Lecture 02: Markov Decision Processes

Paul Swoboda

Preface

- Markov decision processes (MDP) are a mathematically idealized form of RL problems.
- They allow precise theoretical statements (e.g., on optimal solutions).
- They deliver insights into suitable RL algorithms since many real-world problems can be abstracted as MDPs.
- In the following we'll focus on:
  - fully observable MDPs (i.e., $\boldsymbol{s}_k = \boldsymbol{o}_k$) and
  - finite MDPs (i.e., finite number of states & actions).

		All states observable?	
		Yes	No
Actions?	No	Markov chain	Hidden Markov model
	Yes	Markov decision process (MDP)	Partially observable MDP

Tab. 1.1: Different Markov models

Scalar and Vectorial Representations in Finite MDPs

- The position of a chess piece can be represented in two ways:
  - Vectorial: $\boldsymbol{s} = \begin{bmatrix} s_h & s_v \end{bmatrix}^\mathsf{T}$, i.e., a two-element vector with horizontal and vertical information (e.g., $\boldsymbol{s} = \begin{bmatrix} c & 1 \end{bmatrix}^\mathsf{T}$) or
  - Scalar: simple enumeration of all available positions (e.g., $s = 3$).
- Both ways represent the same amount of information and can be easily converted into each other.
- For sake of readability we will stick to the scalar representation of states and actions in finite MDPs whenever possible.

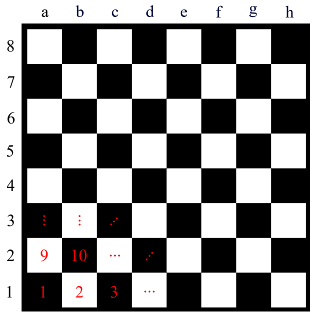


Table of Contents

Recap on Markov Property

Information state

A state S_k is called an information or Markov state if and only if

$$\mathbb{P}\left[S_{k+1}|S_k\right] = \mathbb{P}\left[S_{k+1}|S_0, S_1, \ldots, S_k\right] \,. \tag{1.1}$$

- History is fully condensed in the sample S_k, i.e., S_{k+1} is only depending on S_k.
- A given system can be fully described by S_k.
- Further past observations $S_{k-1}, S_{k-2}, \ldots$ are irrelevant.

State Transition Matrix

Definition 1.1: State transition matrix

Given a Markov state $S_k = s$ and its successor $S_{k+1} = s'$, the state transition probability $\forall \{s, s'\} \in \mathcal{S}$ is defined by the matrix

$$\boldsymbol{\mathcal{P}}_{ss'} = \mathbb{P}\left[S_{k+1} = s' | S_k = s\right]. \tag{1.2}$$

Here, $\boldsymbol{\mathcal{P}}_{ss'} \in \mathbb{R}^{n \times n}$ has the form

$$\boldsymbol{\mathcal{P}}_{ss'} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & & & \vdots \\ \vdots & & & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{bmatrix}$$

with $p_{ij} \in \{\mathbb{R} | 0 \leq p_{ij} \leq 1\}$ being the specific probability to go from state $s = S_i$ to state $s' = S_j$. Obviously, $\sum_j p_{ij} = 1 \, \forall i$ must hold.

Markov Chain

Definition 1.2: Finite Markov chain

A finite Markov chain is a tuple $\langle \mathcal{S}, \boldsymbol{\mathcal{P}} \rangle$ with

- $\mathcal{S}$ being a finite set of discrete-time states $S_k \in \mathcal{S}$,
- $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}_{ss'} = \mathbb{P}\left[S_{k+1} = s' | S_k = s\right]$ is the state transition probability.

- Specific stochastic process model
- Sequence of random variables $S_k, S_{k+1}, \ldots$
- 'Memoryless'
- In continuous-time framework: Markov process[1]

[1] However, this results in a literature nomenclature inconsistency with Markov decision/reward 'processes'.

Chapman-Kolmogorov Equation

Define $\boldsymbol{p}_k \in \mathbb{R}^n$ as a probability row vector where $p_{i,k}$ gives the probability of being in state $S_k \in \mathcal{S}$ at time-step k.

Definition 1.3: Chapman-Kolmogorov equation for finite Markov chains

The probability of being in state S_{k+m} at time-step $(k+m)$ starting from state s_k is given by:

$$\boldsymbol{p}_{k+m} = \boldsymbol{p}_k \boldsymbol{\mathcal{P}}_{ss'}^m. \tag{1.3}$$

Hence, for $m \to \infty$ ('*steady state*') the following equation must hold:

$$\boldsymbol{p} = \boldsymbol{p} \boldsymbol{\mathcal{P}}_{ss'}. \tag{1.4}$$

Solving (1.4) for $\boldsymbol{p}$ given the transition probability $\boldsymbol{\mathcal{P}}_{ss'}$ gives insights which states of the Markov chain are visited in the long-run.

Example of a Markov Chain (1)

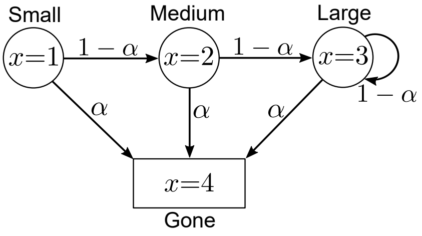


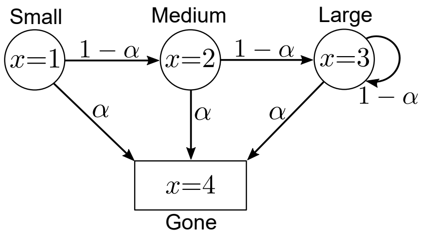
Fig. 1.1: Forest tree Markov chain

$$
\begin{aligned}
s = s &\in \{1, 2, 3, 4\} \\
&= \{\text{small}, \text{medium}, \text{large}, \text{gone}\}
\end{aligned}
$$

$$
\boldsymbol{\mathcal{P}} = \begin{bmatrix} 0 & 1-\alpha & 0 & \alpha \\ 0 & 0 & 1-\alpha & \alpha \\ 0 & 0 & 1-\alpha & \alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

- At $s = 1$ a small tree is planted ('starting point').
- A tree grows with $(1 - \alpha)$ probability.
- If it reaches $s = 3$ (large) its growth is limited.
- With α probability a natural hazard destroys the tree.
- The state $s = 4$ is terminal ('infinite loop').

