4F13: Machine Learning

Lectures 12-14: Reinforcement Learning

Zoubin Ghahramani
`zoubin@eng.cam.ac.uk`

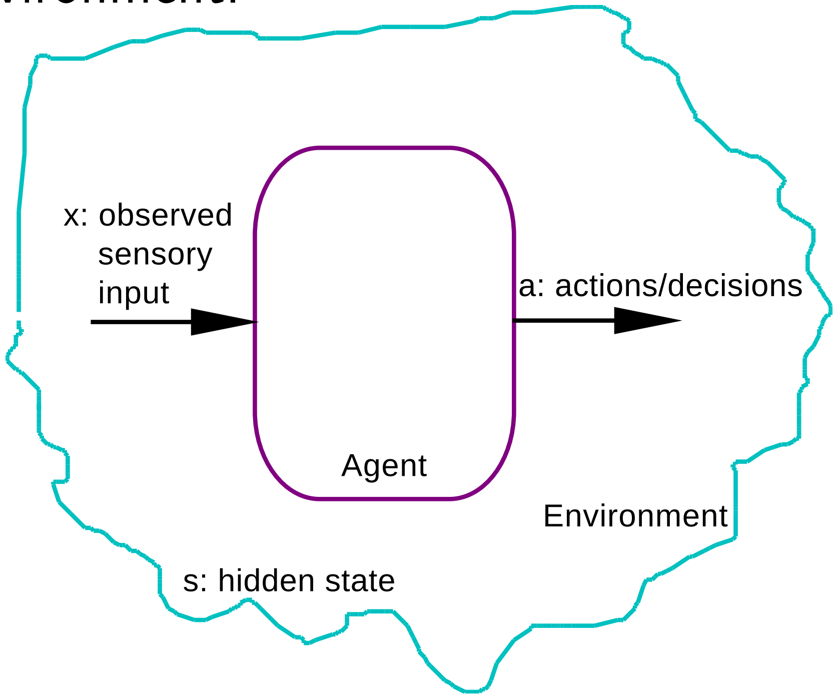
Department of Engineering
University of Cambridge

Michaelmas, 2006

http://learning.eng.cam.ac.uk/zoubin/ml06/

Intelligent Behaviour?

Imagine a creature/agent (human/animal/machine) which receives sensory inputs and can take some actions in an environment:



Assume that the creature also receives rewards (or penalties/losses) from the environment.

The goal of the creature is to maximise the rewards it receives (or equivalently minimise the losses).

A theory for choosing actions that minimize losses is a theory for how to behave optimally...

Bayesian Decision Theory

Bayesian decision theory deals with the problem of making optimal decisions—that is, decisions or actions that minimize an expected loss.

- Let's say we have a choice of taking one of k possible **actions** $a_1 \ldots a_k$.
- Assume that the world can be in one of m different **states** $s_1, \ldots, s_m$.
- If we take action a_i and the world is in state s_j we incur a **loss** ℓ_{ij}
- Given all the observed data $\mathcal{D}$ and prior background knowledge $\mathcal{B}$, our **beliefs** about the state of the world are summarized by $p(s|\mathcal{D}, \mathcal{B})$.
- *The optimal action is the one which is expected to minimize loss (or maximize utility)*:

$$a^* = \underset{a_i}{\operatorname{argmin}} \sum_{j=1}^{m} \ell_{ij} \, p(s_j|\mathcal{D}, \mathcal{B})$$

Bayesian sequential decision theory	(statistics)
Optimal control theory	(engineering)
Reinforcement learning	(computer science / psychology)

A simple example

Assume we have two actions:
a_1 : play
a_2 : don't play

And two outcomes:
s_1 : win lottery
s_2 : don't win lottery

Optimal action:

$$a^* = \operatorname*{argmin}_{a_i} \sum_{j=1}^{m} \ell_{ij}\, p(s_j|a_i, \mathcal{B})$$

$p(s_1\|a_1, \mathcal{B}) = 0.000001$	$\ell_{11} = -100000$
$p(s_2\|a_1, \mathcal{B}) = 0.999999$	$\ell_{12} = +0.9$
$p(s_1\|a_2, \mathcal{B}) = 0$	$\ell_{21} = 0$
$p(s_2\|a_2, \mathcal{B}) = 1$	$\ell_{22} = 0$

What is the optimal action for this decision problem?

