Lecture 2 - *Examples of Decision Problems*

OBJECTIVE: This lecture introduces several examples of decision problems such bandits, linear quadratic (LQ) control, sequential resource allocation, optimal stopping and queuing systems. The goal is to illustrate the broad scope of application of these ideas as well as to improve our understanding of the concepts introduced in the previous lecture.

◇ BANDITS

Bandits are important in that they are widely applicable and enjoy some mathematical tractability. Initially, think of a bandit as a slot machine in a casino. When in Vegas, a gambler might have to choose among K bandits. Each bandit has an internal dynamic state that determines how much reward the gambler obtains.

In classical bandits, it is assumed that only the chosen bandit changes state. The other bandits remain at their current state. In **restless bandits**, the other bandits can also evolve. Restless bandits are more expressive, but less tractable.

The gambler has to decide which of the bandits are worth playing. If the gambler is doing well at a machine, there is no guarantee that another machine might not be better (or worse). The gambler faces a dilemma known as the **exploitation-exploration trade-off**.

More formally, the components of a bandit are:

- **[States]**: $\mathbf{x}_t^i$ is the state of bandit i. The entire state is $\mathbf{x}_t^{1:K} \in \mathcal{X}^1 \times \cdots \times \mathcal{X}^K$.
- **[Actions]**: $\mathbf{a}_t \in \mathcal{A}(x) = \{1, 2, \ldots, K\}$.
- **[Rewards]**: $r_t(\mathbf{x}_t^{1:K}, \mathbf{a}_t) = r_t(\mathbf{x}_t^i, \mathbf{a}_t = 1)$.

- **[Transition law]**: For classical bandits, the selected bandit evolves according to: $p_t(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i)$.

The goal is to choose a policy that maximizes the expected rewards over a planning horizon.

Bandits can be applied in many domains:

1. Project selection.
2. Web crawling. That is, a crawler decides which webpages should be visited. This is a restless bandit problem.
3. Allocation of treatments or drugs in clinical trials.

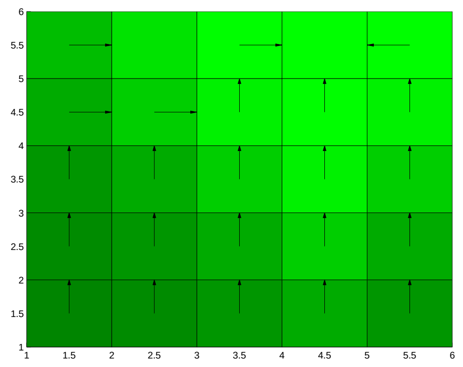
◇ GRIDWORLDS

Gridworlds illustrate the concepts of stochastic planning clearly. Many examples appear in the introductory book of Sutton and Barto. Consider the following example:

⋆

- **[States]**:
- **[Actions]**:
- **[Rewards]**:
- **[Transition law]**:
- **[Policy]**:

The optimal policy found by running an algorithm (which will be described in the next lecture) is shown below:



Note that in cell (2,2), it is optimal to go east instead of taking the greedy action (north).

The shading of each box indicates the value of the state. Lighter states are more desirable. We will formalize this notion of state value in the following lecture.

◇ BAYESIAN DECISION MAKING

In the Bayesian paradigm the state is only partially observed, but often one has access to a prior distribution over such state $p(\mathbf{x})$.

Let us consider the myopic case first. This is hard enough already. Given the obervations $\mathbf{y}$, the goal is to minimize the **Bayes risk**:

$$\min_{\mathbf{a}} \int_{\mathcal{X}} \int_{\mathcal{Y}} l(\mathbf{x}, \mathbf{a}(\mathbf{y}))\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y}\ p(\mathbf{x})\ d\mathbf{x}$$

where the loss function $l(\cdot)$ is simply the negative reward function.

As an example, consider the problem of selecting which input data samples (covariates) should be labelled within a supervised learning framework. This problem arises for example in medical diagnosis where one has many patients, but labels such as response to treatment can be very expensive. In this example, we have the following components:

- **[Observations]**: The known input variables $\mathbf{a}$ and the unknown output labels $\mathbf{y}$.
- **[States]**: The parameters $\mathbf{x}$ of a nonlinear supervised model $f(\mathbf{y}|\mathbf{x},\mathbf{a})$
- **[Actions]**: $\mathbf{a}$ indicates which input point should be selected.
- **[Loss]**: The squared error between the output and the model prediction $(\mathbf{y} - f(\mathbf{x},\mathbf{a}))^2$.
- **[Bayes rule]**:

$$\min_{\mathbf{a}} \int_{\mathcal{X}} \int_{\mathcal{Y}} (\mathbf{y} - f(\mathbf{x},\mathbf{a}))^2 \; p(\mathbf{y}|\mathbf{x},\mathbf{a}) \; d\mathbf{y} \; p(\mathbf{x}) \; d\mathbf{x}$$

The **posterior risk** is another important quantity in Bayesian decision theory. It is defined as follows:

$$\mathbb{E}[l(\mathbf{x},\mathbf{a}(\mathbf{y}))] = \int_{\mathcal{X}} l(\mathbf{x},\mathbf{a}(\mathbf{y})) \; p(\mathbf{x}|\mathbf{y}) \; d\mathbf{x}$$

Proposition: Given a specific dataset $\mathbf{y}$, minimizing the posterior risk is equivalent to minimizing the Bayes risk.