Example of a Markov Chain (2)



Possible samples for the given Markov chain example starting from $s = 1$ (small tree):

- Small $\rightarrow$ gone
- Small $\rightarrow$ medium $\rightarrow$ gone
- Small $\rightarrow$ medium $\rightarrow$ large $\rightarrow$ gone
- Small $\rightarrow$ medium $\rightarrow$ large $\rightarrow$ large $\rightarrow \dots$

Example of a Markov Chain (3)

For $k = 0$ the initial state probability is given $\boldsymbol{p}_{k=0} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$, i.e., we are in state $s = 1$. What is the state probability after $k = 1, 2, \ldots$ steps with $\alpha = 0.2$?

$$\boldsymbol{p}_1 = \begin{bmatrix} 0 & 0.8 & 0 & 0.2 \end{bmatrix} \quad = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{p}_0} \underbrace{\begin{bmatrix} 0 & 0.8 & 0 & 0.2 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{\mathcal{P}}},$$

$$\boldsymbol{p}_2 = \begin{bmatrix} 0 & 0 & 0.64 & 0.36 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0.8 & 0 & 0.2 \end{bmatrix}}_{\boldsymbol{p}_1} \underbrace{\begin{bmatrix} 0 & 0.8 & 0 & 0.2 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{\mathcal{P}}},$$

$$\boldsymbol{p}_3 = \begin{bmatrix} 0 & 0 & 0.51 & 0.49 \end{bmatrix}, \quad \boldsymbol{p}_{10} = \begin{bmatrix} 0 & 0 & 0.11 & 0.89 \end{bmatrix},$$
$$\boldsymbol{p}_\infty = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.$$

Table of Contents

Markov Reward Process

Definition 1.4: Finite Markov reward process

A finite Markov reward process (MRP) is a tuple $\langle \mathcal{S}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$ with

- $\mathcal{S}$ being a finite set of discrete-time states $S_k \in \mathcal{S}$,
- $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}_{ss'} = \mathbb{P}\left[S_{k+1} = s' | S_k = s\right]$ is the state transition probability,
- $\mathcal{R}$ is a reward function, $\mathcal{R} = \mathcal{R}_s = \mathbb{E}\left[R_{k+1} | S_k = s_k\right]$ and
- γ is a discount factor, $\gamma \in \{\mathbb{R} | 0 \leq \gamma \leq 1\}$.

- Markov chain extended with rewards
- Still an autonomous stochastic process without specific inputs
- Rewards R_k only dependent on state S_k

Example of a Markov Reward Process

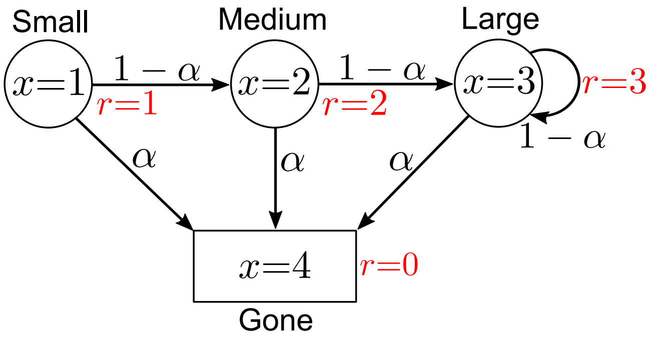


Fig. 1.2: Forest Markov reward process

- Growing larger trees is rewarded, since it will be
  - appreciated by hikers and
  - useful for wood production.
- Loosing a tree due to a hazard is unrewarded.

Recap on Return

Return

The return G_k is the total discounted reward starting from step k onwards. For episodic tasks it becomes the finite series

$$G_k = R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \cdots = \sum_{i=0}^{N} \gamma^i R_{k+i+1} \tag{1.5}$$

terminating at step N while it is an infinite series for continuing tasks

$$G_k = R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \cdots = \sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \,. \tag{1.6}$$

- The discount γ represents the value of future rewards.

Remark on Notation: Episodic and Continuing Tasks

- In episodic tasks there is not only one (long/infinite) sequence of time steps.
- Instead there is a sequence of time steps of an arbitrary episode j.
  - Formally: $S_{k,j}, A_{k,j}, R_{k,j}, \ldots$
- Nevertheless, to increase the readability the episode index is normally dropped in this course:
  - Simplified: $S_k \leftarrow S_{k,j}, A_k \leftarrow A_{k,j}, R_k \leftarrow R_{k,j}, \ldots$
- Only in certain cases the episode indexing will be re-activated.

Fig. 1.3: RL in practice (source: Vlad Sargu on Unsplash)

Value Function in MRP

Definition 1.5: Value function in MRP

The state-value function $v(s_k)$ of an MRP is the expected return starting from state s_k

$$v(s_k) = \mathbb{E}\left[G_k | S_k = s_k\right]. \tag{1.7}$$

- Represents the long-term value of being in state S_k.

