Tackling Sequential Decision Problems

What is sequential decision problems? Utility of the final result of the problem depends on a sequence of decisions of actions. These problems can be categorised into two types according to agents' abilities: **fully observable**, and **partially observable** problems. We will first look at problems with fully observable environments, then move on to partially observable environments.

1. Markov Decision Process

Assumptions in MDP:

- *States*: set of all possible states that agents may end up in.
- *Actions*: set of all actions agents can choose to take.
- *Transition model*: **assumes that we have a stochastic environment where the outcome of an action taken with a probability.** Therefore, we denote the output of the transition model as a probability: $P(s'|s, a)$, meaning the probability of being in s' when action a is taken in state s. $P(s'|s, a)$, thus, gives us a **3D table** with s', s, and a being its 3 dimensions.
- *Rewards*: R(s) must be bounded, else gives an infinite value.
- *Utility*: two types of utility functions are considered:
  - Discounted rewards: $U[s_0, s_1, s_2, ...] = R(s_0) + \gamma R(s_1) + (\gamma)^2 R(s_2) + ...$
  - Additive rewards: $U[s_0, s_1, s_2, ...] = R(s_0) + R(s_1) + R(s_2) + ...$, a special case for discounted rewards when $\gamma = 1$
- *Policy*: a policy gives direction to an agent about which actions to take in a certain state.

Remark (Utility Function)**.** *We will utilise discounted rewards for all the problems by setting up γ to a value between 1 and 0. This is because for problems with* ***infinite horizons****, where agent can go on forever in a game,*

the states sequence is infinite (which leads to an infinite utility value), but we still want to compare the utility of this infinite sequence of states with one another, thus, the utility value must be finite. Discounted rewards gives us exactly a finite utility value by apply a sum formula for a geometric series with $\gamma \leq 1$.

Remark (Policy)**.** *One important point is that a fixed sequence of actions will never be a policy for stochastic environments. This is because in stochastic environment agents may end up in any state as the action given is probability based. Thus a good policy for MDP is the one which recommends agents which actions to take at any state at any episode. This leads us to a policy which is a state based, meaning gives agents guidance based on which state the agent is in, regardless of historical actions.*

Remark (Finite & Infinite horizons)**.** *MDP problems can be further classified as finite and infinite horizons. There is a difference in between these two concepts. For finite horizons, there is a fixed deadline for an agent to complete the game, thus with this deadline set differently, the agent may* ***balance rewards and risks*** *differently given how much time left, thus leads to different courses of actions (****nonstationary****) . However, for infinite horizons, we can think of it in this way: no matter how much time passed, we are always left with infinite amount of time, thus the agent does not have to change its plan of actions/policy (****stationary****) to achieve its optimal utility. The stationarity of policy assumes the stationarity of preference over state sequence as well. This means that if* $[s_0, s_1, s_2, ...]$ *is preferred over* $[s_0, s'_1, s'_2, ...]$*, then* $[s_1, s_2, ...]$ *is preferred over* $[s'_1, s'_2, ...]$*. The reason is that the same policy/optimal action at a certain state is the same at time* $t + 1$ *and time* t*, therefore* $[s_t, ...]$ *should always preferred over* $[s'_t, ...]$.

To summarize:

Markovian Property: the probability of reaching s' from s depends only on s and not on the history of earlier states.

Markov Decision Process: a sequential decision problem for a fully observable, stochastic environment with a Markovian transition and additive rewards.

2. Looking For Optimal Policy

To know which policy is optimal, we should know how to evaluate each policy and compare them. To evaluate policy, we take the expected utility generated by each policy and order them in their natural numerical order. Note that we use 'expected' utility because sequences of actions generated by a policy corresponds to a probability.

We denote a policy as π, and an optimal policy as π^*. Thus, the expected utility of executing π starting from s is: $U^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t)]$. $\pi^*(s) = argmax U^\pi(s) = argmax \sum_{s'} P(s'|s,a)U(s')$. We use $U(s)$ to denote the utility of executing an optimal policy from s.

There are two approaches for searching π^*, namely, value iteration and policy iteration.

2.1 Value iteration

Value iteration algorithm repeatedly updates the utility value of each state by executing Bellman update equation util an equilibrium (convergence) is reached where the utility values of stated are unchanged.

Bellman equation helps us to decide the utility value for a certain state when an optimal policy is deployed:
$U(s) = R(s) + \gamma max(a) \sum_{s'} P(s'|s,a)U(s')$.

Bellman update updates the utility value for a state at time $t+1$ by consulting its neighbour's values at time t:
$U_{t+1}(s) \Leftarrow R(s) + \gamma max(a) \sum_{s'} P(s'|s,a)U_t(s')$

Value iteration will **converge** at some point for discounted rewards with $\gamma < 1$. This is because Bellman update $U_{t+1} \Leftarrow BU_t$ is a **contraction**: $||BU_{t+1} - BU_t|| \leq \gamma ||U_{t+1} - U_t||$.

