Roadmap

- Control
  - Feedback control
  - Adaptive control
- Adaptive optimal control
  - Model-free RL
  - Model-based RL
- Optimal and learning control
  - Open-loop
    - Indirect methods
    - Direct methods
  - MPC
  - Closed-loop
    - DP
    - HJB / HJI
Outline

Stochastic Optimal Control: Markov Decision Process (MDP)

The dynamic programming algorithm (stochastic case)

Stochastic LQR

Infinite-Horizon MDPs:
• Exact Methods:
  • (Policy Evaluation)
  • Value Iteration
  • Policy Iteration
Stochastic Optimal Control Problem: Markov Decision Problem (MDP)

- **System**: \( x_{k+1} = f_k (x_k, u_k, w_k), k = 0, \ldots, N - 1 \)
- **Probability distribution**: \( w_k \sim P_k (\cdot | x_k, u_k) \)
- **Control constraints**: \( u_k \in U (x_k) \)
- **Policies**: \( \pi = \{ \pi_0, \ldots, \pi_{N-1} \} \), where \( u_k = \pi_k (x_k) \)
- **Expected Cost**:

\[
J_{\pi} (x_0) = \mathbb{E}_{w_k, k=0, \ldots, N-1} \left[ g_N (x_N) + \sum_{k=0}^{N-1} g_k (x_k, \pi_k (x_k), w_k) \right]
\]

Stochastic Optimal Control Problem:

\[
J^* (x_0) = \min_{\pi} J_{\pi} (x_0)
\]
Key points

• Discrete-time model
• Markovian model
• Objective: find optimal \textit{closed-loop} policy
• Additive cost (central assumption in DP)
• Risk-neutral formulation

Other communities use different notation:
[Powell, W. B. \textit{AI, OR and control theory: A Rosetta Stone for stochastic optimization}. Princeton University, 2012.]
Principle of optimality (stochastic case)

**Principle of optimality:**
- Let $\pi^* := \{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\}$ be an optimal policy
- Consider the tail subproblem $\mathbb{E}_{w_k} \left[ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \pi_k(x_k), w_k) \right]$

the tail policy $\{\pi_i^*, \ldots, \pi_{N-1}^*\}$ is optimal for the tail subproblem

**Intuition:**
- DP first solves ALL tail subproblems at the final stage
- At the generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length
DP Algorithm (stochastic case)

Like in the deterministic case, start with:

\[ J^*_N (x_N) = g_N (x_N) \]

and iterate backwards in time using

\[ J^*_k (x_k) = \min_{u_k \in U(x_k)} \mathbb{E}_{w_k} \left[ g_k (x_k, u_k, w_k) + J^*_{k+1} \left( f (x_k, u_k, w_k) \right) \right], \quad k = 0, \ldots, N - 1 \]

for which the optimal cost \( J^*(x_0) \) is equal to \( J_0(x_0) \) and the optimal policy is constructed by setting

\[ \pi^*_k (x_k) = \arg\min_{u_k \in U(x_k)} \mathbb{E}_{w_k} \left[ g_k (x_k, u_k, w_k) + J^*_{k+1} \left( f (x_k, u_k, w_k) \right) \right] \]
Example: Inventory Control Problem

\( x_k \in \mathbb{N} \): stock available
\( u_k \in \mathbb{N} \): inventory
\( w_k \in \mathbb{N} \): demand

Dynamics: \( x_{k+1} = \max(0, x_k + u_k - w_k) \)

Constraints: \( x_k + u_k \leq 2 \)

Probabilistic structure:
\[
\begin{align*}
p(w_k = 0) &= 0.1 \\
p(w_k = 1) &= 0.7 \\
p(w_k = 2) &= 0.2
\end{align*}
\]

Objective:
\[
\mathbb{E}_{w_k} \left[ 0 + \sum_{k=0}^{2} \left( u_k + (x_k + u_k - w_k)^2 \right) \right]
\]

