Making Decisions[1]

Dmitry Adamskiy, David Barber

University College London

[1] These slides accompany the book *Bayesian Reasoning and Machine Learning*. The book and demos can be downloaded from `www.cs.ucl.ac.uk/staff/D.Barber/brml`. Feedback and corrections are also available on the site. Feel free to adapt these slides for your own purposes, but please include a link the above website.

Expected Utility

You are asked if you wish to take a bet on the outcome of tossing a fair coin. If you bet and win, you gain $£100$. If you bet and lose, you lose $£200$. If you don't bet, the cost to you is zero.

$$U(\text{win, bet}) = 100 \quad U(\text{lose, bet}) = -200$$
$$U(\text{win, no bet}) = 0 \quad U(\text{lose, no bet}) = 0$$

Our expected winnings/losses are:

$$\begin{aligned} U(\text{bet}) &= p(\text{win}) \times U(\text{win, bet}) + p(\text{lose}) \times U(\text{lose, bet}) \\ &= 0.5 \times 100 - 0.5 \times 200 = -50 \end{aligned}$$

$$U(\text{no bet}) = 0$$

Based on taking the decision which maximises expected utility, we would therefore be advised not to bet.

Utility of Money

You have $£1,000,000$ in your bank account. You are asked if you would like to participate in a fair coin tossing bet in which, if you win, your bank account will become $£1,000,000,000$. However, if you lose, your bank account will contain only $£1000$. Should you take the bet?

$$U(\text{bet}) = 0.5 \times 1,000,000,000 + 0.5 \times 1000 = 500,000,500.00$$

$$U(\text{no bet}) = 1,000,000$$

Based on expected utility, we are therefore advised to take the bet.

Utility of Money

- In reality few people who are millionaires are likely to be willing to risk losing almost everything in order to become a billionaire.
- This means that the subjective utility of money is not simply the quantity of money.

Being 'better off' (not in time, but in relation!)

My personal belief is that the 'subjective utility of money' is the probability that your income exceeds someone else's

$$p(x > y) = \int_{y<x} p(y) = cumpdf(x)$$

where $p(y)$ is the distribution of incomes in your comparison population.

Decision Trees

Consider the decision problem as to whether or not to go ahead with a fund-raising garden party. If we go ahead with the party and it subsequently rains, then we will lose money (since very few people will show up). If we don't go ahead with the party and it doesn't rain we're free to go and do something else fun.

$$p(Rain = \text{rain}) = 0.6,\ p(Rain = \text{no rain}) = 0.4$$

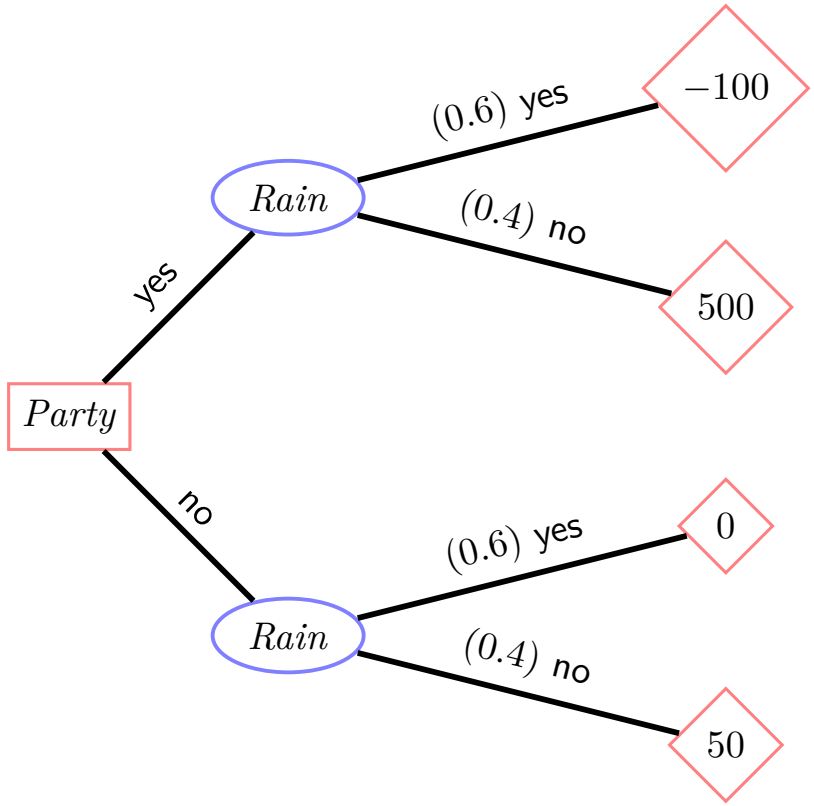
$$U\,(\text{party}, \text{rain}) = -100, \quad U\,(\text{party}, \text{no rain}) = 500$$
$$U\,(\text{no party}, \text{rain}) = 0, \quad U\,(\text{no party}, \text{no rain}) = 50$$

Should we go ahead with the party? Since we don't know what will actually happen to the weather, we compute the expected utility of each decision:

$$U\,(\text{party}) = \sum_{Rain} U(\text{party}, Rain)p(Rain) = -100 \times 0.6 + 500 \times 0.4 = 140$$
$$U\,(\text{no party}) = \sum_{Rain} U(\text{no party}, Rain)p(Rain) = 0 \times 0.6 + 50 \times 0.4 = 20$$

Based on expected utility, we are therefore advised to go ahead with the party.

