Approximate Dynamic Programming for High-Dimensional Problems

2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning
April, 2007

Warren Powell
CASTLE Laboratory
Princeton University
http://www.castlelab.princeton.edu

© 2007 Warren B. Powell, Princeton University
The fractional jet ownership industry
Planning for a Risky World

**Weather**

- Robust design of emergency response networks.
- Design of financial instruments to hedge against weather emergencies to protect individuals, companies, and municipalities.
- Design of sensor networks and communication systems to manage responses to major weather events.

**Disease**

- Models of disease propagation for response planning.
- Management of medical personnel, equipment, and vaccines to respond to a disease outbreak.
- Robust design of supply chains to mitigate the disruption of transportation systems.
Blood management

![Blood Type Diagram]

© 2007 Warren B. Powell
Managing financial portfolios

- Money can be invested and then reinvested....
Energy Management

Applications

- Jet fuel hedging – Designing strategies to hedge against fluctuations in jet fuel (and other commodities).
- Valuing energy contracts
- Planning the use of future technologies
- R&D portfolio management

Research in ADP

- Convergence proofs
- Rate of convergence research
- Design and evaluation of approximation strategies
- Design of advanced approximation strategies
Challenges

- **Real-time control**
  - Scheduling aircraft, pilots, generators, tankers
  - Pricing stocks, options

- **Near-term tactical planning**
  - Can I accept a customer request?
  - Should I lease equipment?
  - Do I have to purchase extra energy?

- **Strategic planning**
  - What is the right equipment mix?
  - What is the value of this contract?
  - What is the value of more reliable aircraft?
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
## Languages

- The languages of “optimization over time”

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Engineering</th>
<th>OR/AI/Probability</th>
<th>OR/Math programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal control</td>
<td>Control</td>
<td>Action</td>
<td>Decision</td>
</tr>
<tr>
<td>Decision (English)</td>
<td>u</td>
<td>a</td>
<td>x</td>
</tr>
<tr>
<td>&quot;Value function&quot; (English)</td>
<td>Cost-to-go</td>
<td>Value function</td>
<td>Recourse function</td>
</tr>
<tr>
<td>&quot;Value function&quot; (Math)</td>
<td>J</td>
<td>V</td>
<td>Q</td>
</tr>
<tr>
<td>State variable</td>
<td>x</td>
<td>S</td>
<td>Huh? (oh, &quot;tenders&quot;)</td>
</tr>
<tr>
<td>Optimality equations</td>
<td>Hamilton-Jacobi</td>
<td>Bellman</td>
<td>Huh?</td>
</tr>
</tbody>
</table>
Languages

“Approximate dynamic programming” has been discovered independently by different communities under different names:

» Neuro-dynamic programming
» Reinforcement learning
» Forward dynamic programming
» Adaptive dynamic programming
» Heuristic dynamic programming
» Iterative dynamic programming
Languages

- How to land a plane:

  » Control: angle, velocity, acceleration, pitch, yaw…
  » Noise: wind, measurement

\[ V_t(x_t) = \max_u \left( C(x_t, u_t) + EV_{t+1}(x_{t+1}) \right) \]
Where to send a plane:

- Control: Where to send the plane to accomplish a goal.
- Noise: demands on the system, equipment failures.

\[ V_t(S_t) = \max_a \left( C(S_t, a_t) + EV_{t+1}(S_{t+1}) \right) \]
Languages

How to manage a fleet of aircraft:

» Control: Which plane to assign to each customer.

» Noise: demands on the system, equipment failures.

\[ V_t(S_t) = \max_x \left( C(S_t, x_t) + EV_{t+1}(S_{t+1}) \right) \]
A progression of models

- Major problem classes

<table>
<thead>
<tr>
<th></th>
<th>Simple attributes</th>
<th>Complex attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single entity</td>
<td>Textbook Markov decision process</td>
<td>Classical AI applications</td>
</tr>
<tr>
<td>Multiple entities</td>
<td>Classical OR applications</td>
<td>Opportunity for combining AI/OR</td>
</tr>
</tbody>
</table>
Sample applications

- Single entity problems
  - Playing a board game
  - Routing a truck around the country
  - Planning a set of courses through college

- Storage problems (single resource class)
  - Maintaining product inventories
  - Purchasing commodity futures (oil, orange juice, ...)