⋆ Proof:

Bayesian decision theory is very important. We will dedicate several lectures to myopic as well as sequential Bayesian strategies.

Bayesian decision problems are everywhere. Examples include:

- Which question(s) to ask?
- Experimental design. The papers of Peter Muller using Monte Carlo methods are recommended.
- Selecting data points for labelling. See for example the paper of Maryam Mahdaviani and AIBO.
- Selecting parameter values for complex differential equation models (such as the ones used by Eric Brochu for smoke animation).
- Deciding where to place sensors in a sensor network. This problem has been studied extensively by Carlos Guestrin.

- Choosing an exploration strategy for a robot learning a map of its surroundings (active SLAM). Rob Sim and Nic Roy have been investigating this.
- Deciding which entities should be presented to a user in a relevance feedback interactive human-computer interface.
- Locating the beginning and ending of regions with a high concentration of the cytosine-guanine sequence in long DNA sequences. Andreas Krause and Carlos Guestrin discuss this in their IJCAI 2005 paper.
- Structured classification for part-of-speech tagging.

◇ ALGORITHM DESIGN

Suppose we have three sorting algorithms: mergeSort, quickSort and insertionSort. Which one should we use when? Michael Littman suggests using the following MDP:

- **[State]**: Size of the input.
- **[Actions]**: Choose one of the 3 sorting algorithms.
- **[Transition law]**: If $\mathbf{a} = insertionSort$, go to terminal state (sorted input) with probability 1. If $\mathbf{a} = mergeSort$, recurse to two new states $\mathbf{x}/2$ and $\mathbf{x}/2$ with probability 1. If $\mathbf{a} = quickSort$, recurse to two states p and $\mathbf{x} - p$. $p = 1$ with probability $2/\mathbf{x}$ and takes any value between 2 and $\mathbf{x} - 1$ with probability $1/\mathbf{x}$.
- **[Running time cost]**: We can run all algorithms in different problem instances to obtain an empirical estimate of the running time. It is possible to combine this with heuristics.

This idea can also be applied to choosing among branching heuristics in branch and bound algorithms.

◇ DISCRETE CONTROL

The typical discrete control example is inventory control. The goal is to decide how many goods to order so as to avoid excess or shortage of stock.

- **[State]**: Number of stock units $\mathbf{x}_t$.
- **[Actions]**: How many units to order $\mathbf{a}_t$.
- **[Transition law]**: The stock follows the following stochastic evolution $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{a}_t - \mathbf{w}_t$, where the random variable $\mathbf{w}_t$ is the demand at time t.
- **[Cost]**: The cost has two components: purchasing cost $K\mathbf{a}_t$, where K is a constant, and stock excess and shortage cost $c(\mathbf{x}_t + \mathbf{a}_t - \mathbf{w}_t)$. The cost over N decisions is:

$$\mathbb{E}\left[\sum_{t=0}^{N-1} K\mathbf{a}_t + c(\mathbf{x}_t + \mathbf{a}_t - \mathbf{w}_t)\right]$$

It is possible to derive the optimal policy for the type of problem.

◇ LINEAR QUADRATIC CONTROL

LQ control is a ubiquitous type of optimal control. In the observable (**perfect state information**) case, it has the following components:

- **[States]**: $\mathbf{x}_t \in \mathbb{R}^n$.
- **[Actions]**: $\mathbf{a}_t \in \mathbb{R}^m$.
- **[Disturbances]**: $\mathbf{w}_t$ are independent, zero mean, finite variance random vectors. They do not depend on the actions or states either.
- **[Transition law]**: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{a}_t + \mathbf{w}_t$, where the matrices A and B are either given or learned (identified).
- **[Cost]**: In the finite horizon case, we have:

$$\mathbb{E}\left[\mathbf{x}_N^T Q \mathbf{x}_N \sum_{t=0}^{N-1} (\mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{a}_t^T R \mathbf{a}_t)\right]$$

  where Q and R are spd.

In the imperfect state information case, we need a model describing the stochastic mapping between the states and the observations:

$$\mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}_t$$

where $\mathbf{v}_t$ is the observation noise, which obeys the same assumptions at $\mathbf{w}_t$.

The initial state $\mathbf{x}_0$ has prior distribution $p(\mathbf{x}_0)$.

If we define the **information vector** as $I_t = (\mathbf{y}_{0:t}, \mathbf{a}_{1:t-1}) \in \mathcal{I}_t$, then the admissible decisions are: $\mathbf{d}_t : \mathcal{I}_t \mapsto \mathcal{A}_t$.