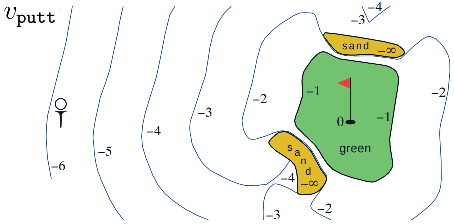
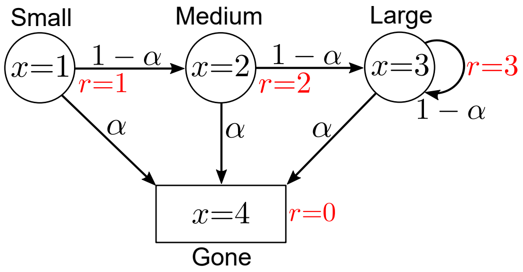


Fig. 1.4: Isolines indicate state value of different golf ball locations (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)

State-Value Samples of Forest MRP



Exemplary samples $\hat{v}$ with $\gamma = 0.5$ starting in $s = 1$:

$$
\begin{aligned}
s &= 1 \to 4, & \hat{v} &= 1, \\
s &= 1 \to 2 \to 4, & \hat{v} &= 1 + 0.5 \cdot 2 = 2.0, \\
s &= 1 \to 2 \to 3 \to 4, & \hat{v} &= 1 + 0.5 \cdot 2 + 0.25 \cdot 3 = 3.75, \\
s &= 1 \to 2 \to 3 \to 3 \to 4, & \hat{v} &= 1 + 0.5 \cdot 2 + 0.25 \cdot 3 + 0.125 \cdot 3 = 4.13.
\end{aligned}
$$

Bellman Equation for MRPs (1)

Problem: How to calculate all state values in closed form?
Solution: Bellman equation.

$$
\begin{aligned}
v(s_k) &= \mathbb{E}\left[G_k | S_k = s_k\right] \\
&= \mathbb{E}\left[R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \ldots | S_k = s_k\right] \\
&= \mathbb{E}\left[R_{k+1} + \gamma\left(R_{k+2} + \gamma R_{k+3} + \ldots\right) | S_k = s_k\right] \\
&= \mathbb{E}\left[R_{k+1} + \gamma G_{k+1} | S_k = s_k\right] \\
&= \mathbb{E}\left[R_{k+1} + \gamma v(S_{k+1}) | S_k = s_k\right]
\end{aligned}
\tag{1.8}
$$

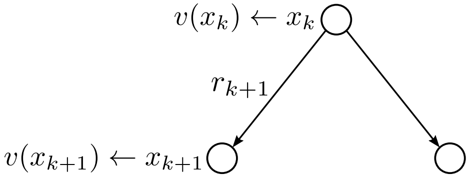


Fig. 1.5: Backup diagram for $v(s_k)$

Bellman Equation for MRPs (2)

Assuming non-random rewards for every state $S = s \in \mathcal{S}$

$$\boldsymbol{r}_{\mathcal{S}} = \begin{bmatrix}\mathcal{R}(s_1) & \cdots & \mathcal{R}(s_n)\end{bmatrix}^{\mathsf{T}} = \begin{bmatrix}\mathcal{R}_1 & \cdots & \mathcal{R}_n\end{bmatrix}^{\mathsf{T}} \tag{1.9}$$

for a finite number of n states with state-values

$$\boldsymbol{v}_{\mathcal{S}} = \begin{bmatrix}v(s_1) & \cdots & v(s_n)\end{bmatrix}^{\mathsf{T}} = \begin{bmatrix}v_1 & \cdots & v_n\end{bmatrix}^{\mathsf{T}} \tag{1.10}$$

one can derive a linear equation system based on Fig. 1.5:

$$\begin{gathered}\boldsymbol{v}_{\mathcal{S}} = \boldsymbol{r}_{\mathcal{S}} + \gamma \boldsymbol{\mathcal{P}}_{ss'} \boldsymbol{v}_{\mathcal{S}}, \\ \begin{bmatrix}v_1\\ \vdots \\ v_n\end{bmatrix} = \begin{bmatrix}\mathcal{R}_1\\ \vdots \\ \mathcal{R}_n\end{bmatrix} + \gamma \begin{bmatrix}p_{11} & \cdots & p_{1n}\\ \vdots & & \vdots \\ p_{n1} & \cdots & p_{nn}\end{bmatrix} \begin{bmatrix}v_1\\ \vdots \\ v_n\end{bmatrix}.\end{gathered} \tag{1.11}$$

Solving the MRP Bellman Equation

Above, (1.11) is a normal equation in $\boldsymbol{v}_{\mathcal{S}}$:

$$\begin{aligned} \boldsymbol{v}_{\mathcal{S}} &= \boldsymbol{r}_{\mathcal{S}} + \gamma \boldsymbol{\mathcal{P}}_{ss'} \boldsymbol{v}_{\mathcal{S}}, \\ \Leftrightarrow \underbrace{(\boldsymbol{I} - \gamma \boldsymbol{\mathcal{P}}_{ss'})}_{\boldsymbol{A}} \underbrace{\boldsymbol{v}_{\mathcal{S}}}_{s} &= \underbrace{\boldsymbol{r}_{\mathcal{S}}}_{\boldsymbol{b}}. \end{aligned} \tag{1.12}$$

Possible solutions:

- Direct inversion (Gaussian elimination, $\mathcal{O}(n^3)$)
- Matrix decomposition (QR, Cholesky, etc. , $\mathcal{O}(n^3)$)
- Iterative solutions (e.g., Krylov-subspaces, often better than $\mathcal{O}(n^3)$)

In RL identifying and solving (1.12) is a key task, which is often realized only approximately for high-order state spaces.