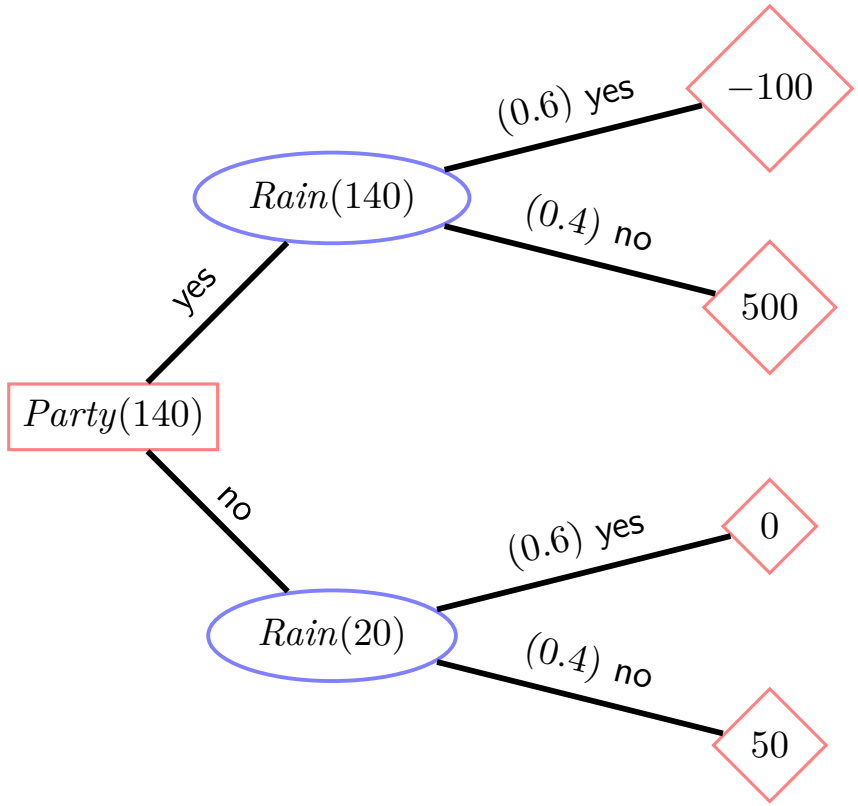
Decision Trees



A decision tree containing chance nodes (denoted with ovals), decision nodes (denoted with squares) and utility nodes (denoted with diamonds). A DT is not a belief network. A DT is an explicit enumeration of the possible choices that can be made, beginning with the leftmost decision node, with probabilities on the links out of 'chance' nodes.

Solving a Decision Tree

Working backwards from the leaves: A chance parent is the expectation of its children. A decision parent is the max of its children:



$$-100 \times 0.6 + 500 \times 0.4 = 140$$

$$0 \times 0.6 + 50 \times 0.4 = 20$$

$$\max(140, 20) = 140$$

The Party-Friend problem

If we decide not to go ahead with the party, we will consider going to visit a friend. In making the decision not to go ahead with the party we have utilities

$$U_{party}\left(\text{no party}, \text{rain}\right) = 0, \quad U_{party}\left(\text{no party}, \text{no rain}\right) = 50$$

$$U_{party}\left(\text{party}, \text{rain}\right) = -100, \quad U_{party}\left(\text{party}, \text{no rain}\right) = 500$$

$$p(Rain = \text{rain}) = 0.6, \quad p(Rain = \text{no rain}) = 0.4$$

The probability that the friend is in depends on the weather according to

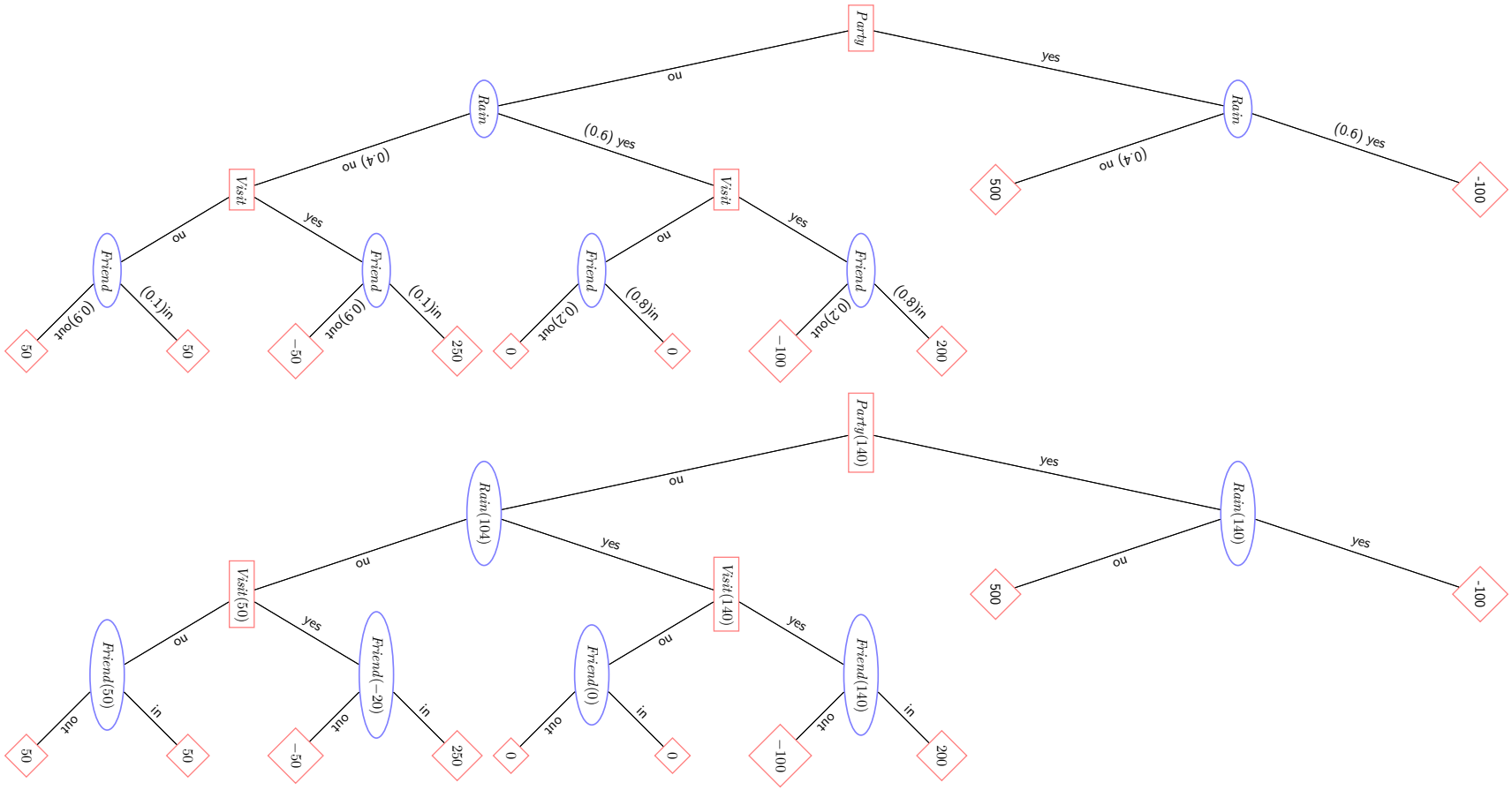
$$p(Friend = \text{in}|\text{rain}) = 0.8, \quad p(Friend = \text{in}|\text{no rain}) = 0.1,$$

The other probabilities are determined by normalisation. We additionally have

$$U_{visit}\left(\text{friend in}, \text{visit}\right) = 200, \quad U_{visit}\left(\text{friend out}, \text{visit}\right) = -100$$

with the remaining utilities zero. The two sets of utilities add up so that the overall utility of any decision sequence is $U_{party} + U_{visit}$.