- Managing multiple resource classes
  - Blood inventories
  - Fleet management (with different equipment types)

- Managing multiple, discrete resources
  - Locomotives, jets, people
Single entity problems
Single entity problems
Single entity problems
Single entity problems
Single entity problems
Asset acquisition problems

Storage problems

\[ R_0 \]

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6 \]

\[ \hat{P}_1, \hat{D}_1 \hat{P}_2, \hat{D}_2 \hat{P}_3, \hat{D}_3 \hat{P}_4, \hat{D}_4 \hat{P}_5, \hat{D}_5 \hat{P}_6, \hat{D}_6 \hat{P}_7, \hat{D}_7 \]

\[ R_{t+1} = \left[ R_t + x_t - \hat{D}_{t+1} \right]^+ \]

\[ P_{t+1} = P_t + \hat{P}_t \]
Multiple inventory types

- Managing blood inventories
### Multiple inventory types

- Managing blood inventories over time

<table>
<thead>
<tr>
<th>Week 0</th>
<th>Week 1</th>
<th>Week 2</th>
</tr>
</thead>
</table>

![Diagram showing blood inventory types across weeks](image-url)
Multiple discrete assets
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
A resource allocation model

- Modeling resources:
  - The state of a single resource:
    - \( a = \) The attributes of a single resource
    - \( a \in \mathcal{A} \) The attribute space
  - The state of multiple resources:
    - \( R_{ta} = \) The number of resources with attribute \( a \)
    - \( R_t = \left( R_{ta} \right)_{a \in \mathcal{A}} \) The resource state vector
  - The information process:
    - \( \hat{R}_{ta} = \) The change in the number of resources with attribute \( a \).
A resource allocation model

Modeling demands:

- The attributes of a single demand:
  \[ b = \text{The attributes of a demand to be served.} \]
  \[ b \in \mathcal{B} \quad \text{The attribute space} \]

- The demand state vector:
  \[ D_{tb} = \text{The number of demands with attribute } b \]
  \[ D_t = \left( D_{tb} \right)_{b \in \mathcal{B}} \quad \text{The demand state vector} \]

- The information process:
  \[ \hat{D}_{tb} = \text{The change in the number of demands with attribute } b. \]
A resource allocation model

- The system:
  - The state vector
    
    \[ S_t = \left( R_t, D_t \right) \]
  - The information process:
    
    \[ W_t = \text{Exogenous changes to resources and demands} \]
    
    \[ = \left( \hat{R}_t, \hat{D}_t \right) \]
A resource allocation model

- The three states of our system
  - The state of a single resource/entity
    \[
    a_t = \begin{bmatrix}
    a_{t1} \\
    a_{t2} \\
    a_{t3}
    \end{bmatrix}
    \]
  - The resource state vector
    \[
    R_t = \begin{bmatrix}
    R_{ta_1} \\
    R_{ta_2} \\
    R_{ta_3}
    \end{bmatrix}
    \]
  - The system state vector
    \[S_t = (R_t, D_t)\]
A resource allocation model
A resource allocation model

\[ t \quad t+1 \quad t+2 \]
A resource allocation model

Optimizing at a point in time

Optimizing over time
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Laying the foundation

Dynamic programming review:

» Let:

\[ S_t = "\text{State}" \text{ of our } "\text{system}" \text{ at time } t. \]
\[ x_t = "\text{Action}" \text{ that we take to change the system.} \]
\[ C(S_t, x_t) = \text{Contribution earned when we take action } x \text{ from state } S_t. \]

» We model system dynamics using:

\[ p(S_{t+1} \mid S_t, x_t) = \text{Probability that action } x_t \text{ takes us from state } S_t \text{ to state } S_{t+1} \]
Laying the foundation

Bellman’s equation:

» Standard form:

$$V_t(S_t) = \max_x \left( C_t(S_t, x_t) + \sum_{s'} p(s' \mid S_t, x_t) V_{t+1}(S_{t+1} = s') \right)$$

» Expectation form:

$$V_t(S_t) = \max_x \left( C_t(S_t, x_t) + E \{ V_{t+1}(S_{t+1}(S_t, x_t)) \mid S_t \} \right)$$
Do not use weather report