Example of a Markov Reward Process with State Values

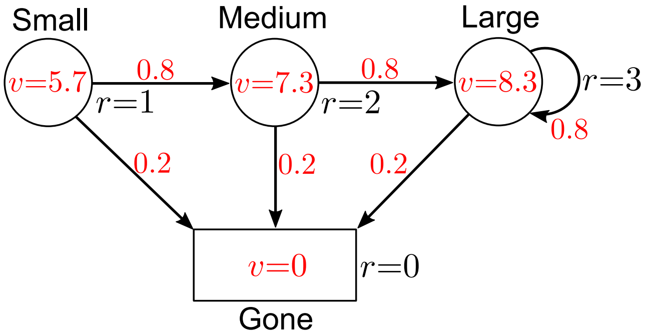


Fig. 1.6: Forest Markov reward process including state values

- Discount factor $\gamma = 0.8$
- Disaster probability $\alpha = 0.2$

Table of Contents

Markov Decision Process

Definition 1.6: Finite Markov decision process

A finite Markov decision process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$ with

- $\mathcal{S}$ being a finite set of discrete-time states $S_k \in \mathcal{S}$,
- $\mathcal{A}$ as a finite set of discrete-time actions $A_k \in \mathcal{A}$,
- $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}_{ss'}^{u}$ is the state transition probability $\boldsymbol{\mathcal{P}} = \mathbb{P}\left[S_{k+1} = s' | S_k = s_k, A_k = a_k\right]$,
- $\mathcal{R}$ is a reward function, $\mathcal{R} = \mathcal{R}_s^u = \mathbb{E}\left[R_{k+1} | S_k = s_k, A_k = a_k\right]$ and
- γ is a discount factor, $\gamma \in \{\mathbb{R} | 0 \leq \gamma \leq 1\}$.

- Markov reward process is extended with actions / decisions.
- Now, rewards also depend on action A_k.

Example of a Markov Decision Process (1)

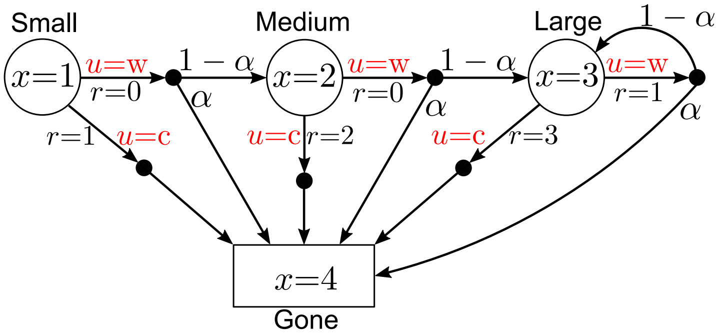


Fig. 1.7: Forest Markov decision process

- Two actions possible in each state:
  - Wait $u = w$: let the tree grow.
  - Cut $u = c$: gather the wood.
- With increasing tree size the wood reward increases as well.

Example of a Markov Decision Process (2)

For the previous example the state transition probability matrix and reward function are given as:

$$\boldsymbol{\mathcal{P}}_{ss'}^{u=c} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\mathcal{P}}_{ss'}^{u=w} = \begin{bmatrix} 0 & 1-\alpha & 0 & \alpha \\ 0 & 0 & 1-\alpha & \alpha \\ 0 & 0 & 1-\alpha & \alpha \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{r}_{\mathcal{S}}^{u=c} = \begin{bmatrix} 1 & 2 & 3 & 0 \end{bmatrix}^{\mathsf{T}}, \quad \boldsymbol{r}_{\mathcal{S}}^{u=w} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}}.$$

- The term $\boldsymbol{r}_{\mathcal{S}}^{u}$ is the abbreviated form for receiving the output of $\mathcal{R}$ for the entire state space $\mathcal{S}$ given the action u.

Policy (1)

Definition 1.7: Policy in MDP (1)

In an MDP environment, a policy is a distribution over actions given states:

$$\pi(a_k|s_k) = \mathbb{P}\left[A_k = a_k|S_k = s_k\right] . \tag{1.13}$$

- In MDP, policies depend only on the current state.
- A policy fully defines the agent's behavior.
- Might relax to a deterministic policy for certain applications.

Fig. 1.8: What is your best Monopoly policy? (source: Ylanite Koppens on Pexels)

Policy (2)

Given an MDP $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{\mathcal{P}}, \mathcal{R}, \gamma \rangle$ and a policy $\boldsymbol{\pi}$:

- The state sequence $S_k, S_{k+1}, \ldots$ is a Markov chain $\langle \mathcal{S}, \boldsymbol{\mathcal{P}}^{\pi} \rangle$ since the state transition probability is only depending on the state:

$$\boldsymbol{\mathcal{P}}_{ss'}^{\pi} = \sum_{a_k \in \mathcal{A}} \boldsymbol{\pi}(a_k|s_k) \boldsymbol{\mathcal{P}}_{ss'}^{u} \,. \tag{1.14}$$

- Consequently, the sequence $S_k, R_{k+1}, S_{k+1}, R_{k+2}, \ldots$ of states and rewards is a Markov reward process $\langle \mathcal{S}, \boldsymbol{\mathcal{P}}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$:

$$\mathcal{R}_{ss'}^{\pi} = \sum_{a_k \in \mathcal{A}} \boldsymbol{\pi}(a_k|s_k) \mathcal{R}_{s}^{u} \,. \tag{1.15}$$

Recap on MDP Value Functions

Definition 1.8: State-value function

The state-value function of an MDP is the expected return starting in s_k following policy π:

$$v_\pi(s_k) = \mathbb{E}_\pi\left[G_k | S_k = s_k\right] = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \middle| S_k\right].$$

Definition 1.9: Action-value function

The action-value function of an MDP is the expected return starting in s_k taking action a_k following policy π:

$$q_\pi(s_k, a_k) = \mathbb{E}_\pi\left[G_k | S_k = s_k, A_k = a_k\right] = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \middle| S_k, A_k\right].$$

Bellman Expectation Equation (1)

Analog to (1.8), the state value of an MDP can be decomposed into a Bellman notation:

$$v_\pi(s_k) = \mathbb{E}_\pi\left[R_{k+1} + \gamma v_\pi(S_{k+1}) | S_k = s_k\right] . \tag{1.16}$$

In finite MDPs the state value can be directly linked to the action value (cf. Fig. 1.9):

$$v_\pi(s_k) = \sum_{a_k \in \mathcal{A}} \pi(a_k|s_k) q_\pi(s_k, a_k) . \tag{1.17}$$

