Our goal today

Dynamic Approximate Programming
  Introduction

Markov decision problems

Approximate dynamic programming
Literature (incomplete, but growing):

  [http://www.deeplearningbook.org](http://www.deeplearningbook.org)


- CRAN Task View: Machine Learning, available at [https://cran.r-project.org/web/views/MachineLearning.html](https://cran.r-project.org/web/views/MachineLearning.html)


From now on, we study the field of dynamic approximate programming (ADP) following Powell (2011). As we already learned, there are many dialects in this field and we treat them here. This includes reinforcement learning, and a classic reference is Sutton & Barto. For further references consider Powell (2011).

Examples are: moving a robot, investing in stocks, playing chess or go.

The system contains four main elements: a policy, a reward function, a value function and (optional) a model of the environment.

---


An Example

Let us start with a simple example.

It is our goal to find the shortest path from Start to End.
By $\mathcal{I}$ we denote the set of intersections $(S, 1, \ldots, E)$, if we are at intersection $i$ we can go to $j \in \mathcal{I}_i^+$ at cost $c_{ij}$, we start at $S$ and end in $E$. Denote

$$v_i := \text{cost from } i \text{ to } E$$

and we could iterate

$$v_i \leftarrow \min \left\{ v_i, \min_{j \in \mathcal{I}_i} (c_{ij} + v_j) \right\}, \quad v_i \in \mathcal{I}$$

and stop if the iteration does not change.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>S</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>10</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>18</td>
<td>10</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>18</td>
<td>10</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

What is an efficient algorithm for solving this problem?
This is a **shortest-path problem**. Let us introduce some notation for this. At time $t$, we start from a state $S_t$ and can choose **action** $a_t$ which leads to the transition to state $S_{t+1}$ given by the **transition function** $S$, s.t.

$$S_{t+1} = S(S_t, a_t)$$

Additionally there is a **reward**, denoted by $C_t(S_t, a_t)$ and we define the value of being in state $S_t$ by

$$V_t(S_t) = \max_{a_t} \left\{ C_t(S_t, a_t) + V_{t+1}(S_{t+1}) \right\}, \quad S_t \in \mathcal{S}_t,$$

$cS_t$ denoting the possible states at time $t$.

Let us visit some further examples.
Consider a gambler who plays $T$ rounds, on an i.i.d. $(W_t)_{t=1,...,T}$ game with probability $p = \mathbb{P}(W_t = 1) > 1 - p$ of winning. We want to maximize $\mathbb{E}[\log(S_T)]$. It can be shown that it is optimal to proceed backwards in time using conditional expectations (this is dynamic programming)!

Here, $a_t$ is the amount he bets at $t$ and we require $a_t \leq S_{t-1}$. Then,

$$S_t = S_{t-1} + a_t W_t - a_t (1 - W_t).$$

The value at time $t$, given his stock is in state $S_t$ is

$$V_t(S_t) = \max_{0 \leq a_{t+1} \leq S_t} \mathbb{E}[V_{t+1}(S_{t+1}) | S_t].$$
Now we proceed backwards. Clearly,

\[ V_T(s) = \log s \]

\[ V_{T-1}(s) = \max_{0 \leq a \leq s} \mathbb{E}[V_T(s + aW_T - a(1 - W_T)) | S_{T-1} = s] \]

\[ = \max_{0 \leq a \leq s} \left( p \log(s + a) + (1 - p) \log(s - a) \right). \]

The maximum is attained for \( a^* = (2p - 1)s \) and \( V_{T-1}(s) = \log(s) + K \), with constant \( K = p \log(2p) + (1 - p) \log(2(1 - p)) \). Backward in time we obtain

\[ V_t(s) = \log S_t + K_t, \]

with an explicit constant \( K_t \). Our **optimal policy** is

\[ a_t = (2p - 1)S_{t-1}. \]
The bandit problem

- When the distribution of the game is not known, one has to acquire information, and the classical example is the bandit problem. Consider a gambler who can choose between $K$ machines.
- The probability of winning might be different and are unknown to us.
- A trade-off arises between playing only the optimal machine or trying other machines with (estimated) lower probability for minimizing the variance which is one-to-one to learning better their true probability.
We give a short introduction into the field\textsuperscript{3}. Assume that the state space \( \mathcal{S} \) if finite.

