

Stochastic Policy - probability of selecting an action in state s . **Orderability** - Exactly one of $A \succ B$, $B \succ A$, or $A \sim B$. **Transitivity** - if $A \succ B$ and $B \succ C$, then $A \succ C$. **Continuity** - if $A \succ B \succ C$, then there exists a probability p such that $B \sim pA + (1-p)C$. **Substitutability** - if $A \sim B$, then $pA + (1-p)C \sim pB + (1-p)C$. **Monotonic Preference** - preference for A over B . **Monotonicity** - if $A \succ B$, then $pA + (1-p)C \succ pB + (1-p)C$ for all $0 \leq p \leq 1$. **Decomposability** - if $A \sim B$ and $C \sim D$, then $pA + (1-p)C \sim pB + (1-p)D$ for all $0 \leq p \leq 1$. **Utility** - The quality of being useful. **Utility Theory** - every state has a quality of being useful. **Sequential Decision Problems** - the agent's utility depends on a sequence of decisions which incorporate utilities, uncertainty, sensing, and decisions. Fully observable environment. Environment is non-deterministic. Actions aren't reliable. The outcome is stochastic. Transitions are **Markovian** - the next state depends only on the current state and action. **Markov Decision Process (MDP)** is fully observable, stochastic, sequential decision problem. Consists of states, actions, transition model, and rewards. Methods include dynamic programming, value iteration, policy iteration, and Q-learning. A solution is a **policy** - a mapping from states to actions. The goal is to find an optimal policy that maximizes expected utility over time. **Utility** is a measure of the desirability of a state or sequence of states. **Discount factor** is a value between 0 and 1 that reduces the importance of future rewards. **Optimal Policy** is a policy that maximizes expected utility from any initial state. **Finite Horizon** - after a fixed time the game is over. **Infinite Horizon** - the game continues indefinitely. **Non-stationary** - depends on time. **optimal policy** is stationary. **Proper Policy** - guaranteed to reach a final state. **Best policy** - argmax expected Q function. **Sparse** - most transition probabilities are zero. **State Space** - cartesian product of all state variables. **Bellman Update** - sets $U+1$ based on outcome of max action Bellman equation. **contraction** - two inputs produce similar results. **Policy Improvement** - calculate a new MEU policy π_{+1} using one-step lookahead with U . **Offline Algorithms** - compute the policy before execution. **Online Algorithms** - compute the policy during execution. **Value Iteration** - iteratively update utilities using Bellman equation until convergence. **Policy Iteration** - iteratively evaluate and improve policy until convergence. **Q-learning** - model-free reinforcement learning algorithm that learns the value of actions in states. **Bandit Problems** - fixed but unknown probabilities. **Exploration and Exploitation** - trade-off between trying new actions and using known actions. **Regret** - difference between actual reward and optimal reward. **Bernoulli Bandit** - each action has a fixed but unknown probability of success. **Contextual Bandit** - each action's reward depends on the current context or state.

Decision Theory - probability theory plus utility theory.

$$p(a | b) = \frac{p(b | a)p(a)}{p(b)}$$

Independence - two events A and B are independent if $P(A | B) = P(A)$. **Conditional Independence** - two events A and B are conditionally independent given C if $P(A | B, C) = P(A | C)$. **Bayes' Theorem** - a way to update probabilities based on new evidence.

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

Naive Bayes Classifier - a simple probabilistic classifier based on Bayes' theorem with strong independence assumptions.

$$P(Cause, Effect, ..Effect_n) = P(Cause) \prod_{i=1}^n P(Effect_i | Cause)$$

$$V^*(s) = \max_a \sum_{s', s} P(s' | s, a) [R(s, a, s') + \gamma V^*(s')]$$

Model based reinforcement learning - learn the transition and reward models from experience, then use them to compute the optimal policy. **Model free reinforcement learning** - learn the optimal policy directly from experience without learning. **Adaptive Dynamic Programming** - learn the model and use value iteration to compute the optimal policy. **Temporal Difference Learning** - update the utility of the current state based on the observed reward and the estimated utility of the next state. **Active Learning Agent** - needs to learn all transitions and choose the highest. Learn Q values for state-action pairs. **Catastrophic Forgetting** - neural networks forget old information when learning new information. **Policy Search** - search the space of policies directly without using value functions. **uncertainty** - partial observability, stochastic actions, unknown environment dynamics. **Laziness** - impossible to plan for every eventuality. **Theoretical Ignorance** - we don't know everything about the world. **Practical Ignorance** - we can't compute everything we need to know.