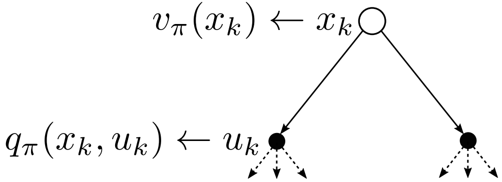


Fig. 1.9: Backup diagram for $v_\pi(s_k)$

Bellman Expectation Equation (2)

Likewise, the action value of an MDP can be decomposed into a Bellman notation:

$$q_\pi(s_k, a_k) = \mathbb{E}_\pi \left[R_{k+1} + \gamma q_\pi(S_{k+1}, A_{k+1}) | S_k = s_k, A_k = u_k\right] . \qquad (1.18)$$

In finite MDPs the action value can be directly linked to the state value (cf. Fig. 1.10):

$$q_\pi(s_k, a_k) = \mathcal{R}_s^a + \gamma \sum_{s_{k+1} \in \mathcal{S}} p_{ss'}^a v_\pi(s_{k+1}) . \qquad (1.19)$$

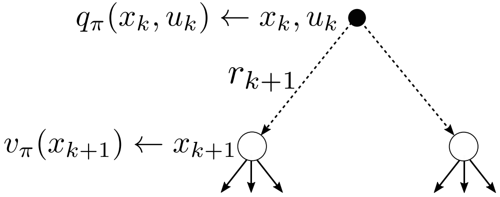

$q_\pi(x_k, u_k) \leftarrow x_k, u_k$
r_{k+1}
$v_\pi(x_{k+1}) \leftarrow x_{k+1}$

Fig. 1.10: Backup diagram for $q_\pi(s_k, a_k)$

Bellman Expectation Equation (3)

Inserting (1.19) into (1.17) directly results in:

$$v_\pi(s_k) = \sum_{a_k \in \mathcal{A}} \pi(a_k|s_k) \left(\mathcal{R}_s^a + \gamma \sum_{s_{k+1} \in \mathcal{S}} p_{ss'}^a v_\pi(s_{k+1}) \right) . \tag{1.20}$$

Conversely, the action value becomes:

$$q_\pi(s_k, a_k) = \mathcal{R}_s^a + \gamma \sum_{s_{k+1} \in \mathcal{S}} p_{ss'}^a \left(\sum_{a_{k+1} \in \mathcal{A}} \pi(a_{k+1}|s_{k+1}) q_\pi(s_{k+1}, a_{k+1}) \right) . \tag{1.21}$$

Bellman Expectation Equation in Matrix Form

Given a policy π and following the same assumptions as for (1.11), the Bellman expectation equation can be expressed in Matrix form:

$$\boldsymbol{v}_{\mathcal{S}}^{\pi} = \boldsymbol{r}_{\mathcal{S}}^{\pi} + \gamma \boldsymbol{\mathcal{P}}_{ss'}^{\pi} \boldsymbol{v}_{\mathcal{S}}^{\pi},$$

$$\begin{bmatrix} v_1^{\pi} \\ \vdots \\ v_n^{\pi} \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1^{\pi} \\ \vdots \\ \mathcal{R}_n^{\pi} \end{bmatrix} + \gamma \begin{bmatrix} p_{11}^{\pi} & \cdots & p_{1n}^{\pi} \\ \vdots & & \vdots \\ p_{n1}^{\pi} & \cdots & p_{nn}^{\pi} \end{bmatrix} \begin{bmatrix} v_1^{\pi} \\ \vdots \\ v_n^{\pi} \end{bmatrix}. \tag{1.22}$$

Here, $\boldsymbol{r}_{\mathcal{S}}^{\pi}$ and $\boldsymbol{\mathcal{P}}_{ss'}^{\pi}$ are the rewards and state transition probability following policy π. Hence, the state value can be calculated by solving (1.22) for $\boldsymbol{v}_{\mathcal{S}}^{\pi}$, e.g., by direct matrix inversion:

$$\boldsymbol{v}_{\mathcal{S}}^{\pi} = \left(\boldsymbol{I} - \gamma \boldsymbol{\mathcal{P}}_{ss'}^{\pi}\right)^{-1} \boldsymbol{r}_{\mathcal{S}}^{\pi}. \tag{1.23}$$

Bellman Expectation Equation & Forest Tree Example (1)

Let's assume following very simple policy ('*fifty-fifty*')

$$\pi(u=\text{cut}|s)=0.5,\quad \pi(u=\text{wait}|s)=0.5\quad \forall s\in\mathcal{S}\,.$$

Applied to the given environment behavior

$$\boldsymbol{\mathcal{P}}_{ss'}^{u=c}=\begin{bmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\\0&0&0&1\end{bmatrix},\quad \boldsymbol{\mathcal{P}}_{ss'}^{u=w}=\begin{bmatrix}0&1-\alpha&0&\alpha\\0&0&1-\alpha&\alpha\\0&0&1-\alpha&\alpha\\0&0&0&1\end{bmatrix},$$

$$\boldsymbol{r}_{\mathcal{S}}^{u=c}=\begin{bmatrix}1&2&3&0\end{bmatrix}^\mathsf{T},\quad \boldsymbol{r}_{\mathcal{S}}^{u=w}=\begin{bmatrix}0&0&1&0\end{bmatrix}^\mathsf{T},$$

one receives:

$$\boldsymbol{\mathcal{P}}_{ss'}^{\pi}=\begin{bmatrix}0&\frac{1-\alpha}{2}&0&\frac{1+\alpha}{2}\\0&0&\frac{1-\alpha}{2}&\frac{1+\alpha}{2}\\0&0&\frac{1-\alpha}{2}&\frac{1+\alpha}{2}\\0&0&0&1\end{bmatrix},\quad \boldsymbol{r}_{\mathcal{S}}^{\pi}=\begin{bmatrix}0.5\\1\\2\\0\end{bmatrix}.$$

Bellman Expectation Equation & Forest Tree Example (2)