Comments about the above framework

The optimal action is the one which is expected to minimize loss (or maximize utility):

$$a^* = \operatorname*{argmin}_{a_i} \sum_{j=1}^{m} \ell_{ij}\, p(s_j|\mathcal{D}, \mathcal{B})$$

- This is a theory for how to make a *single decision*. How do we make a *sequence of decisions* in order to achieve some long-term goals/rewards?
- This assumes that we *know* what the losses are for each action-state pair. The losses may in fact have to be learned from experience.
- We need a model for how the observed data $\mathcal{D}$ relates to the states of the world s.
- It may be impossible to *enumerate* all possible actions and states. What about continuous state and action spaces?

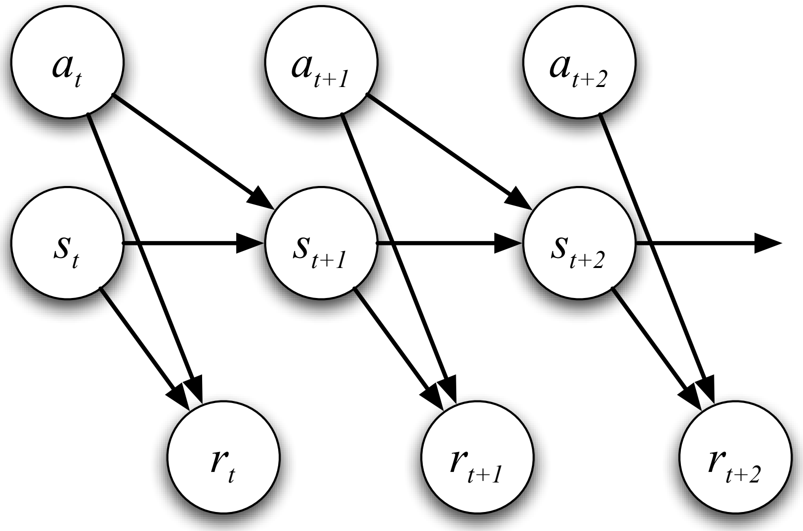
Markov Decision Problems (MDPs)

States: s_t
Actions: a_t
Rewards: r_t

The variable s_t is the state of the world and agent at time t

The agent takes action a_t and receives reward r_t (or loss, if you like to think negatively...)

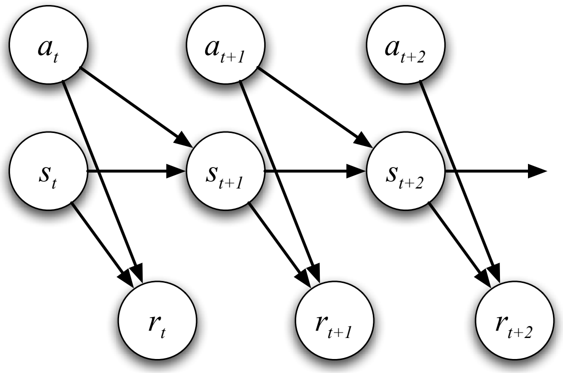
The reward is assumed to depend on the state and the action.



Markov property: $P(s_{t+1}, r_t | s_t, a_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots) = P(s_{t+1}, r_t | s_t, a_t)$

Markov Decision Problems (MDPs)

States: s_t
Actions: a_t
Rewards: r_t



The goal in an MDP is not to maximize immediate rewards but to maximize long term accumulated rewards. How should we accumulate rewards?

Average (undiscounted) future return:

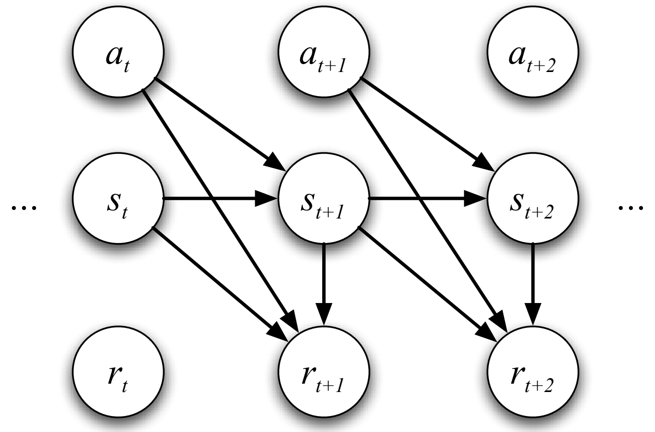
$$R_t = \lim_{k\to\infty} \frac{1}{k} \sum r_{t+k+1}$$