More generally, could imagine costs:
\( H(x_k) \): holding inventory
\( B(u_k) \): buying inventory
\( S(x_k, u_k, w_k) \): selling (matching stock with demand)
Example: Inventory Control Problem

Algorithm takes the form

\[ J^*_k(x_k) = \min_{0 \leq u_k \leq 2-x_k} \mathbb{E}_{w_k} \left[ u_k + (x_k + u_k - w_k)^2 + J^*_{k+1} \left( \max \left( 0, x_k + u_k - w_k \right) \right) \right] \]

for \( k = 0, 1, 2 \)

For example

\[ J^*_2(0) = \min_{u_2=0,1,2} \mathbb{E}_{w_2} \left[ u_2 + (u_2 - w_2)^2 \right] = \]

\[ \min_{u_2=0,1,2} u_2 + 0.1 (u_2)^2 + 0.7 (u_2 - 1)^2 + 0.2 (u_2 - 2)^2 \]

Which yields \( J^*_2(0) = 1.3 \) and \( \pi^*_2(0) = 1 \)
Example: Inventory Control Problem

Final solution:

\[ J^*_0(0) = 3.7 \]
\[ J^*_0(1) = 2.7 \]
\[ J^*_0(2) = 2.818 \]

(See this spreadsheet)
Stochastic LQR

Find control policy that minimizes

$$\mathbb{E}_{w_k} \left[ \frac{1}{2} x_N^T Q x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k) \right]$$

Subject to

- Dynamics $x_{k+1} = A_k x_k + B_k u_k + w_k$, $k \in \{0,1,\ldots,N-1\}$

with $x_0 \sim \mathcal{N}(\bar{x}_0, \Sigma_{x_0}), \left\{ w_k \sim \mathcal{N}(0, \Sigma_{w_k}) \right\}$ independent and Gaussian vectors


**Stochastic LQR**

As in the deterministic case, with $J^*_{k+1}(x_{k+1}) = \frac{1}{2}x_{k+1}^T P_{k+1} x_{k+1}$

$$J^*_k(x_{k+1}) = \min_{u_k} \mathbb{E}_{w_k} \left[ g_k(x_k, u_k, w_k) + J^*_{k+1}(f(x_k, u_k, w_k)) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_{k+1}^T Q_{k+1} x_{k+1} + u_{k+1}^T R_{k+1} u_{k+1} + (A_k x_k + B_k u_k + w_k)^T P_{k+1} (A_k x_k + B_k u_k + w_k) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_{k+1}^T Q_{k+1} x_{k+1} + u_{k+1}^T R_{k+1} u_{k+1} + (A_k x_k + B_k u_k)^T P_{k+1} (A_k x_k + B_k u_k) \right.$$

$$\left. 2 (A_k x_k + B_k u_k)^T P_{k+1} w_k + w_{k+1}^T P_{k+1} w_k \right]$$

$$= \min_{u_k} \frac{1}{2} \left( x_{k+1}^T Q_{k+1} x_{k+1} + u_{k+1}^T R_{k+1} u_{k+1} + (A_k x_k + B_k u_k)^T P_{k+1} (A_k x_k + B_k u_k) + \text{tr} \left( P_{k+1} \Sigma w_k \right) \right)$$

- The optimal cost to go is increased by some constant related to the magnitude of the noise (on which we have no control on)
- The optimal policy is the same as in the deterministic case
Infinite Horizon MDPs

State: \( x \in \mathcal{X} \)
Action: \( u \in \mathcal{U} \)
Transition function / Dynamics: \( T(x_t \mid x_{t-1}, u_{t-1}) = p(x_t \mid x_{t-1}, u_{t-1}) \)
Reward function: \( r_t = R(x_t, u_t) : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \)
Discount factor: \( \gamma \in (0,1) \)
Stationary policy: \( u_t = \pi(x_t) \)

Goal: choose a policy that maximizes cumulative (discounted) reward

\[
\pi^* = \arg\max_{\pi} \mathbb{E}_p \left[ \sum_{t \geq 0} \gamma^t R(x_t, \pi(x_t)) \right]
\]