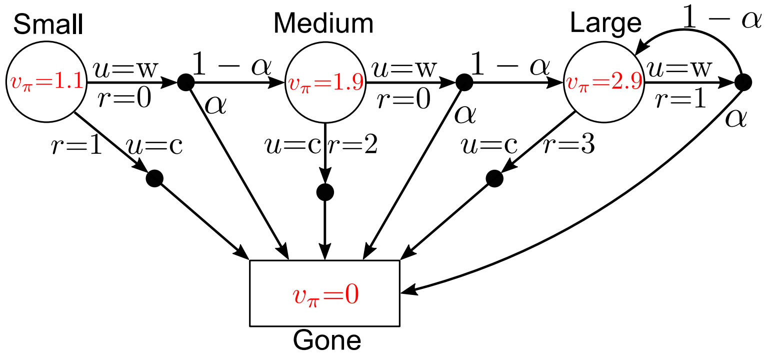


Fig. 1.11: Forest MDP with fifty-fifty-policy including state values

- Discount factor $\gamma = 0.8$
- Disaster probability $\alpha = 0.2$

Bellman Expectation Equation & Forest Tree Example (3)

Using the Bellman expectation eq. (1.19) the action values can be directly calculated:

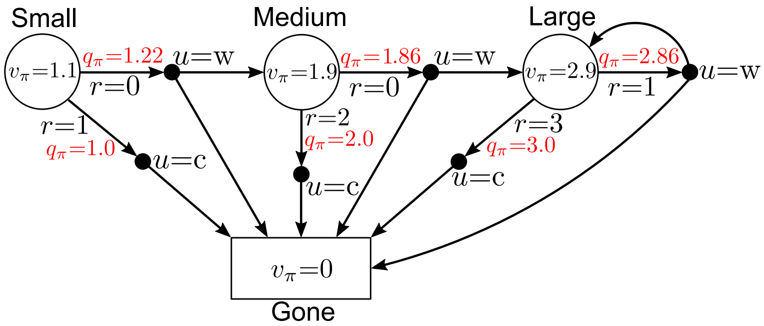


Fig. 1.12: Forest MDP with fifty-fifty-policy plus action values

Is the '*fifty-fifty-policy*' maximizing our long-term reward? How can you evaluate your answer?

Table of Contents

Optimal Value Functions in an MDP

Definition 1.10: Optimal state-value function

The optimal state-value function of an MDP is the maximum state-value function over all polices:

$$v^*(s) = \max_\pi v_\pi(s)\,. \tag{1.24}$$

Definition 1.11: Optimal action-value function

The optimal action-value function of an MDP is the maximum action-value function over all polices:

$$q^*(s,u) = \max_\pi q_\pi(s,u)\,. \tag{1.25}$$

- The optimal value function denotes the best possible agent's performance for a given MDP / environment.
- A (finite) MDP can be easily solved in an optimal way if $q^*(s,u)$ is known.

Optimal Policy in an MDP

Define a partial ordering over polices

$$\pi \geq \pi' \quad \text{if} \quad v_\pi(s) \geq v_{\pi'}(s) \quad \forall s \in \mathcal{S}\,. \tag{1.26}$$

Theorem 1.1: Optimal policies in MDPs

For any finite MDP

- there exists an optimal policy $\pi^* \geq \pi$ that is better or equal to all other policies,
- all optimal policies achieve the same optimal state-value function $v^*(s) = v_{\pi^*}(s)$,
- all optimal policies achieve the same optimal action-value function $q^*(s,u) = q_{\pi^*}(s,u)$.

Proof for Optimal Policy in an MDP: Lemma

Lemma

Let π and π' be two policies and assume that for all $s \in \mathcal{S}$ we have $q_\pi(s, \pi'(s)) \geq v_\pi(s)$. Then the policy π' is as good as or better than π.

Proof.

For simplicity we assume deterministic policies. Then

$$
\begin{aligned}
v_\pi(s) \leq & q_\pi(s, \pi'(s)) \\
= & \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
= & E_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
\leq & E_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
= & E_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
= & E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s] \\
\leq & E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) | S_t = s] \\
\ldots & \\
= & v_{\pi'}(s)
\end{aligned}
$$

Proof for Optimal Policy in an MDP: Continuation

Optimal policy: Construct iteratively $\pi'(s) = \arg\max_a q_\pi(s,a)$. Since there are only finitely many policies, we will converge ultimately. The policy will fulfill (1.30).

Bellman Optimality Equation (1)

Theorem 1.2: Bellman's principle of optimality

"*An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*" (R.E. Bellman, Dynamic Programming, 1957)

- Any policy (i.e., also the optimal one) must satisfy the self-consistency condition given by the Bellman expectation equation.
- An optimal policy must equal the expected return for the best action of a given state:

$$\begin{aligned} v^*(s_k) &= \max_u q_{\pi^*}(s_k, a)\,, \\ &= \max_a \mathbb{E}_{\pi^*}\left[G_k | S_k = s_k, A = a\right]\,, \\ &= \max_a \mathbb{E}_{\pi^*}\left[R_{k+1} + \gamma G_{k+1} | S_k = s_k, A = a\right]\,, \\ &= \max_a \mathbb{E}\left[R_{k+1} + \gamma v^*(S_{k+1}) | S_k = s_k, A = a\right]\,. \end{aligned} \tag{1.27}$$

Bellman Optimality Equation (2)

Again, the Bellman optimality equation can be visualized by a backup diagram:

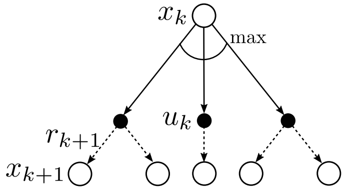


Fig. 1.13: Backup diagram for $v^*(s_k)$

For a finite MDP the following expression results:

$$v^*(s_k) = \max_{a_k} \mathcal{R}_s^u + \gamma \sum_{s_{k+1} \in \mathcal{S}} p_{ss'}^a v_{\pi^*}(s_{k+1}). \tag{1.28}$$