The above assumes that a reward infinitely into the future is as valuable as a reward now. Discounting is a sensible heuristic for how future rewards may relate to immediate rewards (think inflation, or probability that the game will end, etc):

Discounted future return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Markov Decision Problems (MDPs)



The world is characterized by

Transition Probabilities: $\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$

Expected rewards: $\mathcal{R}^a_{ss'} = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$

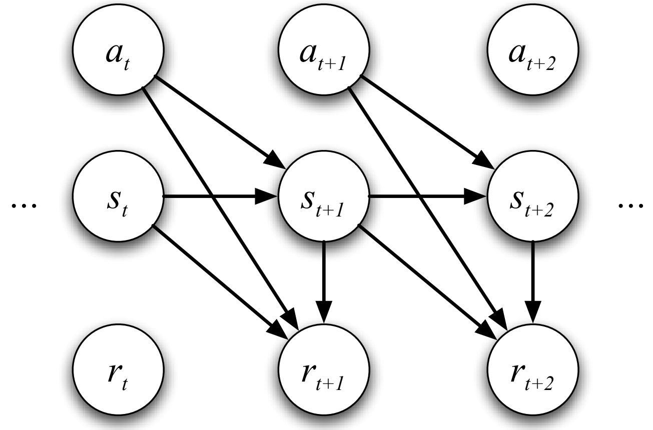
The agent is characterized by

Policy: $\pi(s, a) = P(a_t = a | s_t = s)$

Why is the action at time t only dependent on the state at time t?

Markov Decision Problems (MDPs)

Why is the action at time t only dependent on the state at time t?



The action a_t should be chosen to maximize sum of (discounted) future rewards R_t

By the Markov properties in the graph (i.e. by conditional independence), future rewards and states are independent of past rewards, actions, and states given s_t and a_t:

$$P(s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, \ldots | s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = P(s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, \ldots | s_t, a_t)$$

If s_t is known, the *expected* value of the return R_t depends only on a_t, so previous states and actions are irrelevant.

Value Functions

Value Function: how good is it to be in a given state? This obviously depends on the agent's policy:

$$V^\pi(s) = E_\pi(R_t | s_t = s) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right)$$

State-action value function: how good is it to be in a given state and take a given action, and then follow policy π:

$$Q^\pi(s,a) = E_\pi(R_t | s_t = s, a_t = a) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right)$$

The relation between the state value function and the state-action value function:

$$V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$$

Self-Consistency of Value Functions

A fundamental property of value functions is that they satisfy a set of recursive consistency equations. V^π is the unique solution to these equations.

$$\begin{aligned}
V^\pi(s) &= E_\pi(R_t | s_t = s) \\
&= E_\pi\left(r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right) \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+l+2} \middle| s_{t+1} = s' \right) \right] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]
\end{aligned}$$

We can solve them using a “backup operation” from $s' \to s$ (or other means). Linear system of $N \equiv |s|$ equations in N unknowns.

$$\mathbf{v} = (I - \gamma \sum_a \mathrm{diag}(\boldsymbol{\pi}_a) \mathcal{P}^a)^{-1} (\sum_a \boldsymbol{\pi}_a \odot \mathrm{diag}(\mathcal{P}^a \mathcal{R}^{a\top}))$$

There is a similar equation for $Q^\pi(s,a)$

Optimal Policies and Values

Optimal Policy: π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)\ \forall s$. There may be more than one optimal policy.