A bigger DT

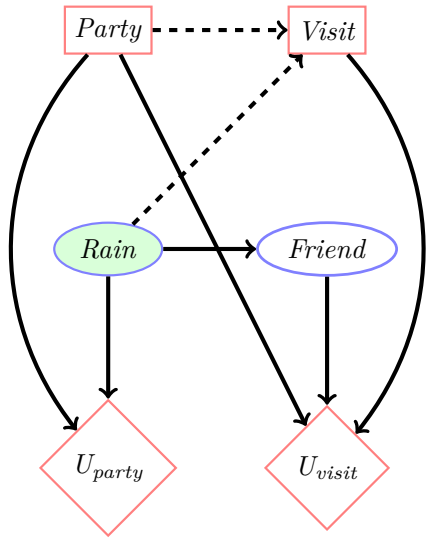


DTs can get very unwieldily even for simple problems. Need a more compact graphical description.

Influence Diagrams

The optimal expected utility for the Party-Visit example is given by summing over un-revealed variables and optimising over future decisions:

$$\max_{Party} \sum_{Rain} p(Rain) \max_{Visit} \sum_{Friend} p(Friend|Rain)$$
$$\times \left[U_{party}(Party, Rain) + U_{visit}(Visit, Friend) \mathbb{I}\left[Party = \text{no}\right] \right]$$

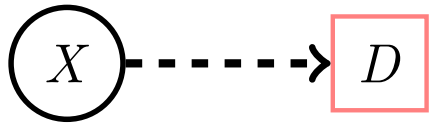


The partial ordering is $Party^* \prec Rain \prec Visit^* \prec Friend$.

NB: Confusingly, an ID looks similar to a DT. However, a DT is analogous to a state-transition diagram. An ID is analogous to a belief network.

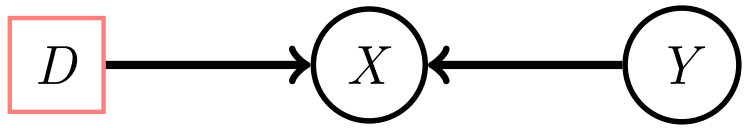
Influence Diagrams

Information Links An information link from a random variable into a decision node



indicates that the state of the variable X will be known before decision D is taken. Information links from another decision node d in to D similarly indicate that decision d is known before decision D is taken. We use a dashed link to denote that decision D is not functionally related to its parents.

Random Variables Random variables may depend on the states of parental random variables (as in belief networks), but also decision node states:



As decisions are taken, the states of some random variables will be revealed. To emphasise this we typically shade a node to denote that its state will be revealed during the sequential decision process.

Utilities A utility node is a deterministic function of its parents. The parents can be either random variables or decision nodes.

Partial Ordering

- Begin by writing those variables $\mathcal{X}_0$ whose states are known (evidential variables) before the first decision D_1.
- Then find that set of variables $\mathcal{X}_1$ whose states are revealed before the second decision D_2.
- Subsequently the set of variables $\mathcal{X}_t$ is revealed before decision D_{t+1}.
- The remaining fully-unobserved variables are placed at the end of the ordering:

$$\mathcal{X}_0 \prec D_1 \prec \mathcal{X}_1 \prec D_2, \ldots, \prec \mathcal{X}_{n-1} \prec D_n \prec \mathcal{X}_n$$

with $\mathcal{X}_k$ being the variables revealed between decision D_k and D_{k+1}.

'Partial' refers to the fact that there is no order implied amongst the variables within the set $\mathcal{X}_n$. For notational clarity, we may indicate decision variables with * to reinforce that we maximise over these variables, and sum over the non-starred variables. Where the sets are empty we may omit writing them.

Information Links

No forgetting assumption

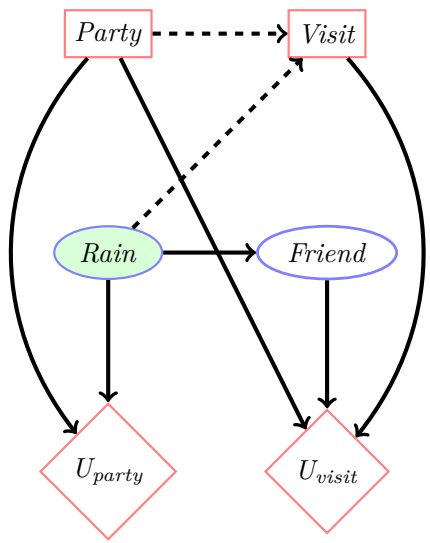
- An information link specifies explicitly which quantities are known before that decision is taken.
- We also implicitly assume that all past decisions and revealed variables are available at the current decision.
- (If we were to include all such information links, IDs would get potentially rather messy.)

Fundamental links

We call an information link fundamental if its removal would alter the partial ordering.

Causal Consistency

- For an Influence Diagram to be consistent a current decision cannot affect the past.
- This means that any random variable descendants of a decision D in the ID must come later in the partial ordering.
- Assuming no-forgetting, this means that for any valid ID there must be a directed path connecting all decisions. This can be a useful check on the consistency of an ID.
- An ID defines a partial ordering of the nodes.



Solving an ID

- The utility is the sum over all partial utilities
- We are given a partial ordering:

$$\mathcal{X}_0 \prec D_1 \prec \mathcal{X}_1 \prec D_2, \ldots, \prec \mathcal{X}_{n-1} \prec D_n \prec \mathcal{X}_n$$

with $\mathcal{X}_k$ being the variables revealed between decision D_k and D_{k+1}.

A simple algorithm

- Start from the final variables in the partial ordering ($\mathcal{X}_n$)
- Perform the sum/max over the current variables
- Move to the next variables in the ordering and repeat the sum/max *etc.*

Other algorithms

- We'll look at special cases for which the operations can be carried out efficiently (*e.g.* Markov Decision Process).
- There are also general purpose approaches to perform inference based on extending the Junction Tree formalism.

Should I do a PhD?

Consider a decision whether or not to do PhD as part of our education (E). Taking a PhD incurs costs, U_C both in terms of fees, but also in terms of lost income. However, if we have a PhD, we are more likely to win a Nobel Prize (P), which would certainly be likely to boost our Income (I), subsequently benefitting our finances (U_B). The ordering is (excluding empty sets)

$$E^* \prec \{I, P\}$$

$\mathrm{dom}(E) = (\text{do PhD}, \text{no PhD})$, $\mathrm{dom}(I) = (\text{low}, \text{average}, \text{high})$,
$\mathrm{dom}(P) = (\text{prize}, \text{no prize})$.