Thus, a fixed point would be reached: $||BU_t - BU|| = ||BU_t - U|| \leq \gamma ||U_{t-1} - U|| \leq \gamma^2 ||U_{t-2} - U|| \leq ... \leq \gamma^t ||U_0 - U||$. (Note: $BU = U$ means U reaching an equilibrium; γ^t: contract exponentially)

Continue from $||BU_t - U|| \leq \gamma^t ||U_0 - U||$. If we can accept a maximum error ϵ between π^* and π, then $||BU_t - U|| \leq \gamma^t ||U_0 - U|| \leq \epsilon$. From the geometric sequence of discounted rewards, we know that $||U_0 - U|| \leq R_{max}/(1-\gamma)$, thus we have $\gamma^t \times R_{max}(1-\gamma) \leq \epsilon$, where t is the number of Bellman update we should go through to find the our imperfect π with error ϵ to π^*. Therefore, the termination condition for the value iteration algorithm is $||U_{t+1} - Ut|| \leq \epsilon(1-\gamma)/\gamma$.

2.2 Policy iteration

Policy iteration seems simpler than value iteration. We randomly initialize a policy and solve linear equations to get the utility value for each state. Then according to this state utility value to check whether any state action should be changed or not. If it is, we change the state action/policy according and re-calculate the utility values with the new policy. We iterate through this process util no state action is changed. Them we reach our equilibrium, and get π^*.

Policy evaluation: given a policy π, construct the state utility function according to the actions given by the policy, and solve the n linear equations with n unknown state utility values: $U(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s))U(s')$.

Policy improvement: using one step look ahead (i.e. check whether the action is the optimal action by using the neighbour state utility values) to calculate a new policy based on the immediate (one step) neighbours' state utility value.

Remark (Linear Equation)**.** *Notice that the equations in policy iteration are* ***linear****. This is because given the policy, the utility value of each state can be calculated. There is no* ***max operator****, which is unlike value iteration where we do not know the policy thus need to use max operator to find the policy. For n states, policy iteration evaluation can solve it in* $O(n^3)$*.*

Remark (Approximation)**.** ***Modified policy iteration*** *is used to iteration k times instead of to convergence.* ***Asynchronous policy iteration*** *is to speed up the algorithm by executing iteration improvement and iteration evaluation for a subset of states.*

3. Online Search

Why do we need online search? because of the curse of dimensionality. For n binary state variables, we have 2^n states to look at in both value and policy iteration. One way to handle this dimensionality (too many states) is to utilize **online search with sampling**.

At every step, if the state **does not have a policy to choose an action**, construct a search tree with the root node as the current state to decide the optimal action to take at the current state:

- Fix the depth of the search tree as D
- $|A|$: the number of action nodes
- $|S|$: the number of observation nodes / chance nodes which gives the number of next state we can transition to
- Tree size: $A^D S^D$

Value of the root: initialize utility values/reward at the leaf nodes, walk way back. At **chance node**, compute the **expected utility**. At **action node**, take the **maximum** from the children.

Conclusion: the above does not solve the curse of dimensionality but makes the situation worse: the search space increases from S to $A^D S^D$.

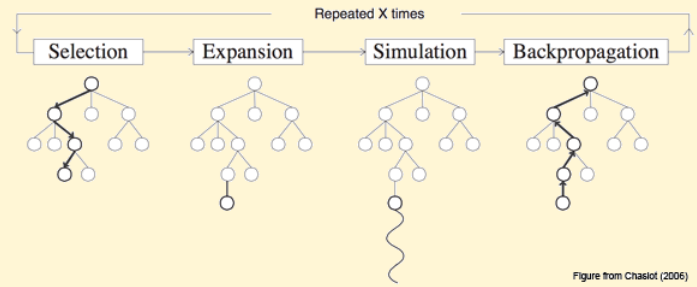
Solution: **sparse sampling**, sample k states at the observation nodes, thus the search space size is reduced to $A^D k^D$, where k does not depend on state space size $|S|$, but how accurate we want our approximation to be (same as D).

Summary: sparse sampling solves the curse of dimensionality, but not the curse of history, exponential with the search depth D.

3.1 Monte Carlo Tree Search

MCTS is a form of online search because it corporates online search algorithm in its procedure. Its application extends beyond games, and MCTS can theoretically be applied to any domain that can be described in terms of {state, action} pairs and simulation used to forecast outcomes.

Basic algorithm:



1. Selection
Starting at root node *R*, recursively select optimal child nodes (explained below) until a leaf node *L* is reached.