Question: Is there always at least one optimal policy? YES

Optimal state value function: $V^*(s) = \max_\pi V^\pi(s)\ \forall s$

Optimal state-action value function: $Q^*(s,a) = \max_\pi Q^\pi(s,a)\ \forall s$. This is the expected return of action a in state s, thereafter following optimal policy.

$$Q^*(s,a) = E\left(r_{t+1} + \gamma V^*(s_{t+1}) \right| s_t = s, a_t = a)$$

Bellman Optimality Equation

$$
\begin{aligned}
V^*(s) &= \max_a Q^{\pi^*}(s,a) \\
&= \max_a E_{\pi^*}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right) \\
&= \max_a E_{\pi^*}\left(r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a \right) \\
&= \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^*(s')]
\end{aligned}
$$

$$
\begin{aligned}
Q^*(s,a) &= E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| s_t = s, a_t = a \right) \\
&= \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')]
\end{aligned}
$$

N *nonlinear* equations in N unknowns for V^*
NA *nonlinear* equations in NA unknowns for Q^*

Solving MDPs

Given the optimal value function, V^*, it is easy to get optimal policy π^*: be **greedy** w.r.t. V^*.

If you have V^*, the actions that appear best after a one-step search will be optimal.

V^* turns a long-term reward into a quantity that is locally and immediately available.

Using Q^* it is even easier to get the optimal policy:

$$\pi^*(s,a) = 0 \;\; \forall a \;\; s.t. \;\; Q^*(s,a) \neq \max_{a'} Q^*(s,a')$$

Policy Improvement Theorem

Policy Evaluation $$V^\pi_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi_k(s')]$$

assumes $\mathcal{P}$ known, $\mathcal{R}$ known, and a full backup (we can also sweep in place)

Policy Improvement Theorem

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s) \; \forall s \; \Rightarrow \; V^{\pi'}(s) \geq V^\pi(s)$$

Proof:

$$\begin{array}{rcl}
V^\pi(s) & \leq & Q^\pi(s, \pi'(s)) \\
& = & E_{\pi'}(r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s) \\
& \leq & E_{\pi'}(r_{t+1} + \gamma Q^\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s) \\
& = & E_{\pi'}(r_{t+1} + \gamma E_\pi(r_{t+2} + \gamma V^\pi(s_{t+2})) | s_t = s) \\
& = & E_{\pi'}(r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(s_{t+2}) | s_t = s) \\
& \vdots & \\
& \leq & V^{\pi'}(s)
\end{array}$$

Policy Iteration

The policy improvement theorem suggests a way of improving policies:

$$\begin{aligned}\pi'(s) &\leftarrow \arg\max_a Q^\pi(s,a) \ \ \forall s \\ &= \arg\max_a E(r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a)\end{aligned}$$

This procedure converges to an optimal policy by policy improvement theorem and Bellman optimality.

$$V^{\pi'}(s) \geq \arg\max_a Q^\pi(s,a) \geq \sum_a \pi(s,a) Q^\pi(s,a) = V^\pi(s)$$

Policy Iteration: Iterates between evaluation and improvement

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \ldots \pi^*$$

Problem with Policy Iteration: Evaluation step can be really slow...

Value Iteration

Do we really need to wait until convergence?

In fact, we can improve after **one** sweep of evaluation!

$$\begin{aligned} V_{k+1}(s) &\leftarrow \max_a E(r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a) \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')] \end{aligned}$$

converges: $V_k \longrightarrow V^*$. At each step we also have a policy.

Problem: it is still not feasible to update the value of every single state.
E.g. backgammon has 10^{20} states!

Bellman called this the **curse of dimensionality**

Asynchronous dynamic programming

These are in-place iterated dynamic programming (DP) algorithms that are not organized in terms of systematic sweeps over all the states.

States are backed-up in order visited or randomly.

To converge the algorithms must continue to visit every state.

Key idea in RL: We can run the DP algorithm at the same time as the agent is *actually experiencing* the MDP.

This leads to an **exploration vs exploitation tradeoff**: act so as to visit new parts of state space or exploit already visited part of state-space?

An example of a simple exploration strategy are ϵ-greedy policies:

$$\pi_\epsilon(s,a) = (1-\epsilon)\pi(s,a) + \epsilon u(a)$$

Can you think of anything wrong with this?

Monte Carlo and TD

Monte Carlo methods solve RL problems by averaging sample returns.

Question: how do you trade off length of sampled trajectory, vs previously estimated values?

$$\begin{aligned}
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (1)\\
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) - V(s_t)]\\
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V(s_{t+3}) - V(s_t)]\\
&\vdots\\
V(s_t) &\leftarrow V(s_t) + \alpha[R_t - V(s_t)] \quad (2)
\end{aligned}$$

Equation (1) is **Temporal Difference learning, TD(0)**. TD(λ) approximates the range eqn (1)–(2), where higher λ is closer to the full MC method.