Typically represented as a tuple \( \mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma) \)
Value functions

State-value function: “the expected total reward if we start in that state and act accordingly to a particular policy”

Action-state value function: “the expected total reward if we start in that state, take an action, and act accordingly to a particular policy”

Optimal state-value function

\[ V^*(x) = \max_{\pi} V_\pi(x) \]

Optimal action-state value function

\[ Q^*(x, u) = \max_{\pi} Q_\pi(x, u) \]

\[ V_\pi(x) = \mathbb{E}_p \left[ \sum_{t \geq 0} \gamma^t R(x_t, \pi(x_t)) \right] \]

\[ Q_\pi(x, u) = \mathbb{E}_p \left[ \sum_{t \geq 0} \gamma^t R(x_t, u_t) \right] \]
Bellman Equations

Value functions can be decomposed into immediate reward plus discounted value of successor state

\[
V_\pi(x_t) = \mathbb E_\pi \left[ R(x_t, \pi(x_t)) + \gamma V_\pi(x_{t+1}) \right] \quad \text{Bellman Expectation Equation}
\]

\[
= R(x_t, \pi(x_t)) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} | x_t, \pi(x_t)) V_\pi(x_{t+1})
\]

Similarly, also optimal value function can be decomposed as:

\[
V^*(x_t) = \max_u \left( R(x_t, u_t) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} | x_t, u_t) V^*(x_{t+1}) \right) \quad \text{Bellman Optimality Equation}
\]
Three paradigms that rely on DP

For *prediction*:
- Policy Evaluation: “given a policy $\pi$, find the value function $V_{\pi}(x)$, i.e., how good is that policy?”

For *control*:
- Policy Iteration: leverages policy evaluation as an inner loop to find the optimal policy
- Value Iteration: applies Bellman’s optimality equation to compute the optimal value function
Policy Evaluation

**Problem:** evaluate a given policy $\pi$

**Solution:** iterative application of Bellman expectation backup ($V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_\pi$)

- At each iteration $k+1$
- For all states $x \in X$
- Update $V_{k+1}(x)$ from $V_k(x)$ through

**Bellman Expectation Equation**

$$V_{k+1}(x_t) = R(x_t, \pi(x_t)) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} \mid x_t, \pi(x_t)) V_k(x_{t+1})$$

- This sequence is proven to converge to $V_\pi$
Example: Grid World

From Sutton and Barto, Reinforcement Learning: An Introduction (Chapter 4)

- Nonterminal states 1, …, 14. Terminal states as shaded squared
- Reward is -1 until the terminal state is reached
- Controls leading out of the grid leave state unchanged
- Undiscounted MDP ($\gamma = 1$)
- We want to evaluate a uniform random policy
$V_k(x)$ for the random policy

$k = 0$

\[
\begin{array}{cccc}
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
\end{array}
\]

$k = 1$

\[
\begin{array}{cccc}
0.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & 0.0 \\
\end{array}
\]

$k = 2$

\[
\begin{array}{cccc}
0.0 & -1.7 & -2.0 & -2.0 \\
-1.7 & -2.0 & -2.0 & -2.0 \\
-2.0 & -2.0 & -2.0 & -1.7 \\
-2.0 & -2.0 & -1.7 & 0.0 \\
\end{array}
\]

Greedy policy w.r.t. $V_k(x)$

random policy
### $V_k(x)$ for the random policy

<table>
<thead>
<tr>
<th></th>
<th>0.0</th>
<th>-2.4</th>
<th>-2.9</th>
<th>-3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 3$</td>
<td>-2.4</td>
<td>-2.9</td>
<td>-3.0</td>
<td>-2.9</td>
</tr>
<tr>
<td></td>
<td>-2.9</td>
<td>-3.0</td>
<td>-2.9</td>
<td>-2.4</td>
</tr>
<tr>
<td></td>
<td>-3.0</td>
<td>-2.9</td>
<td>-2.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### Greedy policy w.r.t. $V_k(x)$

The greedy policies for $k = 3$, $k = 10$, and $k = \infty$ are shown with arrows indicating the optimal policy direction.
Some technical questions

- How do we know that iterative policy evaluation converges to $V^\pi$?
- Is the solution unique?
- How fast does this algorithm converge?