2. Expansion
If *L* is a not a terminal node (i.e. it does not end the game) then create one or more child nodes and select one *C*.

3. Simulation
Run a simulated playout from *C* until a result is achieved.

4. Backpropagation
Update the current move sequence with the simulation result.

- Each node must contain two important pieces of information:
  1. an estimated value based on simulation results
  2. the number of times it has been visited
- In its simplest and most memory efficient implementation, MCTS will add one child node per iteration. Note, however, that it may be beneficial to add more than one child node per iteration depending on the application.
- Node selection during tree descent is achieved by choosing the node that maximises the **Upper Confidence Bounds (UCB)** formula:

$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

exploitation term + exploration term

where v_i is the estimated value/average return of the node, n_i is the number of the times the node has been visited and N is the total number of times that its parent has been visited. C is a tunable bias parameter.

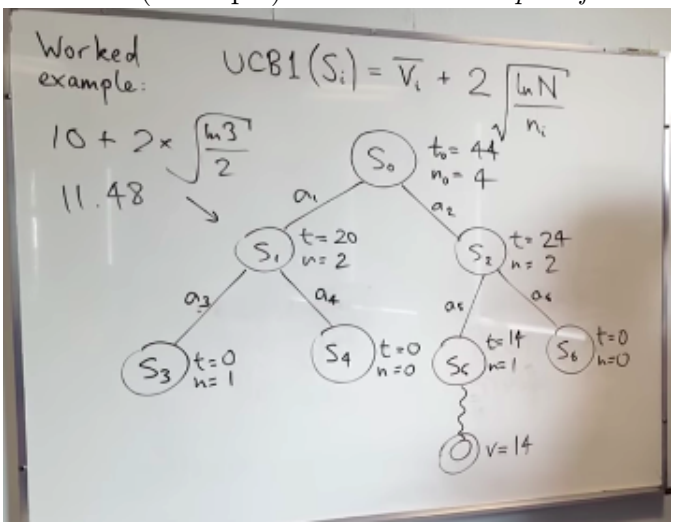
Remark (Anytime Policy)**.** *MCTS deploys an* ***anytime policy****: when time is up, use the action that looks the best at the root at that time.*

Remark (Exploitation vs Exploration)**.** *The UCB formula balances the exploitation of known rewards with the exploration of relatively unvisited nodes to encourage their exercise. Reward estimates are based on random simulations, so nodes must be visited a number of times before these estimates become reliable; MCTS estimates will typically be unreliable at the start of a search but converge to more reliable estimates given sufficient time and perfect estimates given infinite time.*

Remark (Benefits)**.** *Advantages of using MCTS:*

- *Aheuristic: MCTS does not require any strategic or tactical knowledge about the given domain to make reasonable decisions. The algorithm can function effectively with no knowledge of a game apart from its legal moves and end conditions; this means that a single MCTS implementation can be reused for a number of games with little modification, and makes MCTS a potential boon for general game playing.*
- *Asymmetric: MCTS performs asymmetric tree growth that adapts to the topology of the search space. The algorithm visits more interesting nodes more often, and focusses its search time in more relevant parts of the tree. (Trade off between exploration and exploitation).*

Remark (Example)**.** *A worked example of MCTS:*



3.1.1 Monte Carlo Tree Search With MDP

By applying MDP in MCTS, we are adding observation nodes/chance nodes into the search tree. Agents make actions at the action nodes, and environment selects a next state with certain probability distribution at the observation nodes. Thus, action a is selected by balancing the exploration and exploitation.

- $r_t(n)$: the return value of t_{th} trial at node n with state s and next node n' is $R(s) + \gamma r_t(n')$, so $r_t(n) = R(s) + \gamma r_t(n')$
- $\overline{Q}(n, a)$: Q-function/action-value function, the average return of all the trails at node n and action a
- $\overline{V}(n)$: the estimated value at node n is the average return at node n over all the trials, thus $\overline{V}(n) = max\overline{Q}(n, a)$ wrt action a

3.2 Upper Confidence Tree

UCT is built with MCTS, with the following functions to select actions at node n:

$$\pi_{UCT}(n) = \arg\max_a \hat{Q}(n, a) + c\sqrt{\frac{\ln(N(n))}{N(n, a)}}$$

UCT will eventually converge to the optimal policy with enough trials. The worst case can be very bad: $\Omega(exp(exp(...exp(1)...)))$ with $D - 1$ times exp.

3.3 Alpha Go

Alpha Go is a zero-sum game instead of a MDP, but it utilizes MCTS and deep neural network.