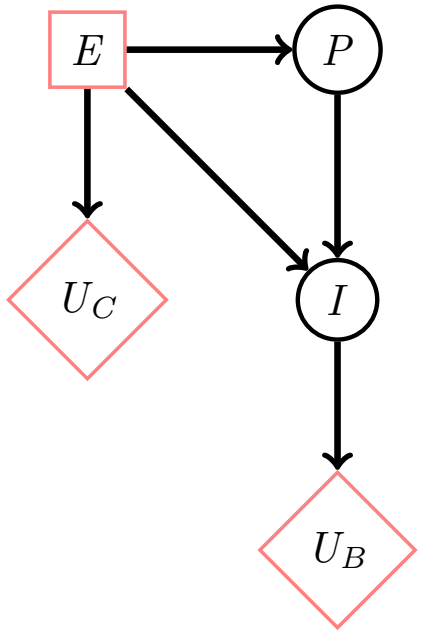
$$p(\text{win Nobel prize}|\text{no PhD}) = 0.0000001 \qquad p(\text{win Nobel prize}|\text{do PhD}) = 0.001$$

$p(\text{low}\|\text{do PhD, no prize}) = 0.1$	$p(\text{average}\|\text{do PhD, no prize}) = 0.5$	$p(\text{high}\|\text{do PhD, no prize}) = 0.4$
$p(\text{low}\|\text{no PhD, no prize}) = 0.2$	$p(\text{average}\|\text{no PhD, no prize}) = 0.6$	$p(\text{high}\|\text{no PhD, no prize}) = 0.2$
$p(\text{low}\|\text{do PhD, prize}) = 0.01$	$p(\text{average}\|\text{do PhD, prize}) = 0.04$	$p(\text{high}\|\text{do PhD, prize}) = 0.95$
$p(\text{low}\|\text{no PhD, prize}) = 0.01$	$p(\text{average}\|\text{no PhD, prize}) = 0.04$	$p(\text{high}\|\text{no PhD, prize}) = 0.95$

The utilities are

$$U_C(\text{do PhD}) = -50000, \quad U_C(\text{no PhD}) = 0,$$
$$U_B(\text{low}) = 100000, \quad U_B(\text{average}) = 200000, \quad U_B(\text{high}) = 500000$$

Should I do a PhD?



The expected utility of Education is

$$U(E) = \sum_{I,P} p(I|E,P)p(P|E)\left[U_C(E) + U_B(I)\right]$$

so that $U(\text{do phd}) = 260174.000$, whilst not taking a PhD is $U(\text{no phd}) = 240000.0244$, making it on average beneficial to do a PhD.

Multi Armed Bandits

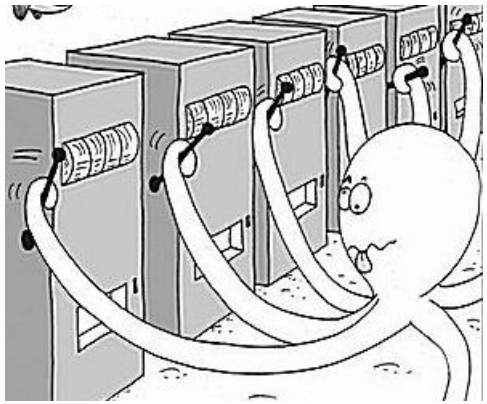


Figure: Multi-armed bandit

- A simple problem of sequential decision making is multi armed bandit setting.
- Suppose there is a row of K slot machines, each of which has some distribution of payoffs.
- We want to maximize cumulative payoff (in some sense)
- How to play?
- What is success?

The Key Issue

- If we knew the distributions of the payoffs, the optimal strategy would be just to pull the one with the highest mean.
- As we don't know the distribution, we could learn it by observing the payoffs.
- When we have some estimates of payoff means, we could pull the arm with the highest one (this is called exploitation).
- However, if our estimate is not accurate, we might be losing, so we need to update our estimates by selecting other arms (and this is called exploration).

Some applications

- Oil drilling: where to drill next?
- Online advertisement: which banner to show?
- Game playing: which move to play?

Formally...

We can describe bandits scenario as a following game:

Players: Learner, Nature
for $t = 1, 2, \ldots, T$ **do**
Learner selects action $a \in \{1, \ldots, K\}$
Nature draws payoff r from the distribution $\pi_a = P(r|a)$
(usually with support on $[0, 1]$)
end for
Learner's cumulative regret is $R_T = \mathbb{E}[\sum_{t=1}^{T}(U^* - U_t)]$, where $U^* = \max_a \mathbb{E}[r|a]$ and $U_t = \mathbb{E}[r|a_t]$

Note: regret can be decomposed as function of gaps and counts:

$$R_T = \sum_a \mathbb{E}[N_t(a)]\Delta_a$$

where $N_t(a)$ is a number of selections for action a and gap is the difference between the value of the optimal action and action a, $\Delta_a = U^* - U_a$

Simple strategies

- *greedy*: play action with the highest current estimate of payoffs:

$$\hat{U}_t(a) = \frac{1}{N_t(a)} \sum_{\tau : a_\tau = a}^{t} r_\tau$$

  Problem: could lock on the suboptimal action ($\Longrightarrow$ linear regret).
- ϵ-*greedy*: With probability $1 - \epsilon$ play *greedy*, with probablity ϵ select action at random with uniform probability (also linear regret, unless time horizon is known in advance and ϵ is tuned for it).
- Can we get sublinear regret?

Theorem (Lai and Robbins)

Asymptotic cumulative regret is at least logarithmic in the number of steps:

$$\lim_{t \to \infty} R_t \geq \log t \sum_{a : \Delta_a > 0} \frac{\Delta_a}{KL(\pi^a || \pi^*)}$$

UCB1 idea

The key idea is known as "Optimism in the face of uncertainty".

- Imagine all the "plausible" environments ("compatible" with the data). Then select the most favourable one and the best action in it.
- "Plausible" here could mean the confidence bounds on the arms' means. The more you pull the arm, the more accurate your estimate becomes.