We have a set \( \mathcal{A}_t(s) \) of possible actions at time \( t \) when the system is in state \( s \). An action at \( t \) is a measurable mapping \( a_t \) such that \( a_t(s) \in \mathcal{A}_t(s) \) for all \( s \in \mathcal{S} \).

A policy is a collection of actions \( \pi = (a_0, \ldots, a_{T−1}) \). We assume that the set of policies is non-empty.

The dynamics of the model is specified via the (conditional) transition matrix

\[
(p_t(s_{t+1}|s_t, a_t))_{s_{t+1}, s_t \in \mathcal{S}}
\]

specifying \( \mathbb{P}(S_{t+1} = s_{t+1}|S_t = s_t, a_t) = p_t(s_{t+1}|s_t, a_t) \).

Hence, the dynamics and with it the probability for evaluation depends on \( \pi \). We denote

\[
\mathbb{P}^{\pi}_{t,s}(\cdot) := \mathbb{P}^{\pi}(\cdot|S_t = s)
\]

and by \( \mathbb{E}^{\pi}_{t,s} \) the associated expectation.

---

Our aim is to **maximize** the contribution given by the functions $C_t(s, a)$ where $C_T(s, a) = C_T(s)$ does not depend on $a$. We additionally assume that the contribution is sufficiently integrable.

Our goal is to aim at

$$\sup_{\pi} \mathbb{E}\pi \left[ \sum_{t=1}^{T} C_t(S_t, a_t) \right].$$

For example, we could consider $C_t(s, a) = \gamma^t C(s, a)$ with possible discounting factor $\gamma > 0$. 
The Bellman Equation

- The key to dynamic programming is that in our set-up, allowing the policy to depend on the full history does not improve the maximal expected reward, see Theorem 2.2.3. in Bäuerle&Rieder (2011).

- We define the **value function** by

\[ V_t(s) = \sup_{\pi} \mathbb{E}_{t,s}^{\pi} \left[ \sum_{s=t}^{T} C_t(S_t, a_t) \right] . \]

**Remark**

*In general \( V_t \) need not be measurable which causes a number of delicate problems, see D. P. Bertsekas und S. Shreve (2004). **Stochastic optimal control: the discrete-time case.** for a detailed treatment. The reason can be traced back to the fact that a projection of a Borel set need not be Borel (which leads to the fruitful notion of analytic sets, however).*
Define
\[ C_t^*(s) := \sup_{a_t \in \mathcal{A}_t} \left( C_t(s, a_t) + \mathbb{E}\left[ V_{t+1}(S_{t+1})|S_t = s, a_t \right] \right) \] (1)

Recall, that \( S_{t+1} \) also depends on \( a_t = a_t(s) \) (which we suppress in the notation).

The optimal policy can be computed backward by reward iteration. Let \( a_t^* \) be a maximizing policy, that is \( a_t^* \) achieves \( C_t^* \) in Equation (1).

One can now show that the **Bellman equation** holds, i.e.
\[ V_t(s) = C_t^*(s) \quad t = 0, \ldots, T. \]

Under an additional (mild) structural assumption, one may verify that there always exist optimal policies \( \pi^* \) which can be obtained by maximizing the value function in each period (Theorem 2.3.8. in Bäuerle Rieder).
Algorithm

Step 0  Initialize by the terminal condition $V_T(S_T)$ and set $t = T - 1$

Step 1  Compute

$$V_t(s) = \sup_{a_t \in \mathcal{A}} \left( C_t(s, a_t) + \mathbb{E} \left[ V_{t+1}(S_{t+1}) | S_t = s, a_t \right] \right)$$

for all $s \in \mathcal{S}$

Step 2  Decrement $t$ and repeat Step 1 until $t = 0$
For this case several algorithms exist, to name **value iteration** and **policy iteration** which will not be discussed here, see Powell Section 3.3. ff.

For more mathematical details (and there are many!) we refer to Powell, Bäuerle&Rieder and the excellent source Bertsekas&Shreve.
While we introduce a nice theory beforehand, the core equation

$$\sup_{\pi} \mathbb{E}^\pi \left[ \sum_{t=0}^{T} C_t(S_t, a_t) \right]$$

might be intractable even for very small problems.

ADP now offers a powerful set of strategies to solve these problems approximately.

We have the problem of curse of dimensionality in state space, outcome space and action space.