- Deep neural network with two heads:
  - Value head: output a real value estimate of the value function
  - Policy head: output a vector where each component represents the probability that the policy will play the game

- MCTS: is used to select actions
  - Variant of UCT that exploits policy head output of neural network $P(s, a)$.

$$\pi_{UCT}(s) = \arg\max_a Q(s, a) + cP(s, a)\frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}$$

  - When the leaf node is reached, the value head of the neural network is used to estimate the state instead of doing a simulation/roll-out.
- Since this is a zero-sum turn taking game instead of a MDP, the search alternates between selecting action that maximizes when it is first player's turn and action that minimizes (multiply estimate by -1 then maximize) for second player's turn. At termination, reward $+1$ for first player win and -1 for second player win.

Additional Notes:

Decision Making Under Uncertainty

What makes **planning** hard is that we have to consider utilities of **long sequences of actions** to reach a certain goal. **Uncertainty** makes decision making for each decision hard. We will use **decision theory** to make rational decisions under uncertainty base on **preferences** over the underlying outcome states. Therefore, it is necessary for us to first look at what is decision theory.

Decision Theory

Decision theory is a calculus for decision-making under uncertainty. To utilize this theory, we will make the following assumptions:

- A set of possible **outcomes** in the domain of interest
- A set of **preferences** (of the agent) over the possible outcomes
- The **probabilities** of different outcomes actually happening in various situations

Then, we will model uncertain outcomes as lotteries, e.g. $[p, A; 1-p, B]$. To see how to make decision among different lotteries, we need to determine the expected utility of each lottery, and choose the **maximum** among them.

Principle of maximum expected utility (MEU): a rational agent should choose the action that maximizes its expected utility.

- $P(RESULT(a) = s'|a, e)$: the probability of outcome s' given that action a and evidence e
- $U(s)$: utility function assigning a number to indicate desirability of state s
- $EU(a|e) = \sum_{s'} P(RESULT(a) = s'|a, e)U(s')$: the expected utility of action a given evidence e.

MEU is based on the **utility theory**, which contains the following **six axioms**. If your agent's preferences meet the requirements in the axioms, then decision theory/MEU will tell you how to make your decisions. If you disagree with the axioms, then you have to find another way of choosing actions.

Assume agent has preferences:

- $A \succ B$: agent prefers A over B
- $A \sim B$: agent is indifferent between A and B
- $A \succsim B$: agent prefers A over B or is indifferent

1. **Orderability:** Given any two lotteries, a rational agent must rate with one of three possibilities – prefer one, the other, or equally.
   Exactly one of $(A \succ B)$, $(B \succ A)$, or $(A \sim B)$ holds.
2. **Transitivity:** If the agent prefers A to B and B to C, then the agent must prefer A to C

$$(A \succ B) \cap (B \succ C) \Rightarrow (A \succ C).$$

3. **Continuity:** If B is between A and C in preference, there must be a p, such that the agent is indifferent to getting A with prob p and B with prob $(1 - p)$.

$$A \succ B \succ C \Rightarrow \exists p \; [p, A; 1 - p, C] \sim B.$$

- In the first lottery, you get your best outcome, A, with probability p, and your worst outcome, C, with probability 1-p.
- In the second lottery, you get the outcome, B, with certainty.
- So, the idea is that, by adjusting p, you can make these lotteries equivalently attractive, for any combination of A, B, and C.

4. **Substitutability:** If an agent is indifferent between two lotteries A and B, then the agent is indifferent to two complex lotteries that are the same, except B is substituted for A.

$$A \sim B \Rightarrow [p, A; 1 - p, C] \sim [p, B; 1 - p, C].$$

Also hold if $\succ$ substituted for $\sim$.

- If you prefer A to B, then given two lotteries that are exactly the same, except that one has A in a particular position and the other has B, you should prefer the lottery that contains A.

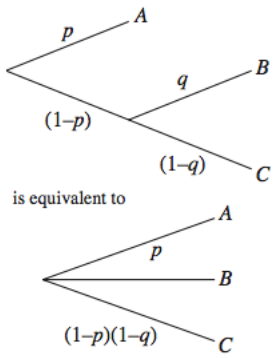
5. **Monotonicity:** Suppose two lotteries have the same two possible outcomes, A and B. If an agent prefers A to B, the agent must prefer the lottery that has higher probability for A, and vice versa.

$$A \succ B \Rightarrow (p > q) \Leftrightarrow [p, A; 1 - p, B] \succ [q, A; 1 - q, B].$$

- Monotonicity says that if you prefer A to B, and if p is greater than q, then you should prefer a lottery that gives A over B with higher probability.