These questions are resolved by the contraction mapping theorem

Sketch of proof:

- Def: $\infty$-norm $\|u - v\|_\infty = \max_{x \in \mathcal{X}} |u(x) - v(x)|$, i.e. the largest difference between state values
- Def: an update operation is a $\gamma$-contraction if $\|U_{i+1} - V_{i+1}\| \leq \|U_i - V_i\|, \forall U_i, V_i$
- Theorem: a $\gamma$-contraction converges to a unique fixed point, no matter the initialization, at a linear convergence rate of $\gamma$
- Fact: the policy evaluation operator is a $\gamma$-contraction in $\infty$-norm
- Corollary: policy evaluation converges to a unique fixed point
Policy Iteration

Given policy $\pi$

Evaluate the policy $\pi$

$$V_{k+1}(x_t) = R(x_t, \pi(x_t)) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} | x_t, \pi(x_t)) V_k(x_{t+1})$$

Improve the policy $\pi$ by acting greedily w.r.t. $V_\pi$

$$\pi_{k+1}(x) = \arg \max_u \left( R(x, u) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} | x_t, u_t) V_{k+1}(x_{t+1}) \right)$$

- In general, policy iteration requires more iterations of evaluation / improvement (in our small Grid World, one was sufficient)
- This process always converges to the optimal policy
Policy Improvement

- Given a deterministic policy $\pi(x)$
- We can improve the policy by acting greedily w.r.t. the current value function

$$\pi'(x) = \arg\max_{u \in \mathcal{U}} q_\pi(x, u)$$

- Consider the one step decision, where we use $\pi'$ for one step and then act accordingly to the old policy $\pi$

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

- If we repeat the same reasoning for all following steps, we can see how this improves the value function $v_\pi(x) \geq v_\pi(x)$
Value Iteration

Problem: find the optimal policy \( \pi^* \)

Solution: iterative application of Bellman optimality backup \((V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V^*)\)

- At each iteration \( k+1 \)
- For all states \( x \in X \)
- Update \( V_{k+1}(x) \) from \( V_k(x) \) through

\[
V_{k+1}^*(x_t) = \max_u \left[ R(x_t, u_t) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} | x_t, u_t) V_k^*(x_{t+1}) \right]
\]

Bellman Optimality Equation

- This sequence is proven to converge to \( V^* \)
Exercise from Pieter Abbeel, CS287

(a) Prefer the close exit (+1), risking the cliff (-10)  
(b) Prefer the close exit (+1), but avoiding the cliff (-10)  
(c) Prefer the distant exit (+10), risking the cliff (-10)  
(d) Prefer the distant exit (+10), avoiding the cliff (-10)

(1) $\gamma = 0.1$, noise = 0.5  
(2) $\gamma = 0.99$, noise = 0  
(3) $\gamma = 0.99$, noise = 0.5  
(4) $\gamma = 0.1$, noise = 0
## Recap

All of these formulations require a **model of the MDP!**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Bellman Equation</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>Bellman Expectation Equation</td>
<td>Iterative Policy Evaluation</td>
</tr>
<tr>
<td>Control</td>
<td>Bellman Expectation Equation + Greedy Policy Improvement</td>
<td>Policy Iteration</td>
</tr>
<tr>
<td>Control</td>
<td>Bellman Optimality Equation</td>
<td>Value Iteration</td>
</tr>
</tbody>
</table>
Outline

Stochastic Optimal Control: Markov Decision Process (MDP)

The dynamic programming algorithm (stochastic case)

Stochastic LQR

Infinite-Horizon MDPs:
• Exact Methods:
  • (Policy Evaluation)
  • Value Iteration
  • Policy Iteration
Next time

- Nonlinear LQR for tracking
- iLQR
- DDP