Use weather report

Forecast sunny .6
- Schedule game - $200
- Cancel game - $200

Forecast cloudy .3
- Schedule game $2300
- Cancel game - $200

Forecast rain .1
- Schedule game $3500
- Cancel game - $200

Do not use weather report
- Schedule game $2400
- Cancel game - $200
Do not use weather report

Forecast sunny 0.6

Schedule game $2400
Cancel game -$200

Forecast cloudy 0.3

$2300

Forecast rain 0.1

-$200

$3500
Bellman’s equation

We just solved Bellman’s equation:

\[ V_t(S_t) = \max_{x \in \mathcal{X}} C_t(S_t, x_t) + E \{ V_{t+1}(S_{t+1}) \mid S_t \} \]

» We found the value of being in each state by stepping backward through the tree.
Bellman’s equation

The challenge of dynamic programming:

\[ V_t(S_t) = \max_{x \in \mathcal{X}} \left( C_t(S_t, x_t) + E \{ V_{t+1}(S_{t+1}) | S_t \} \right) \]
The curses of dimensionality

What happens if we apply this idea to our blood problem?

» State variable is:
  • The supply of each type of blood, along with its age
    – 8 blood types
    – 6 ages
    – = 48 “blood types”
  • The demand for each type of blood
    – 8 blood types

» Decision variable is how much of 48 blood types to supply to 8 demand types.
  • 216- dimensional decision vector

» Random information
  • Blood donations by week (8 types)
  • New demands for blood (8 types)
The curses of dimensionality

The challenge of dynamic programming:

\[ V_t(S_t) = \max_{x \in \mathcal{X}} \left( C_t(S_t, x_t) + E \{ V_{t+1}(S_{t+1}) \mid S_t \} \right) \]

Three curses

Problem: Curse of dimensionality

State space
Outcome space
Action space (feasible region)
The curses of dimensionality

- The computational challenge:

\[ V_t(S_t) = \max_{x \in \mathcal{X}} \left( C_t(S_t, x_t) + E \{V_{t+1}(S_{t+1}) | S_t \} \right) \]

How do we find \( V_{t+1}(S_{t+1}) \)?

How do we compute the expectation?

How do we find the optimal solution?
\[ V_t(S_t) = \max_{x \in \mathcal{X}} \left( C_t(S_t, x_t) + E \left\{ V_{t+1}(S_{t+1}) \mid S_t \right\} \right) \]
Pre- and post-decision states

■ New concept:

» The “pre-decision” state variable:

• \( S_t = \) The information required to make a decision \( x_t \)

• Same as a “decision node” in a decision tree.

» The “post-decision” state variable:

• \( S^x_t = \) The state of what we know immediately after we make a decision.

• Same as an “outcome node” in a decision tree.
Pre- and post-decision states

- Pre-decision, state-action, and post-decision

Pre-decision state | State | Action | Post-decision state
---|---|---|---
3^9 states | 3^9 × 9 state-action pairs | 3^9 states

© 2007 Warren B. Powell
A single, complex entity

- Pre- and post-decision attributes for our nomadic truck driver:

<table>
<thead>
<tr>
<th></th>
<th>Pre-decision</th>
<th>Decision</th>
<th>Post-decision</th>
<th>Pre-decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>(Dallas)</td>
<td>(Chicago)</td>
<td>(Chicago)</td>
<td>(Chicago)</td>
</tr>
<tr>
<td>ETA</td>
<td>41.2</td>
<td>-</td>
<td>54.7</td>
<td>56.2</td>
</tr>
<tr>
<td>Equip</td>
<td>Good</td>
<td>-</td>
<td>Good</td>
<td>Repair</td>
</tr>
</tbody>
</table>

$t = 40$; Pre- and post-decision attributes for our nomadic truck driver:

$t = 50$
Pre- and post-decision states

- Pre-decision: resources and demands

\[ S_t = (R_t, D_t) \]
Pre- and post-decision states

\[ S'_{t,x} = S'^{M,x}_{t,x} (S_t, x_t) \]
Pre- and post-decision states

\[ S_t^x \]

\[ S_{t+1} = S^{M,W}(S_t^x, W_{t+1}) \]

\[ W_{t+1} = (\hat{R}_{t+1}, \hat{D}_{t+1}) \]
Pre- and post-decision states

$S_{t+1}$
System dynamics

- It is traditional to assume you are given the one-step transition matrix:

\[ p(S_{t+1} \mid S_t, x_t) = \text{Probability that action } x_t \text{ takes us from state } S_t \text{ to state } S_{t+1} \]

- Computing the transition matrix is impossible for the vast majority of problems.

