DP is used to solve MDP's and similar problems. ISE-2

DP is used to solve MDP's and similar problems.
DP aim to find an optimal policy

Dynamic Programming

- DP is a method of solving complex prob. by breaking them down into sub problems. The solutions to the sub-problems are combined to solve overall problem.
- The basic assumption is that the model of the env is given.
- The two req. properties of DP are -

1) Optimal substructure -
optimal solⁿ of the sub-problem can be used to solve the overall problem.

2) Overlapping sub-problems -
sub-problems occur many times. sol? can be cached & reused.

- ~~Twopo~~ popular DP algos are -
  a) Policy iteration
  b) Value iteration.

} to find optimal policy of MDP

POLICY ITERATION:
- Involves iteratively improving a policy until it converges to the optimal policy.
- After ~~initit~~ initializing a random policy, this method iterates betⁿ 2 steps
  - Policy evaluation
  - Policy improvement

(P.T.O.)

$V_\pi(s) = E[R_\pi(s)]$

$\cancel{q_\pi(s) = r + \gamma E[R_\pi(s', a)]}$

$q_\pi(s) = r + \gamma V_\pi(s')$

Policy Evaluation:

i) Evaluates the current policy by ~~evaluating~~ computing the expected value of reward obtained by following the policy.
$E[R_\pi(s)]$

ii) Measures how good the policy is by ~~to~~ calculating ~~all~~ the state value function $V_\pi(s)$ for all states until the state value function is converged.

iii) Also known as the prediction problem.

Policy Improvement:

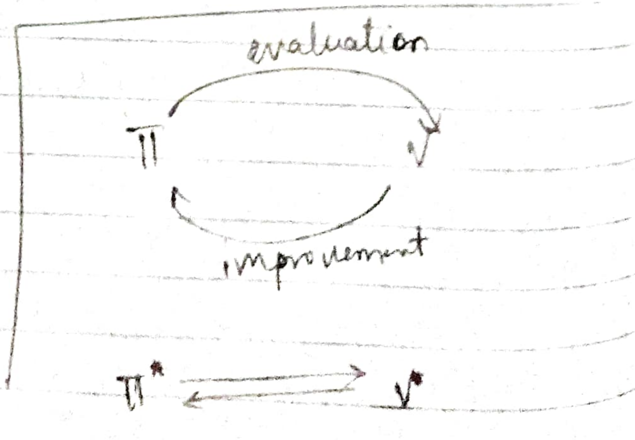
i) Update the policy to be greedy w.r.t. current q-value function

ii) This means the policy chooses the action ~~with max~~ that maximises the expected reward from the current state $q_\pi(s, a)$ for each state

$q_\pi(s, a)$

$$\pi'(a|s) = \begin{cases} 1, & a = \arg\max_a \left[\sum_{s'} \sum_r P(s', r \mid s, a)[r + \gamma V_\pi(s')]\right] \\ 0, & \text{otherwise} \end{cases}$$

iii) Control Problem



VALUE ITERATION:

- simpler than policy iteration
- only performs one step of policy evaluation in each iteration.
- ~~the optimal value funcn is defined~~
- It is an iterative algo that starts with an arbitary value functn and updates it until it converges to optimal value functn.
- optimal value functn - max. expected reward obtained from each state in the given policy:

$$V(s) = max_a [\Sigma_{s'} \Sigma_r P(s', r | s, a) [r + \gamma V(s')]]$$

s.t.,

V(s) - current estimate of the optimal value functn for state s

P(s', r | s, a) - model of the equatn.

γ - discount fator

~~V(s') -~~

	(PI) Policy Iteration	(VI) Value Iteration
Strength	i) Converges to optimal policy faster than VI in some cases	i) Converges to optimal value function faster than PI
	ii) Can handle larger class of MDP's ~~than~~, including those with stochastic policies	ii) Can handle MDP's with infinite horizon
	iii) More stable & less sensitive to discount factor	iii) Simpler & easy to implement.
Weakness	i) Can be computationally expensive	i) sensitive to the choice of discount factor
	ii) Can get stuck in local optima.	ii) cannot handle stochastic values directly
	iii) May not converge if the MDP has infinite horizon	iii) May require a large no. of iterations to converge