6. **Decomposibility:** Compound lotteries can be reduced to simpler ones using the laws of probability.

$$[p, A; 1-p[q, B, 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C].$$



- It, in some sense, defines compound lotteries.
- It says that a two-stage lottery, where in the first stage you get A with probability p, and in the second stage you get B withprobability q and C with probability 1-q is equivalent to a single-stage lottery with three possible outcomes: A with probability p, B with probability (1-p)*q, and C with probability (1-p)*(1-q).

Utility theory is really the axioms about preferences, nothing about the utility function ($U(s)$). However, from the utility theory, the following consequences about $U(s)$ can be derived:

- **Existence of Utility Function**: There exists a realvalued function U such that if you prefer A to B, then U(A) is greater than U(B), and if you prefer A and B equally, then U(A) is equal to U(B).

  $$U(A) > U(B) \Leftrightarrow A \succ B$$
  $$U(A) = U(B) \Leftrightarrow A \sim B.$$

  - Basically, this says that all possible outcomes can be mapped onto a single utility scale, and we can work directly with utilities rather than collections of preferences.

- **Expected Utility of a Lottery**: The utility of a lottery is the expected utility of the outcomes.

  $$U([p_1, S_i; \ldots; p_n, S_n]) = \sum_i p_i U(S_i).$$

An agent can act **rationally** only by choosing an action that **maximises the expected utility**. An agent's behaviour does not change of the utility function is transformed an **affine transformation** $U'(s) = aU(s) + b$ with $a > 0$.

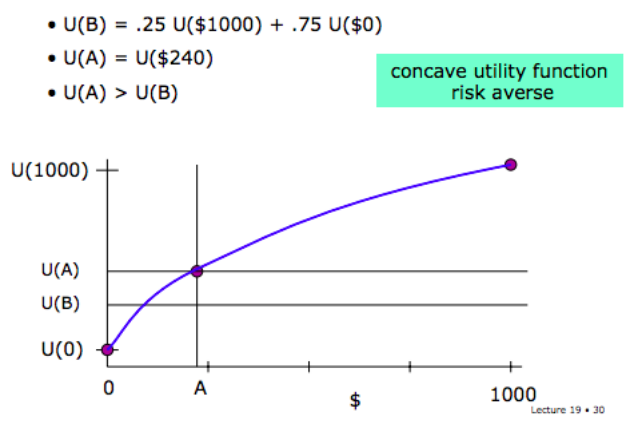
In a deterministic environment, only the axiom of orderability and transitivity are required. The other axioms all involves lotteries.

Utility Assessment

- **Preference elicitation**: the process of presenting choices to agents and working out the utility function from observed responses.
- Fixing utilities of two outcomes sufficient to establish scale, usually fix for worst outcome $u_\perp$ and best outcome $u_\top$. For **normalized utilities**, fix values are $u_\perp = 0$ and $u_\top = 1$.
- We can **assess utility for prize S** by asking agent to choose between prize and a standard lottery $[p, u_\top; 1-p, u_\perp]$. By adjusting p until agent is indifferent between S and lottery, then we can get the utility value for price S.

Utility of Money

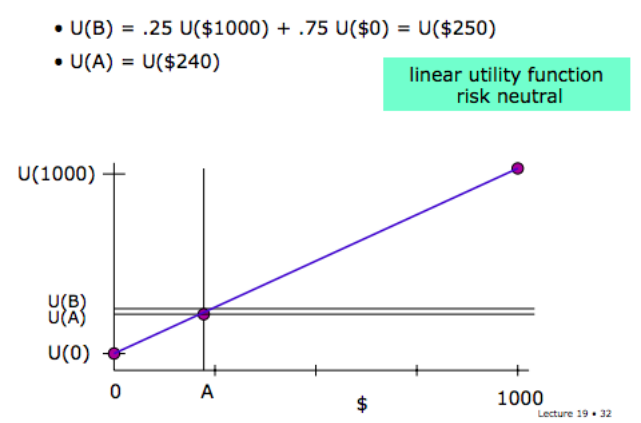
Risk Averse:



- This describes a person who would in general prefers a smaller amount of money for sure, rather than lotteries with a larger expected amount of money (but not expected utility). Most people are risk averse in this way.