- We are going to assume that we are given a transition function:

\[ S_{t+1} = S^M (S_t, x_t, W_{t+1}) \]

- This is at the heart of any simulation model.
- Often rule-based. Very easy to compute, even for large-scale problems.
The transition function

■ Working with pre- and post-decision states
  » The “usual” transition function:

\[ S_{t+1} = S^M \left( S_t, x_t, W_{t+1} \right) \]
  From \( S_t \) to \( S_{t+1} \).

» The transition function broken into two steps:

\[ S_t^x = S^{M,x} \left( S_t, x_t \right) \]
  The pure effect of a decision

\[ S_{t+1} = S^{M,W} \left( S_t^x, W_{t+1} \right) \]
  The effect of the exogenous information
The transition function

- Actually, we have three transition functions:
  - The attribute transition function:
    \[ a_t^x = a^{M,x} (a_t, x_t) \]  
    The pure effect of a decision
    \[ a_{t+1} = a^{M,W} (a_t^x, W_{t+1}) \]  
    The effect of the exogenous information
  - The resource transition function
    \[ R_t^x = R^{M,x} (R_t, x_t) \]  
    The pure effect of a decision
    \[ R_{t+1} = R^{M,W} (R_t^x, W_{t+1}) \]  
    The effect of the exogenous information
  - The general transition function:
    \[ S_t^x = S^{M,x} (S_t, x_t) \]  
    The pure effect of a decision
    \[ S_{t+1} = S^{M,W} (S_t^x, W_{t+1}) \]  
    The effect of the exogenous information
Bellman’s equations with the post-decision state

Bellman’s equations broken into stages:

» Optimization problem (making the decision):

\[ V_t(S_t) = \max_x \left( C_t(S_t, x_t) + V_t^x \left( S_{t+1}^{M,x}(S_t, x_t) \right) \right) \]

• Note: this problem is deterministic!

» Simulation problem (the effect of exogenous information):

\[ V_t^x(S_t^x) = E \left\{ V_{t+1}^{M,W}(S_{t+1}^x, W_{t+1}) \middle| S_t^x \right\} \]
Bellman’s equations with the post-decision state

**Challenges**

» For most practical problems, we are not going to be able to compute $V_t^x(S_t^x)$.

$$V_t(S_t) = \max_x \left( C_t(S_t, x_t) + V_t^x(S_t^x) \right)$$

» Concept: replace it with an approximation $\bar{V}_t(S_t^x)$ and solve

$$V_t(S_t) = \max_x \left( C_t(S_t, x_t) + \bar{V}_t(S_t^x) \right)$$

» So now we face:
  - What should the approximation look like?
  - How do we estimate it?
Approximating the value function

For “resource allocation” problems, we have been using:

» Linear (in the resource state):

\[ \overline{V}_t (R_t^x) = \sum_{a \in A} \overline{v}_{ta} \cdot R_{ta}^x \]

Best when assets are complex, which means that \( R_{ta} \) is small (typically 0 or 1).

» Piecewise linear, separable:

\[ \overline{V}_t (R_t^x) = \sum_{a \in A} \overline{V}_{ta} (R_{ta}^x) \]

Best when assets are simple, which means that \( R_{ta} \) may be larger.
Our general algorithm

Step 1: Start with a post-decision state $S_{t-1}^{x,n}$

Step 2: Obtain Monte Carlo sample of $W_t(\omega^n)$ and compute the next pre-decision state:

$$S_t^n = S^{M,W}_t \left( S_{t-1}^{x,n}, W_t(\omega^n) \right)$$

Step 3: Solve the deterministic optimization using an approximate value function:

$$\hat{v}_t^n = \max_x \left( C_t(S_t^n, x_t) + \bar{V}_t^{n-1}(S_t^{M,x}(S_t^n, x_t)) \right)$$

to obtain $x_t^n$.