Bellman Optimality Equation (3)

Likewise, the Bellman optimality equation is applicable to the action value:

$$q^*(s_k, a_k) = \mathbb{E}\left[R_{k+1} + \gamma \max_{a_{k+1}} q^*(S_{k+1}, A_{k+1}) | s_k = s_k, A_k = a_k\right]. \tag{1.29}$$

And, in the finite MDP case:

$$q^*(s_k, a_k) = \mathcal{R}_s^a + \gamma \sum_{s_{k+1} \in \mathcal{S}} p_{ss'}^a \max_{a_{k+1}} q^*(s_{k+1}, a_{k+1}). \tag{1.30}$$

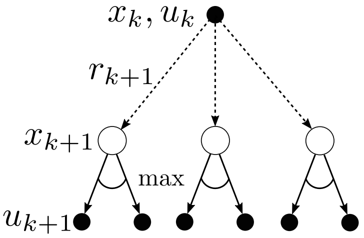


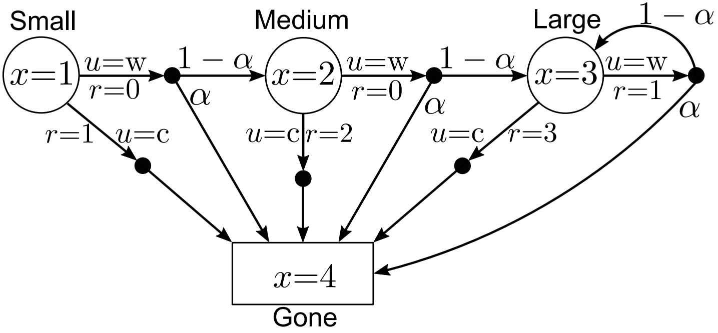
Fig. 1.14: Backup diagram for $q^*(s_k, a_k)$

Solving the Bellman Optimality Equation

- In finite MDPs with n states, (1.28) delivers an algebraic equation system with n unknowns and n state-value equations.
- Likewise, (1.30) delivers an algebraic equation system with up to $n \cdot m$ unknowns and $n \cdot m$ action-value equations ($m =$number of actions).
- Due to $\max$ operator the equation set is generally nonlinear. Direct, closed form solution rarely available.
- Hence, often approximate / iterative solutions are required (upcoming lectures).
- If environment is exactly known, solving for v^* or q^* directly delivers optimal policy.
  - If $v(s)$ is known, a one-step-ahead search is required to get $q(s)$.
  - If $q(s,u)$ is known, directly choose q^*.
- Even though above decisions are very short sighted (one-step-ahead search for v or direct choice of q), by Bellman's principle of optimality one receives the long-term maximum of the expected reward.

Optimal Policy for Forest Tree MDP

Remember the forest tree MDP example:



- Two actions possible in each state:
  - Wait $a = w$: let the tree grow.
  - Cut $a = c$: gather the wood.
- Lets first calculate $v^*(s)$ and then $q^*(s, a)$.

Optimal Policy for Forest Tree MDP: State Value (1)

Start with $v(s=4)$ ('gone') and then continue going backwards:

$$
\begin{aligned}
v^*(s=4) &= 0\,, \\
v^*(s=3) &= \max \begin{cases} 1 + \gamma\left[(1-\alpha)v^*(s=3) + \alpha v^*(s=4)\right]\,, \\ 3 + \gamma v^*(s=4)\,, \end{cases} \\
&= \max \begin{cases} 1 + \gamma\left[(1-\alpha)v^*(s=3)\right]\,, \\ 3\,, \end{cases} \\
v^*(s=2) &= \max \begin{cases} 0 + \gamma\left[(1-\alpha)v^*(s=3) + \alpha v^*(s=4)\right]\,, \\ 2 + \gamma v^*(s=4)\,, \end{cases} \\
&= \max \begin{cases} \gamma\left[(1-\alpha)v^*(s=3)\right]\,, \\ 2\,, \end{cases} \\
v^*(s=1) &= \max \begin{cases} \gamma\left[(1-\alpha)v^*(s=2)\right]\,, \\ 1\,. \end{cases}
\end{aligned}
$$

Optimal Policy for Forest Tree MDP: State Value (2)

- Due to presence of terminal state ($s = 4$) only three unknowns
- Solution: numerical optimization approach (e.g., simplex method, gradient descent,...)

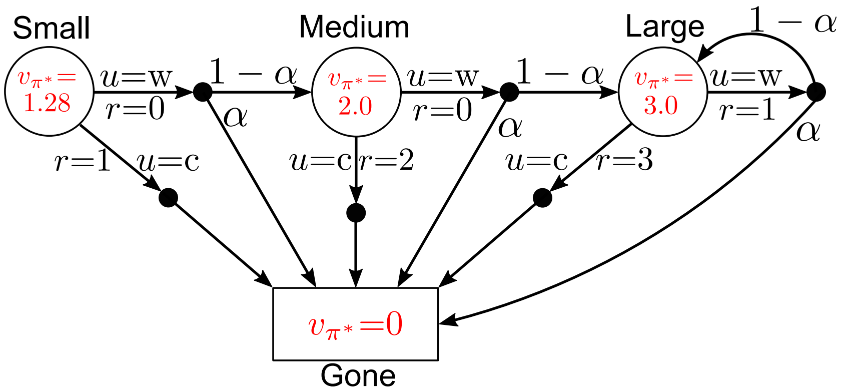


Fig. 1.15: State values under optimal policy ($\gamma = 0.8$, $\alpha = 0.2$)

Optimal Policy for Forest Tree MDP: State Value (3)