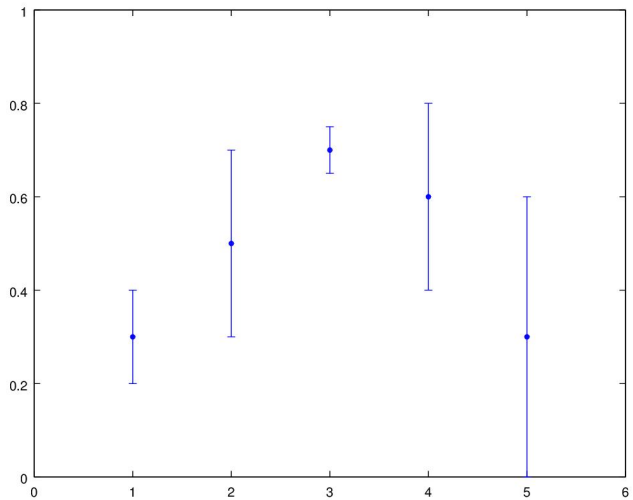


Figure: Confidence bounds on bandit arms

Upper confidence bounds

- We want to have a bound for each arm, $B_t(a)$, such that
- probability of $U(a) < \hat{U}_t(a) + B_t(a)$ is very high
- Then we'll select an arm maximising such bound

$$a_t = \arg\max_a \hat{U}_t(a) + B_t(a)$$

UCB1 algorithm

Theorem (Hoeffding bound)

Let $X_1, X_2, \ldots, X_t$ be i.i.d. random variables in $[0, 1]$ and $\bar{X}_t = \sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[\mathbb{E}[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

Applying it to the particular action a, we get:

$$P[U(a) > \hat{U}_t(a) + B_t(a)] \leq e^{-2N_t(a)B_t(a)}$$

If we want this probability to by less than t^{-4} we can solve this for $B_t(a)$ and get

$$B_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

UCB1 algorithm

- UCB1 algorithm then becomes:

$$a_t = \arg\max_a \hat{U}_t(a) + \sqrt{\frac{2\log t}{N_t(a)}}$$

Theorem (UCB1 asymptotic regret)

The UCB1 algorithm guarantees asymptotic total regret

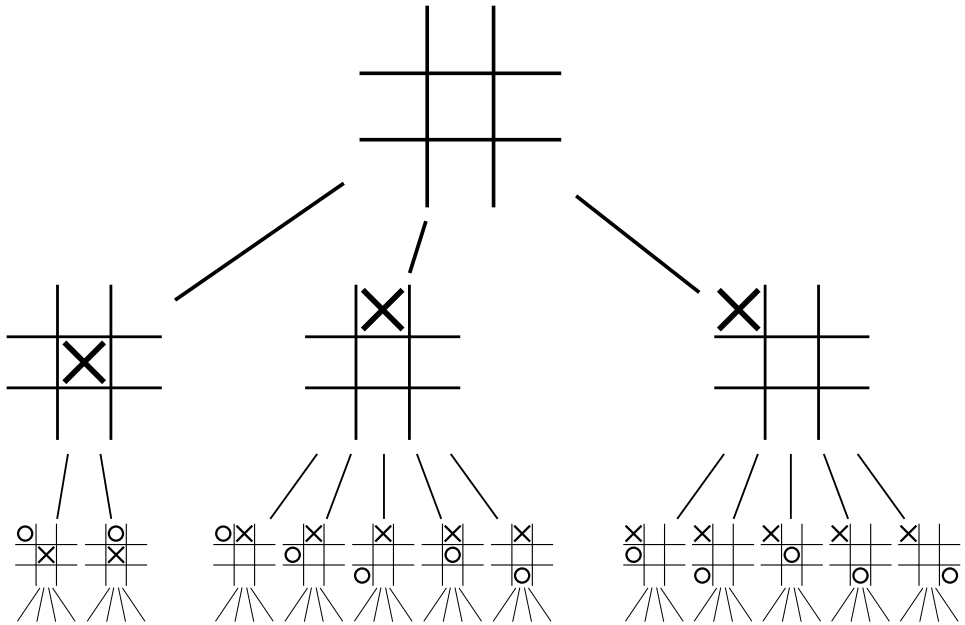
$$\lim_{t\to\infty} \leq 8\log t \sum_{a:\Delta_a>0} \Delta_a$$

Bunch of extensions – frameworks and approaches

- Adversarial Bandits
- Contextual Bandits
- Bayesian Bandits
- Best Arm Identification
- Bandits for Tree Search

Game planning

- Game tree:



- Everything is deterministic, we could solve it from the leaves.
- But for games harder than tic-tac-toe we don't have time/space to do that.
- We can use things like $\alpha - \beta$-pruning to avoid searching the whole tree. But it is still huge. What to do?

Monte Carlo Planning

- Because the tree is so big, we could try to explore it only up to certain depth. But then we need somehow to provide values at these pseudo-leaves.
- We could sample them by doing playouts (or rollouts): finish the game using some simple strategy playing against itself.
- But if we sample everything uniformly at random, we will spend a lot of time exploring bad moves.
- Idea: let's put a UCB algorithm in every node!