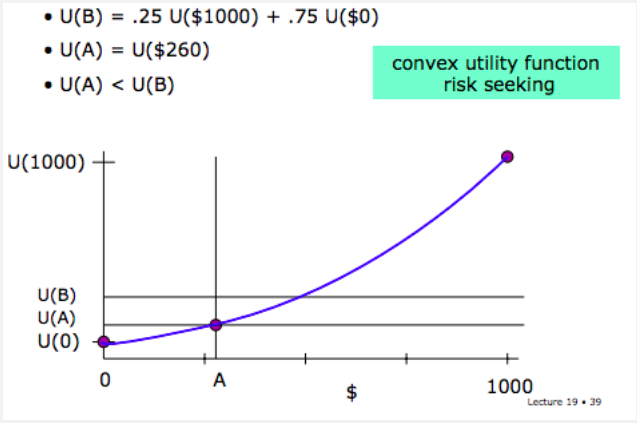
Remark. *It's important to remember that decision theory applies in any case. It's not necessary to have a utility curve of any particular shape (in fact, you could even prefer less money to more!) in order for decision theory to apply to you. You just have to agree to the 6 axioms.*

Risk Neutral:



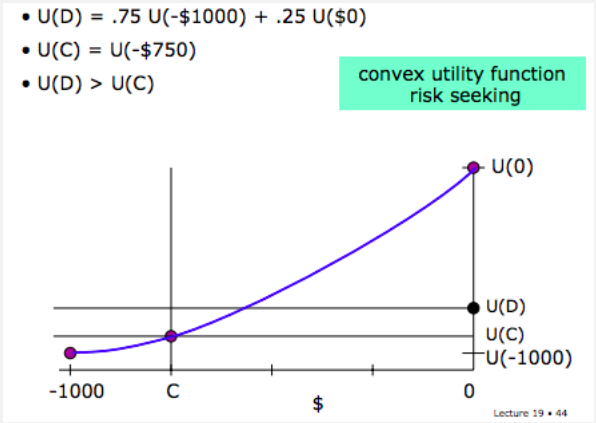
- If you are 'risk neutral', then your utility function over money is linear. In that case, your expected utility for a lottery is exactly proportional to the expected amount of money you'll make. So, in our example, the utility of B would be exactly equal to the utility of $250. And so it would be greater than the utility of A, but not a lot.

Risk Seeking:



- It's possible to argue that people play the lottery for a variety of reasons, including the excitement of the game, etc. But here we've argued that there are utility functions under which it's completely rational to play the lottery, for monetary concerns alone. You could think of this utility function applying, in particular, to people who are currently in very bad circumstances. For such a person, the prospect of winning $10,000 or more might be so dramatically better than their current circumstances, that even though they're almost certain to lose, it's worth $1 to them to have a chance at a great outcome.

Risk Seeking in Losses

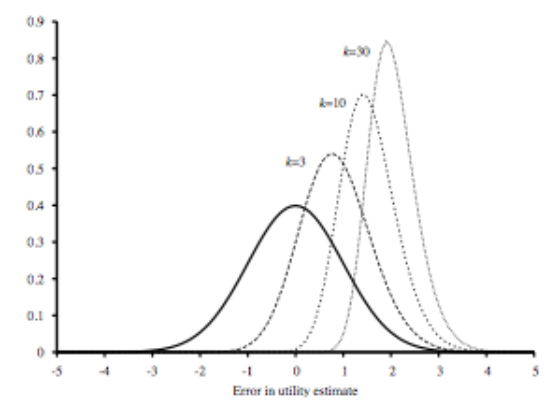


- In any real lottery, the expected amount of money you'll win is always less than the price.
- we can see that these preferences imply a **convex utility function**, which induces risk seeking behavior. It is generally found that people are risk seeking in the domain of losses. Or, at least, in the domain of small losses.
- An interesting case to consider here is that of **insurance**. You can think of insurance as accepting a small guaranteed loss (the insurance premium) rather than accepting a lottery (not risk seeking) in which, with a very small chance, a terrible thing happens to you. So, perhaps, in the domain of large losses (in the case of insurance), people's utility functions tend to change curvature and become concave again.

Optimizer's curse

As our model is the oversimplified version of the reality, we are really working with estimated expected utility value instead if the true expected utility value. This is mainly because, 1) we may not know enough about the reality; 2) the computation of the true expected utility is too difficult. Therefore, the real outcome is usually **worse** than our computation. This is because we are **choosing actions with the highest utility estimate**, obviously favouring the optimistic estimates, and this is **the source of the bias**.

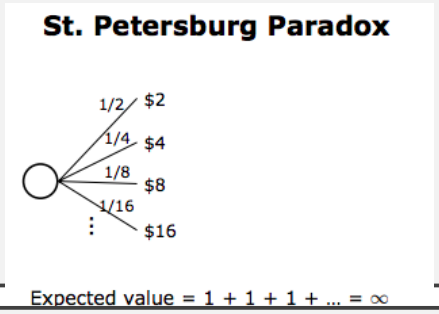
With **more choices** (k indicated the number of choices in the following diagram), extremely **optimistic** estimates are **more** likely to arise, thus the greater the disappointment. It can even be the case that what appears to be the best choice may not be, if the variance in the utility estimate is high.



