning is an extension of classical dynamic programming in that it greatly enlarges the set of problems that can practically be solved.
  - Combining dynamic programming with neural networks, many are optimistic of solving a large class of problems.

© Shyamanta M Hazarika, ME, IIT Guwahati