UCT: UCB for trees

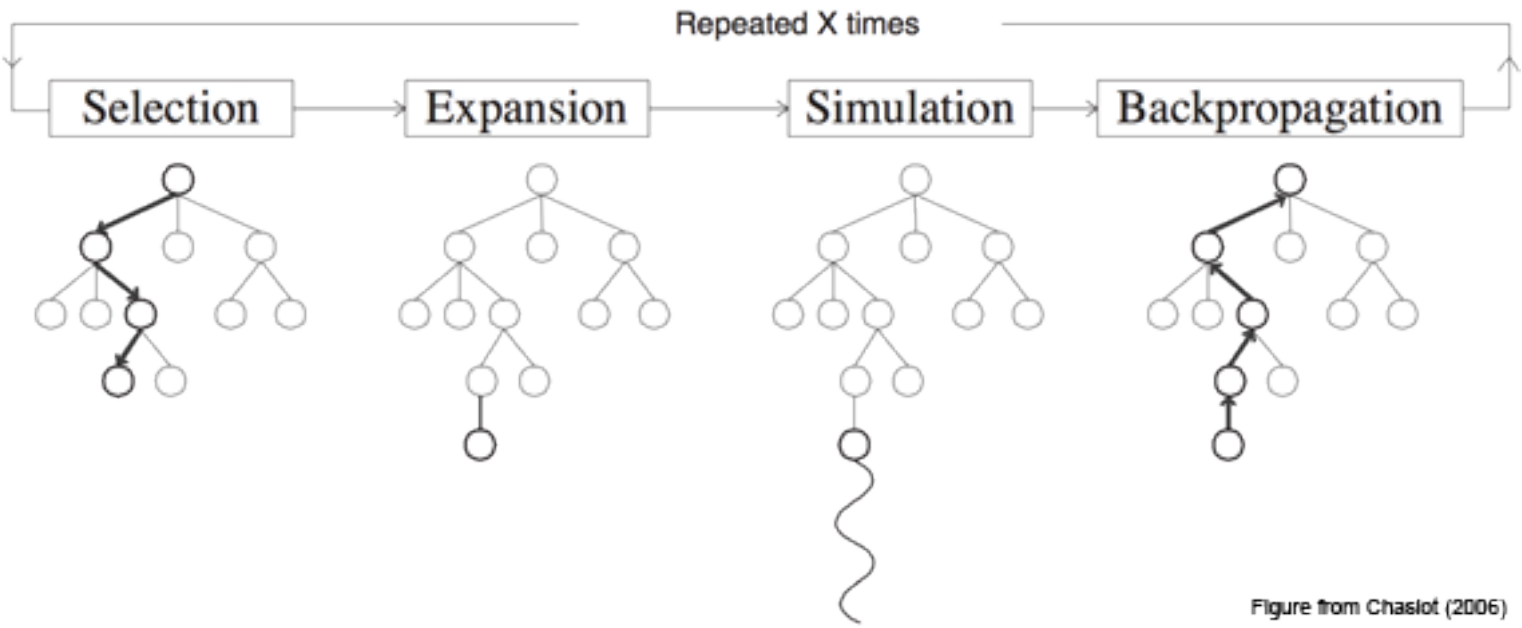


Figure: Monte Carlo Tree Search

- UCT is Monte Carlo Tree Search with UCB used at Selection stage.
- There is one issue, however, with applying UCB bounds. . . Can you spot it?

UCT: UCB for trees

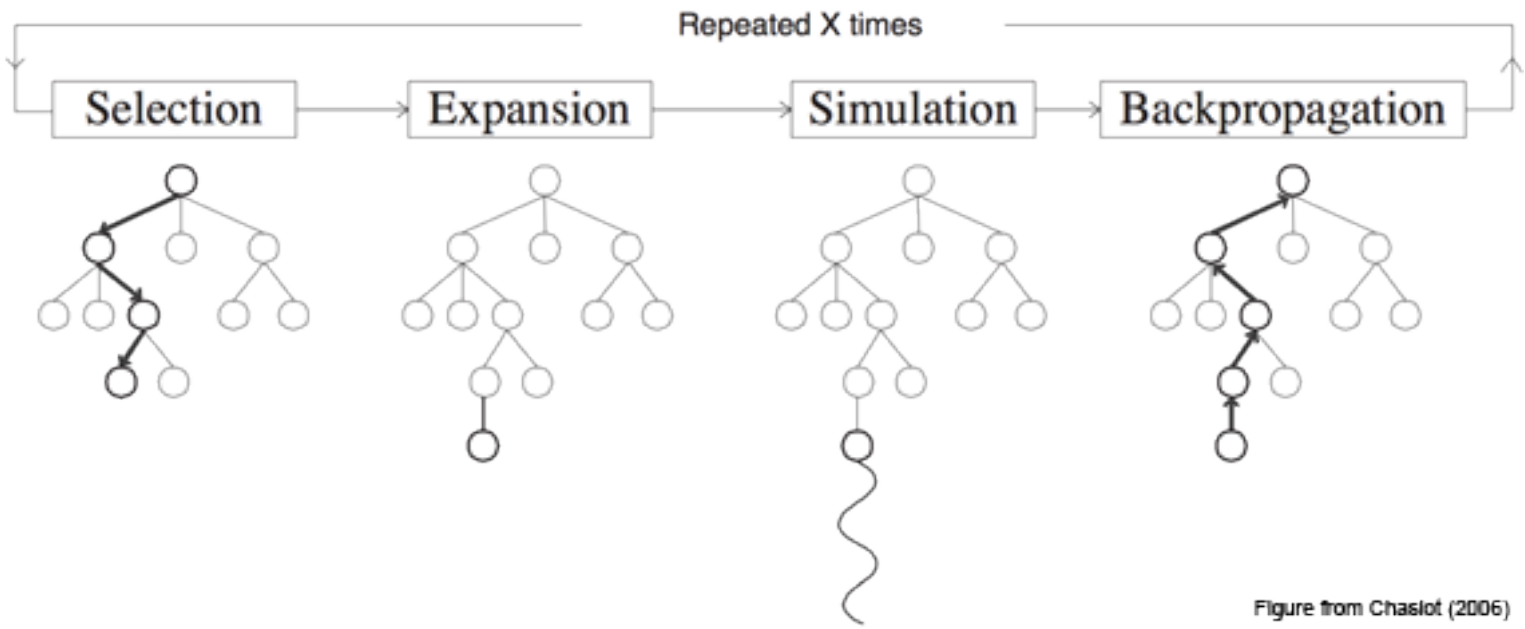
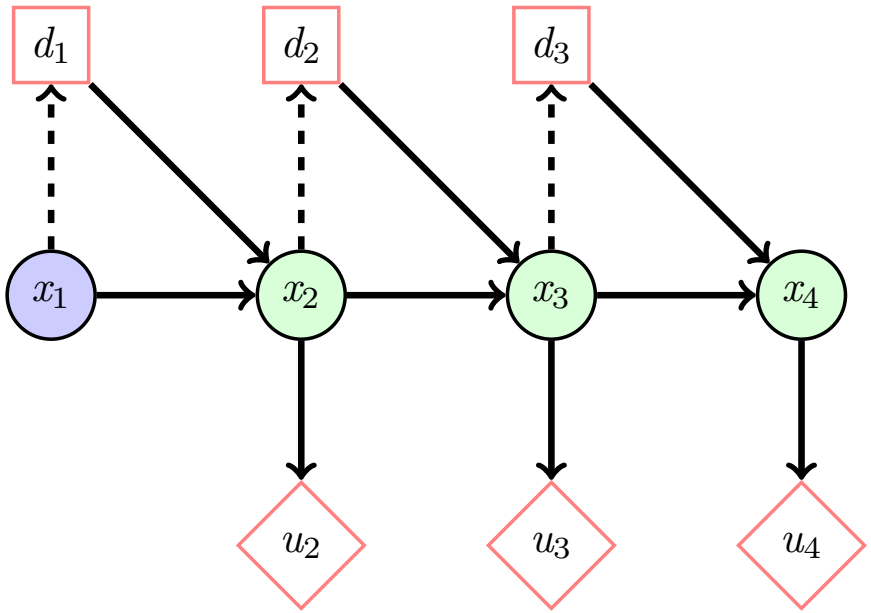


Figure: Monte Carlo Tree Search

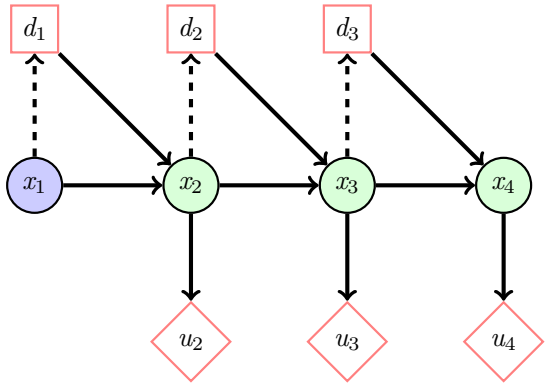
- UCT is Monte Carlo Tree Search with UCB used at Selection stage.
- There is one issue, however, with applying UCB bounds. . . Can you spot it?
- The payoffs are non-stationary anymore! Nevertheless, it's possible to prove the convergence results.
- Started the current Go revolution.