We can **avoid the curse** by using an explicit probability model $P(\overline{EU}|EU)$ of the error in the utility estimates. Given this model and a prior $P(EU)$ on what we might reasonably expect the utility to be, we treat the utility estimate, once obtained, as evidence and compute the **posterior distribution for the utility** using **Bayes' rule**.

Human irrationality

- *Allais paradox:* People are given choices between lotteries A and B and between C and D with the following prizes:
  - A: 80% chance of \$4000 C: 20% chance of \$4000
  - B: 100% chance of \$3000 D: 25% chance of \$3000
- People tend to prefer B over A (taking sure thing with lower EMV) and C over D (taking higher EMV).



It was this paradox that drove Bernoulli to think about concave, risk averse, utility curves, which you can use to show that although the game has an infinite expected dollar value, it will only have a finite expected utility for a risk averse person.

- Certainty effect: people are strongly attracted to gains that are certain
- Ambiguity aversion: prefer known probability to unknown ones
- Framing effect: exact wording of choices have big impact on choices
- Anchoring effect: people prefer to make relate utility judgements to absolute ones. They rely too much on initial information

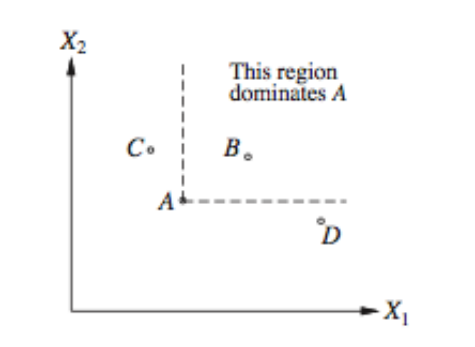
Multiattribute Utility

1. Dominance

When outcomes characterized by two or more attributes:
multiattribute utility theory

- Attributes $\mathbf{X} = X_1, \ldots, X_n$
- Assignment $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$
- Assume ordering of values for x_i, higher value of attribute implies higher utility

- **Strict dominance**: If an option is of lower value on all attributes than some other option, can be eliminated
  - Example:



  - When outcome uncertain, can also eliminate A when all possible outcomes from B, D strictly dominates all possible outcomes for A

- **Stochastic dominance**: a more useful generalization, can also eliminate an option if it is stochastically dominated on all attribute by another option

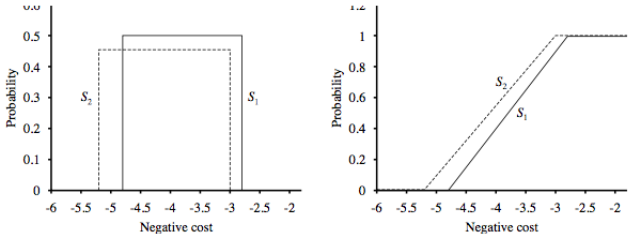
A_1 stochastically dominates A_2 on X if

$$\forall x, \int_{-\infty}^{x} p_1(x')dx' \leq \int_{-\infty}^{x} p_2(x')dx',$$

cumulative distr for option A_1 is no more than for option A_2.

  - Expected utility of A1 at least as high as A2.
  - Example:

Example: density function on left, cumulative distr on right.



  - S_1 dominates S_2

2. Preference structure

Suppose we have n attributes, each of which has d distinct possible values. To specify the complete utility function $U(x_1, x_2, ..., x_n)$ we need d^n values in the **worst case**. This worst case corresponds to a situation in which the agent's preferences have no regularity at all. **Multi-attributes utility theory** is based on the supposition that the preferences of typical agents have much more structure than that. The **basic approach** is to **identify regularities** in the preference behaviour we would expect to see and to use what are called **representation theorem** to show that an agent with a certain kind of preference structure gas a utility function: $U(x_1, x_2, ..., x_n) = F[f_1(x_1), ..., f_n(x_n)]$, where F is, we hope, a simple function such as addition.

- **Preference without uncertainty**
  - This is a deterministic case as there is no uncertainty
  - The basic regularity arises in deterministic preference structure is called **Preference independent (PI)**.
  - Preference independent: if X_1 is preference independent of X_2 if $(x_1, {x_2}^0) > (x_1^{'}, x_2^0)$ implies that $(x_1, x_2) > (x_1^{'}, x_2)$ for all x_2
    - Example: Airport(Noise, Cost, Death), one may propose that Noise and Cost are preferentially independent of Death.