We will see later that this problem admits a nice analytical solution, where the problems of state inference and control are separable.

◇ OPTIMAL STOPPING

Optimal stopping models have many fascinating applications ranging from deciding how many people you should date before committing (**the secretary problem**), to deciding when to sell an asset, when to exercise a **call option**, when to stop trying new drugs (treatments), or when to stop asking questions. Moreover, optimal stopping problems often have analytical solution. In general, they have the following structure:

- **[States]**: The state space has two components $\mathcal{X} = \mathcal{C} \cup \mathcal{T}$, where $\mathcal{C}$ is a continuation space and $\mathcal{T}$ is a termination space. That is, the state evolves according to a Markov chain and at each decision period one has to choose to continue or stop.
- **[Actions]**: $\mathbf{a}_t \in \{c, t\}$ if $\mathbf{x}_t \in \mathcal{C}$. No actions are allowed once the process has terminated.

- **[Transition law]**: The process evolves according to a Markov model

$$p_t(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t) = \begin{cases} p_t(\mathbf{x}_{t+1}|\mathbf{x}_t) & \text{if } \mathbf{x}_t \in \mathcal{C}, \mathbf{x}_{t+1} \in \mathcal{C}, \mathbf{a}_t = c \\ 1 & \text{if } \mathbf{x}_t \in \mathcal{C}, \mathbf{x}_{t+1} \in \mathcal{T}, \mathbf{a}_t = t \end{cases}$$

- **[Cost]**: There is a reward $g_t(\mathbf{x}_t)$ for quitting at time t and a cost $f_t(\mathbf{x}_t)$ for continuing. There might also be a terminal reward r_N in the finite horizon setting.

We will find a solution to the dating (secretary) problem soon.

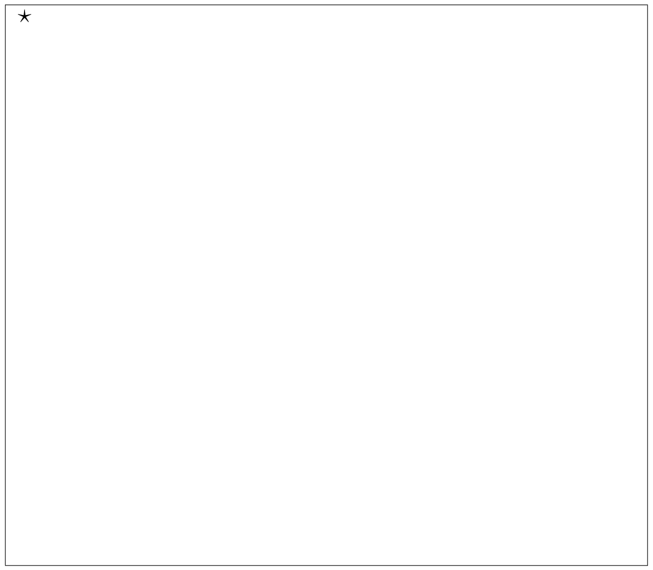
◇ ROUTING

Routing is as **shortest path problem**. Here, the action determines the subsequent state with certainty. That is, we have a **deterministic dynamic program (DDP)**, where the transition probabilities are replaced by **transfer**

functions $\tau : (\mathcal{X}, \mathcal{A}) \mapsto \mathcal{X}$. In particular,

$$p_t(\mathbf{x}_{t+1} = j | \mathbf{x}_t, \mathbf{a}_t) = \begin{cases} 1 & \text{if } \tau_t(\mathbf{x}_t, \mathbf{a}_t) = j \\ 0 & \text{if } \tau_t(\mathbf{x}_t, \mathbf{a}_t) \neq j \end{cases}$$

Here is a directed graph example:



◇ SEQUENTIAL ALLOCATION

Assume we have M units of resource (say wealth or energy) over N decision steps. Let the action be the non-negative amount consumed at each step:

$$\mathbf{a}_1 + \cdots + \mathbf{a}_N = M$$

The objective is to maximize some separable reward function:

$$f(\mathbf{a}_{1:N}) = \sum_{t=1}^{N} r_t(\mathbf{a}_t)$$

Formally we have:

- **[States]**: $\mathbf{x}_t \in [0, M]$ is what is available for consumption at time t.
- **[Actions]**: $\mathbf{a}_t \in [0, \mathbf{x}_t]$ is what is consumed at time t.

- **[Transition law]**:

$$p_t(\mathbf{x}_{t+1} = j | \mathbf{x}_t, \mathbf{a}_t) = \begin{cases} 1 & \text{if } \; j = \mathbf{x}_t - \mathbf{a}_t \\ 0 & \text{if } \; \text{otherwise} \end{cases}$$

- **[reward]**: $r_t(\mathbf{x}_t, \mathbf{a}_t) = r_t(\mathbf{a}_t)$