Markov Decision Process



Can be used to model planning problems of the form ‘how do I get to where I want to be incurring the lowest total cost?’.

Markov Decision Process



We want to make that decision d_1 that will lead to maximal expected total utility

$$U(d_1|x_1) \equiv \sum_{x_2} \max_{d_2} \sum_{x_3} \max_{d_3} \sum_{x_4} \ldots \max_{d_{T-1}} \sum_{x_T} p(x_{2:T}|x_1, d_{1:T-1})\, U(x_{1:T})$$

$$U(x_{1:T}) = u(x_2) + u(x_3) + \ldots + u(x_T)$$

Our task is to compute $U(d_1|x_1)$ for each state of d_1 and then choose that state with maximal expected total utility. To carry out the summations and maximisations efficiently, we can use a simple message passing approach.

Markov Decision Process

To decide on how to take the first optimal decision, we need to compute

$$\sum_{x_2} \max_{d_2} \sum_{x_3} \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)p(x_3|x_2, d_2)p(x_2|x_1, d_1) \left(u(x_2) + u(x_3) + u(x_4)\right)$$

Since only $u(x_4)$ depends on x_4 explicitly, we can write

$$\sum_{x_2} \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)p(x_2|x_1, d_1) \left(u(x_2) + u(x_3) + \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)u(x_4) \right)$$

Defining a message and corresponding value

$$u_{3\leftarrow 4}(x_3) \equiv \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)u(x_4), \qquad v(x_3) \equiv u(x_3) + u_{3\leftarrow 4}(x_3)$$

we can write

$$U(d_1|x_1) = \sum_{x_2} \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)p(x_2|x_1, d_1) \left(u(x_2) + v(x_3)\right)$$

In a similar manner, only the last term depends on x_3 and hence

$$U(d_1|x_1) = \sum_{x_2} p(x_2|x_1, d_1) \left(u(x_2) + \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)v(x_3) \right)$$

Defining similarly the value

$$v(x_2) \equiv u(x_2) + \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)v(x_3)$$

then

$$U(d_1|x_1) = \sum_{x_2} p(x_2|x_1, d_1)v(x_2)$$

Given $U(d_1|x_1)$ above, we can then find the optimal decision d_1 by

$$d_1^*(x_1) = \operatorname*{argmax}_{d_1} \; U(d_1|x_1)$$

Bellman's Equation

In an MDP we can define utility messages recursively as

$$u_{t-1\leftarrow t}(x_{t-1}) \equiv \max_{d_{t-1}} \sum_{x_t} p(x_t|x_{t-1}, d_{t-1}) \left[u(x_t) + u_{t\leftarrow t+1}(x_t)\right]$$

The value

It is more common to define the value of being in state x_t as

$$v_t(x_t) \equiv u(x_t) + u_{t\leftarrow t+1}(x_t), \qquad v_T(x_T) = u(x_T)$$

and write then the equivalent recursion

$$v_{t-1}(x_{t-1}) = u(x_{t-1}) + \max_{d_{t-1}} \sum_{x_t} p(x_t|x_{t-1}, d_{t-1}) v_t(x_t)$$

Backtracking

The optimal decision d_t^* is then given by

$$d_t^* = \operatorname*{argmax}_{d_t} \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) v_{t+1}(x_{t+1})$$

Temporally Unbounded MDPs

The infinite T case would appear to be ill-defined since the sum of utilities

$$u(x_1) + u(x_2) + \ldots + u(x_T)$$

will in general be unbounded. If we let $u^* = \max_s u(s)$ be the largest value of the utility and consider the sum of modified utilities for a chosen discount factor $0 < \gamma < 1$

$$\sum_{t=1}^{T} \gamma^t u(x_t) \leq u^* \sum_{t=1}^{T} \gamma^t = \gamma u^* \frac{1 - \gamma^T}{1 - \gamma}$$

In the limit $T \to \infty$ this means that the summed modified utility $\gamma^t u(x_t)$ is finite:

$$\lim_{T \to \infty} \sum_{t=1}^{T} \gamma^t u(x_t) \leq u^* \frac{\gamma}{1 - \gamma}$$

Temporally Unbounded MDPs

The only modification required to our previous discussion is to include a factor γ in the message definition. Assuming that we are at convergence, we define a value $v(x_t = \mathsf{s})$ dependent only on the state s, and not the time.
This means we replace the time-dependent Bellman's value recursion with the time-independent equation

$$v(\mathsf{s}) \equiv u(\mathsf{s}) + \gamma \max_{\mathsf{d}} \sum_{\mathsf{s}'} p(x_t = \mathsf{s}' | x_{t-1} = \mathsf{s}, d_{t-1} = \mathsf{d}) v(\mathsf{s}')$$

We then need to solve for the value $v(\mathsf{s})$ for all states s. The optimal decision policy when one is in state $x_t = \mathsf{s}$ is then given by

$$\mathsf{d}^*(\mathsf{s}) = \operatorname*{argmax}_{\mathsf{d}} \sum_{\mathsf{s}'} p(x_{t+1} = \mathsf{s}' | x_t = \mathsf{s}, d_t = \mathsf{d}) v(\mathsf{s}')$$

For a deterministic transition p (*i.e.* for each decision d, only one state s′ is available), this means that the best decision is the one that takes us to the accessible state with highest value.

Value Iteration

In the $T = \infty$ case, we need to actually know the value to take a decision. This means we need to solve Bellman's $T = \infty$ equations first.