- Preference without uncertainty (continue)
  - **Mutual preferential independent (MPI)**: whereas each attribute may be important, it does not affect the way in which one trades off the other attributes against each other.
    - Airport(Noise, Cost, Death): (Noise, Cost) is preferentially independent of Death; (Cost, Death) is preferentially independent of Noise; (Noise, Death) s preferentially independent of Cost.
  - MPI implies **addictive preference function**:

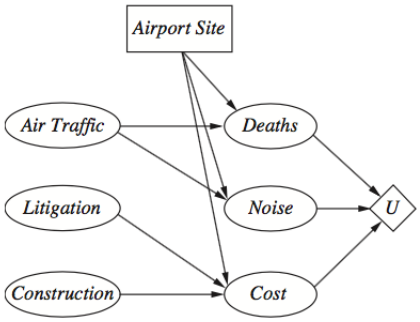
$$V(x_1, \ldots, x_n) = \sum_{i=1}^{n} V_i(x_i).$$

    - V_i is a value function referring only to the attribute X_i
    - For n attributes, assessing an additive value function requires assessing n seperate one-dimensional value functions rather than one n-dimensional function, typically, this represents an **exponential reduction** in the number of preference experiments that are needed.
    - Example: V(noise, cost, death) = -noise $\times 10^4$ - cost - death $\times 10^2$
  - Additive preference function is often a **good approximation** even when MPI does not hold. This is espcially true when the violations of MPI occur in portions of the attributes. MPI is violated when attributes depends strongly on each other.

- Preferences with uncertainty
  - When uncertainty is present, we need to consider the structure of preferences between lotteries and to understand the resulting properties of utility functions, rather than just value functions.
  - **Utility independent**: A set of attributes X is utility independent of Y if preferences between lotteries on X are independent of values of Y.
  - **Mutually utility independent (MUI)**: A set of attributes is mutually utility independent if each of its subset is utility independent of the remainder.
  - MUI implies a **multiplicative utility function**: $U(x_1, x_2, x_3) = k_1U_1(x_1) + k_2U_2(x_2) + k_3U_3(x_3) + k_1k_2U_1(x_1)U_2(x_2) + k_1k_3U_1(x_1)U_3(x_3) + k_3k_2U_3(x_3)U_2(x_2) + k_1k_2k_3U_1(x_1)U_2(x_2)U_3(x_3)$
    - The equation above contains just three single-attribute utility functions and three constants. In general, an **n-attribute problem** exhibiting MUI can be modelled using **n single-attribute utilities** and **n constant**. Each n single-attribute utility can be developed independently with the values under the attribute.

Decision Network

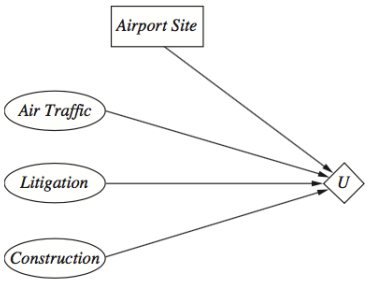
Influence diagrams or **decision networks** combine Bayesian networks with additional nodes to denote actions and utilities.



- **Chance nodes** (ovals) represent random variables, like in Bayesian network. In the example, *Deaths*, *Noise*, *Cost* are chance nodes. Represent conditional distribution given its parents, which can be both decision as well as chance nodes.
- **Decision nodes** (rectangles) represent points where decision maker has choice of actions. In example, this is the choice of airport site.
- **Utility nodes** (diamonds) represent the utility function. Its parents are all the variables that directly affect the utility.

A simplified form allows the action node to be directly connected to the utility node.

- Utility node is associated with **action-utility function**, or **Q-function**, representing expected utility associated with each action.



Evaluating Decision Networks

1. Set evidence variables to the observed states.
2. For each possible value of the decision node:
   (a) Set the decision node to that value.
   (b) Calculate posterior probability of parents of utility node, using probabilistic inference algorithm.
   (c) Calculate resulting utility for the action.
3. Return action with highest utility.

Value of Information

Assume exact evidence can be obtained about variable E_j (the evidence once observed, is certain), compute **value of perfect information**:

- Given current evidence e, expected utility with current best action α: $EU(\alpha|e) = \arg\max_a \sum_{s'} P(RESULT(\alpha) = s'|e, \alpha)U(s')$
- Value of best new action after $E_j = e_j$ is obtained: $EU(\alpha|e_i, e_j) = \arg\max_a \sum_{s'} P(RESULT(\alpha) = s'|e_i, e_j, \alpha)U(s')$
- Variable E_j can take multiple values (for example, observed getting the land with gold, or not getting the land with gold), so must take the expected value: $VPI(E_j) = \sum_k P(e_{jk}|e)EU(\alpha_{jk}|e, e_{jk}) - EU(\alpha|e)$

Properties of value of information:

- Expected value of information in **non-negative**: $\forall e, E_j, VPI_e(E_j) \geq 0$
- VPI is **not additive**, but **order independent**: $VPI_e(E_j, E_k) = VPI_e(E_j) + VPI_{e,E_j}(E_k) = VPI_e(E_k) + VPI_{e,E_k}(E_j)$