TD(λ) has been proven to converge

These are general methods for controlling the **bias-variance tradeoff**.

SARSA and Q Learning

Definitions: *on-policy* methods evaluate or improve the current policy used for control. *Off-policy* methods evaluate or improve one policy, while acting using another (behavior) policy.

In off-policy methods, the behavior policy must have non-zero probability for each state-action the evaluated policy does.

SARSA: on-policy greedy control

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Q Learning: off-policy greedy control

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Converges if $\forall a, s$ are visited and updated infinitely often.

We can also combine the bias-variance ideas with Q and SARSA, to get $Q(\lambda)$ and SARSA(λ).

Function Approximation

For very large or continuous state spaces it is hopeless to store a table with all the state values or state-action values.

It makes sense to use **function approximation**

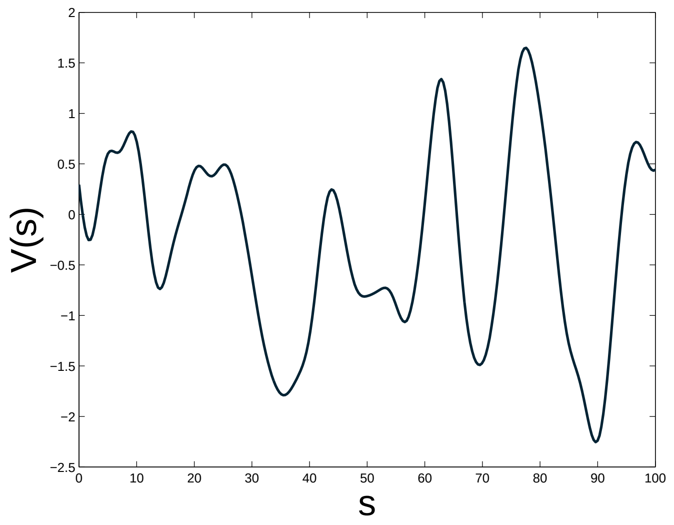
$$V(s) = f_\theta(s)$$

e.g. basis function representation:

$$V(s) = \sum_i \theta_i \phi_i(s)$$

Similarly for $Q(s, a)$.

This should hopefully lead to better **generalization**



Gradient descent methods:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \alpha[v_t - V_t(s_t)]\frac{\partial V_t(s_t)}{\partial \boldsymbol{\theta}}$$

where α is a learning rate and v_t is a measured/estimated value. See chapter 8 of Sutton and Barto.

Optimal Control

Optimal Control: The engineering field of optimal control covers exactly the same topics as RL, except the state and action space is usually assumed to be continuous, and the model is often known.

The **Hamilton-Jacobi-Bellman** optimality conditions are the continuous state generalization of the Bellman equations.

A typical elementary problem in optimal control is the linear quadratic Gaussian control **LQG** problem. Here the cost function is quadratic in states $\mathbf{x}_t$ and actions $\mathbf{u}_t$, and the system is a linear-Gaussian state-space model.

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \epsilon_t$$

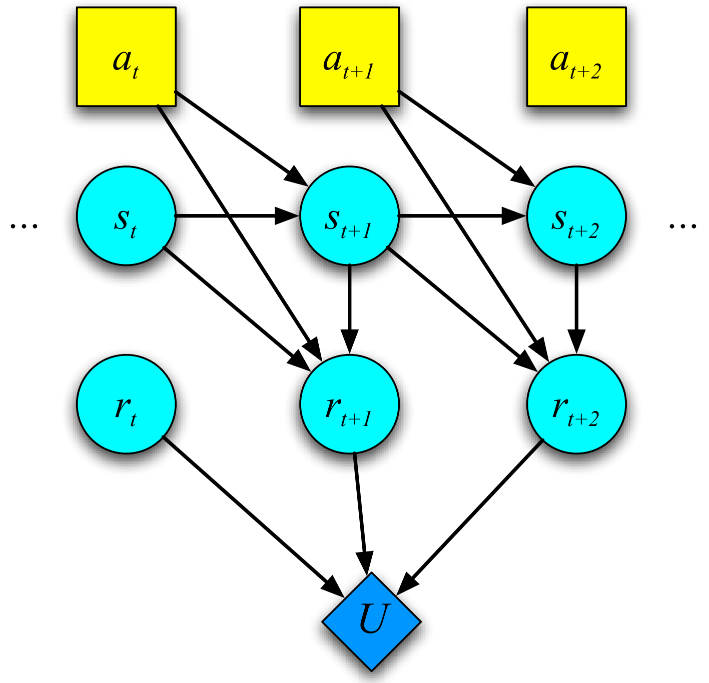
For this model the optimal policy can be computed from the estimated state. It's a linear feedback controller:

$$\mathbf{u}_t = L\hat{\mathbf{x}}_t$$

The optimal policy here happens not depend on the uncertainty in $\mathbf{x}_t$. This is not generally the case.

Influence Diagrams

You can extend the framework of directed acyclic probabilistic graphical models (a.k.a. Bayesian networks) to include **decision nodes** and **value nodes**. These are called **influence diagrams**.



Solving an influence diagram corresponds to finding the settings of the decision nodes that maximize the expectation of the value node.

It is possible to convert the problem of solving an influence diagram into the problem of doing inference in a (usually multiply connected) graphical model (Shachter and Peot, 1992). Exact solutions can be computationally intractable.

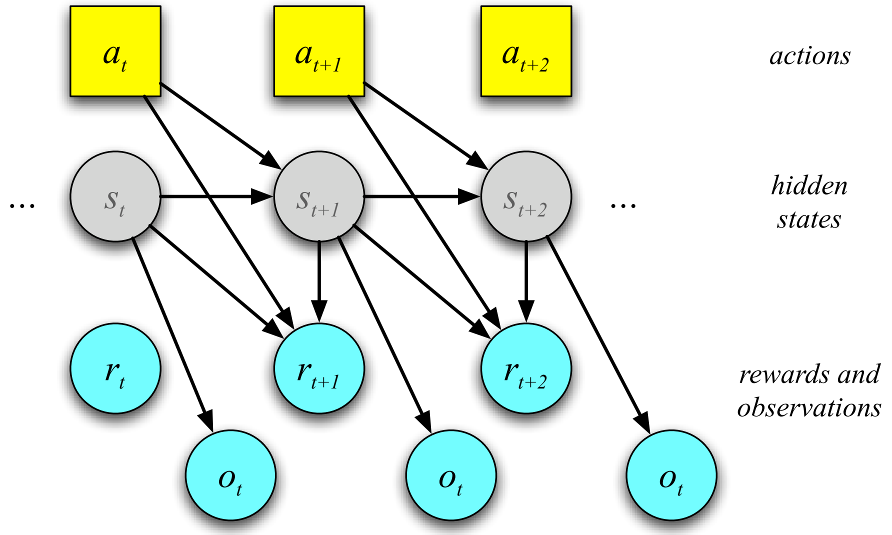
Like other graphical models, influence diagrams can contain both observed and **hidden** variables...

POMDPs

POMDP = Partially-observable Markov decision problem.

The agent does not observe the full state of the environment.

What is the optimal policy?



- If the agent has the correct model of the world, it turns out that the optimal policy is a (piece-wise linear) function of the **belief state**, $P(s_t|a_1, \ldots, a_{t-1}, r_1, \ldots, r_t, o_1, \ldots, o_t)$. Unfortunately, the belief state can grow exponentially complex.

- Equivalently, we can view the optimal policy as being a function of the entire sequence of past actions and observations (this is the usual way the policy in influence diagrams is represented). Again, unfortunately, the set of possible such sequences grows exponentially.

Efficient methods for approximately solving POMDPs is an active area of research.

Some References on Reinforcement Learning

- Kaelbling, L.P., Littman, M.L. and Moore, A.W. (1996) Reinforcement Learning: A Survey. Journal of Aritificial Intelligence Research 4:237-285.
- Sutton, R.S. and Barto, A.G. (2000) Reinforcement Learning: An Introduction. MIT Press.
- Bertsekas, D.P. and Tsitsiklis, J.N. (1996) Neuro-Dynamic Programming. Athena Scientific.
- Bryson, A.E. and Ho, Y.-C. (1975) Applied Optimal Control. Hemisphere Publishing Corp. Washington DC.