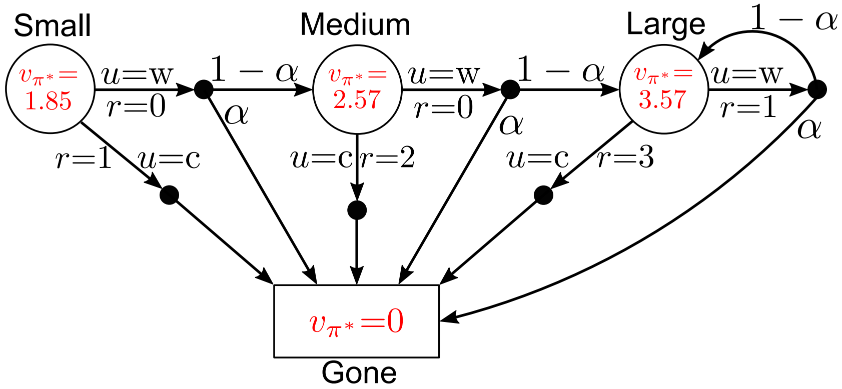


Fig. 1.16: State values under optimal policy ($\gamma = 0.9$, $\alpha = 0.2$)

Optimal Policy for Forest Tree MDP: Action Value (1)

Use $a_{k+1} = a'$ to set up equation system:

$$\begin{aligned}
q^*(s=1, a=c) &= 1\,,\\
q^*(s=1, a=w) &= \gamma(1-\alpha)\max_{a'} q^*(s=2, a')\,,\\
q^*(s=2, a=c) &= 2\,,\\
q^*(s=2, a=w) &= \gamma(1-\alpha)\max_{a'} q^*(s=3, a')\,,\\
q^*(s=3, a=c) &= 3\,,\\
q^*(s=3, a=w) &= 1 + \gamma(1-\alpha)\max_{a'} q^*(s=3, a')\,.
\end{aligned}$$

- There are six action-state pairs in total.
- Three of them can be directly determined.
- Three unknowns and three equations remain.

Optimal Policy for Forest Tree MDP: Action Value (2)

Rearrange $\max$ expressions for unknown action values:

$$q^*(s=1,a=w) = \gamma(1-\alpha)\max\begin{cases}\gamma(1-\alpha)\max\begin{cases}1+\gamma(1-\alpha)q^*(3,w)\\3\end{cases}\\2\end{cases}$$

$$q^*(s=2,a=w) = \gamma(1-\alpha)\max\begin{cases}1+\gamma(1-\alpha)q^*(3,w)\\3\end{cases}$$

$$q^*(s=3,a=w) = 1+\gamma(1-\alpha)\max\begin{cases}q^*(3,w)\\3\end{cases}\quad.$$

Again, retrieve unknown optimal action values by numerical optimization solvers.

Optimal Policy for Forest Tree MDP: Action Value (3)

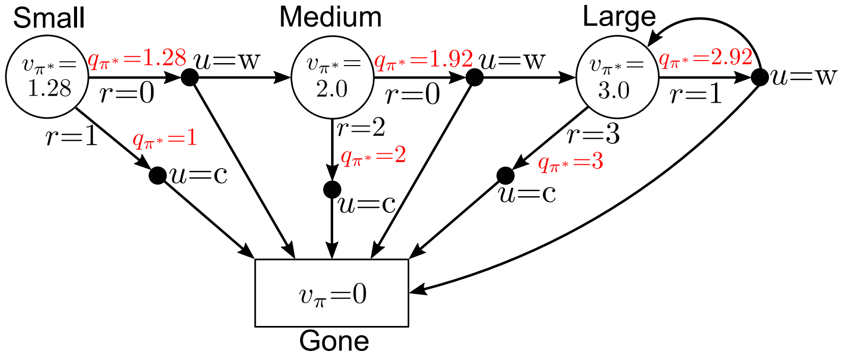


Fig. 1.17: Action values under optimal policy ($\gamma = 0.8$, $\alpha = 0.2$)

Optimal Policy for Forest Tree MDP: Action Value (4)

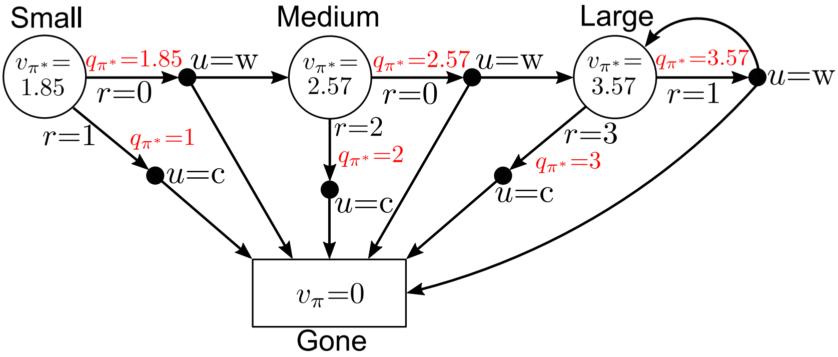


Fig. 1.18: Action values under optimal policy ($\gamma = 0.9$, $\alpha = 0.2$)

Direct Numerical State and Action-Value Calculation

- Possible only for small action and state-space MDPs
  - 'Solving' Backgammon with $\approx 10^{20}$ states?
- Another issue: initial values?
- And yet another issue: nonlinear system with multiple unknowns
  - Local solvers = local solutions
  - Initial-point-dependent
- And finally: total environment knowledge required

Framing the reinforcement learning problem

Facing the above issues, RL addresses mainly two topics:

- Approximate solutions of complex decision problems.
- Learning of such approximations based on environment interactions.

Summary: What You've Learned Today

- Differentiate finite Markov process models with or w/o rewards and actions.
- Interpret such stochastic processes as simplified abstractions of real-world problems.
- Understand the importance of value functions to describe the agent's performance.
- Formulate value-function equation systems by the Bellman principle.
- Recognize optimal policies.
- Setting up nonlinear equation systems for retrieving optimal policies by the Bellman principle.
- Solving for different value functions in MRP/MDP contexts by numerical optimization.