- A naive procedure is to iterate the value recursion until convergence, assuming some initial guess for the values (say uniform)

$$v^{new}(\mathsf{s}) \equiv u(\mathsf{s}) + \gamma \max_{\mathsf{d}} \sum_{\mathsf{s}'} p(x_t = \mathsf{s}' | x_{t-1} = \mathsf{s}, d_{t-1} = \mathsf{d}) v^{old}(\mathsf{s}')$$

- One can show that this value iteration procedure is guaranteed to converge to a unique optimum.
- The convergence rate depends somewhat on the discount γ – the smaller γ is, the faster is the convergence.

Policy Iteration

First assume we know the optimal decision $\mathsf{d}^*(\mathsf{s})$ for any state s. Then

$$v(\mathsf{s}) = u(\mathsf{s}) + \gamma \sum_{\mathsf{s}'} p(x_t = \mathsf{s}' | x_{t-1} = \mathsf{s}, \mathsf{d}^*(\mathsf{s})) v(\mathsf{s}')$$

The maximisation over d has disappeared since we have assumed we already know the optimal decision for each state s. For fixed $\mathsf{d}^*(\mathsf{s})$, the above is now linear in the value. Defining the value $\mathbf{v}$ and utility $\mathbf{u}$ vectors and transition matrix $\mathbf{P}$,

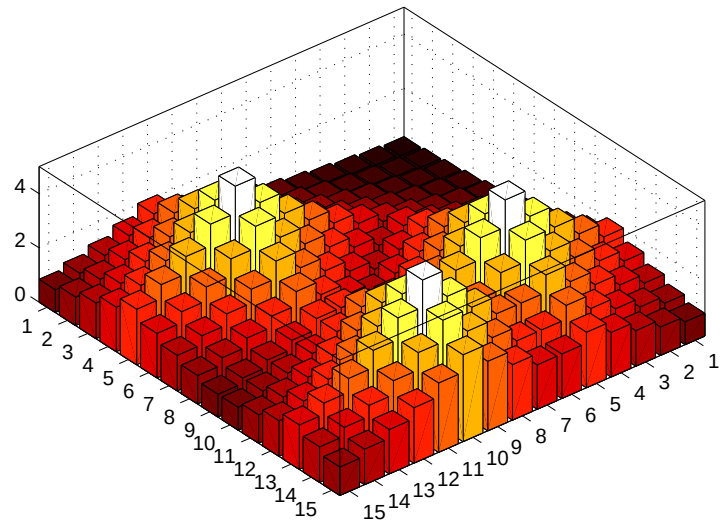
$$[\mathbf{v}]_{\mathsf{s}} = v(\mathsf{s}), \qquad [\mathbf{u}]_{\mathsf{s}} = u(\mathsf{s}), \qquad [\mathbf{P}]_{\mathsf{s}',\mathsf{s}} = p(\mathsf{s}' | \mathsf{s}, \mathsf{d}^*(\mathsf{s}))$$

in matrix notation:

$$\mathbf{v} = \mathbf{u} + \gamma \mathbf{P}^{\mathsf{T}} \mathbf{v} \Leftrightarrow \left(\mathbf{I} - \gamma \mathbf{P}^{\mathsf{T}}\right) \mathbf{v} = \mathbf{u} \Leftrightarrow \mathbf{v} = \left(\mathbf{I} - \gamma \mathbf{P}^{\mathsf{T}}\right)^{-1} \mathbf{u}$$

These linear equations are readily solved with Gaussian Elimination. Using this, the optimal policy is recomputed. The two steps of solving for the value, and recomputing the policy are iterated until convergence.

Solving an MDP



Value Iteration on a set of 225 states, corresponding to a 15×15 two dimensional grid. Deterministic transitions to neighbours on the grid, $\{\text{stay}, \text{left}, \text{right}, \text{up}, \text{down}\}$. There are three goal states, each with utility 1 – all other states have utility 0. Plotted is the value $v(\text{s})$ for $\gamma = 0.9$ after 30 updates of Value Iteration, where the states index a point on the $x - y$ grid. The optimal decision for any state on the grid is to go to the neighbouring state with highest value.

Reinforcement Learning

A form of learning

- This is an MDP in which the transition and possibly reward function (utility) is not a priori known.
- As one interacts with the environment, one may begin to build an understanding of these and then define a policy accordingly.

Exploration Exploitation dilemma

- But wait! If I don't collect enough information, why should I start to act as if I know the transition and reward perfectly? This is the 'exploration-exploitation' dilemma.
- There are nice ways to address this using Bayesian methods.

Q Learning

- Consider the RL situation in which we don't have access to the transition, only samples from this. (The 'model-free' setting).
- We cannot use the Bellman equations in this case.
- Q-learning is a form that allows us to solve for the optimal policy using samples from the environment.

The expected utility of taking decision d_t when in state x_t:

$$U(d_t|x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) \left(u(x_{t+1}) + \gamma \max_d \sum_{x'} p(x'|x_{t+1}, d) v(x') \right)$$

Using the value recursion

$$v(x_{t+1}) = u(x_{t+1}) + \gamma \max_d \sum_{x'} p(x'|x_{t+1}, d) v(x')$$

we can derive

$$U(d_t|x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) \left(u(x_{t+1}) + \max_d U(d|x_{t+1}) \right)$$

Q Learning

- If we are in state x_t and take decision d_t the environment returns for us a sample x_{t+1}.
- This gives the one-sample estimate:

$$\tilde{U}(d_t|x_t) = u(x_{t+1}) + \gamma \max_d \tilde{U}(d|x_{t+1})$$

- To ensure convergence it is preferable to use

$$\tilde{U}_{t+1}(d_t|x_t) = (1-\alpha_t)\,\tilde{U}_t(d_t|x_t) + \alpha_t \left(u(x_{t+1}) + \gamma \max_d \tilde{U}_t(d|x_{t+1}) \right)$$

  which can be written

$$\tilde{U}_{t+1}(d_t|x_t) = \tilde{U}_t(d_t|x_t) + \alpha_t \left(u(x_{t+1}) + \gamma \max_d \tilde{U}_t(d|x_{t+1}) - \tilde{U}_t(d_t|x_t) \right)$$

  where the learning rate satisfies $0 \leq \alpha < 1$, $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$. (For example $\alpha_t = 1/t$.)
- Provided all decisions are repeatedly sampled for each state, the sample estimate $\tilde{U}_t(d|x)$ converges to the exact $U(d|x)$ in the limit $t \to \infty$.

Further Topics

More complex structures

- Can solve more general IDs using a form of Junction Tree algorithm.
- Many real world problems are structured MDPs. This creates a huge state space (Games, Scheduling, Landing airplanes, Life)

POMDPs

- In a Partially Observable (POMDP) not all the states of the MDP are observed.
- This creates a more complex problem for which no efficient message passing algorithms are available.
- Still an active research area.