Step 4: Update the value function approximation

$$\bar{V}_t^n(S_{t-1}^{x,n}) = (1 - \alpha_{n-1}) \bar{V}_{t-1}^{n-1}(S_{t-1}^{x,n}) + \alpha_{n-1} \hat{v}_t^n$$

Step 5: Find the next post-decision state:

$$S_t^{x,n} = S^{M,x}_t (S_t^n, x_t^n)$$
Competing updating methods

Comparison to other methods:

» Classical MDP (value iteration)

\[
V^n(S) = \max_x \left( C(S, x) + EV^{n-1}(S_{t+1}) \right)
\]

» Classical ADP (pre-decision state):

\[
\hat{v}_t^n = \max_x \left( C_t(S^n_t, x_t) + \sum_{s'} p(s'| S^n_t, x_t) \bar{V}_{t+1}^{n-1}(s') \right)
\]

\[
\bar{V}_t^n(S^n_t) = (1 - \alpha_{n-1}) \bar{V}_{t+1}^{n-1}(S^n_t) + \alpha_{n-1} \hat{v}_t^n \quad \hat{v}_t \text{ updates } \bar{V}_t(S_t)
\]

» Our method (update \( \bar{V}_{t,x}^{n-1} \) around post-decision state):

\[
\hat{v}_t^n = \max_x \left( C_t(S^n_t, x_t) + \bar{V}_{t}^{n-1}(S^{M_x}(S^n_t, x_t)) \right)
\]

\[
\bar{V}_{t-1}^{n}(S^{x,n}_{t-1}) = (1 - \alpha_{n-1}) \bar{V}_{t-1}^{n-1}(S^{x,n}_{t-1}) + \alpha_{n-1} \hat{v}_t^n \quad \hat{v}_t \text{ updates } \bar{V}_{t-1}(S^{x}_{t-1})
\]
APPROXIMATE DYNAMIC PROGRAMMING
Solving the curses of dimensionality

Warren B. Powell
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Nomadic trucker illustration

- The previous post-decision state: trucker in Texas

\[ S_{t-1}^x = \left( \begin{array}{c} \text{Location} \\ \text{Time avail} \end{array} \right) = \left( \begin{array}{c} TX \\ t \end{array} \right) \]
Nomadic trucker illustration

- Pre-decision state: we see the demands

\[ S_t = \left( \frac{TX}{t}, \hat{D}_t \right) \]
Nomadic trucker illustration

- We use initial value function approximations...

\[ S_t = \left( \frac{TX}{t}, \hat{D}_t \right) \]
Nomadic trucker illustration

... and make our first choice: $x^1$

$$S^x_t = \begin{pmatrix} NY \\ t+1 \end{pmatrix}$$
Nomadic trucker illustration

- Update the value of being in Texas.

\[ S_t^x = \left( \begin{array}{c} NY \\ t+1 \end{array} \right) \]

© 2007 Warren B. Powell
Now move to the next state, sample new demands and make a new decision

\[ S_{t+1} = \left( \begin{array}{c} NY \\ t + 1 \end{array} \right), \hat{D}_{t+1} \]
Nomadic trucker illustration

- Update value of being in NY

\[ S_{t+1}^x = \begin{pmatrix} CA \\ t + 2 \end{pmatrix} \]
Nomadic trucker illustration

- Move to California.

\[ S_{t+2} = \left( \begin{array}{c} CA \\ t + 2 \end{array} \right), \hat{D}_{t+2} \]
Nomadic trucker illustration

- Make decision to return to TX and update value of being in CA

\[ S_{t+2} = \left( \begin{array}{c} CA \\ t + 2 \end{array} \right, \hat{D}_{t+2} \]
Nomadic trucker illustration

- Back in TX, we repeat the process, observing a different set of demands.

\[
\begin{align*}
\bar{V}^0(CO) &= 0 \\
\bar{V}^0(MN) &= 0 \\
\bar{V}^0(CA) &= 800 \\
\bar{V}^0(NY) &= 500 \\
\bar{V}^1(TX) &= 450 \\
\end{align*}
\]

\[
S_{t+3} = \left( \begin{array}{c} TX \\ t + 3 \end{array}, \hat{D}_{t+3} \right)
\]
We get a different decision and a new estimate of the value of being in TX

\[
\hat{S}_{t+3} = \left( \begin{array}{c} TX \\ t + 3 \end{array} \right), \hat{D}_{t+3}
\]
Nomadic trucker illustration

- Updating the value function:

  Old value:
  \[ \bar{V}^1(TX) = $450 \]

  New estimate:
  \[ \hat{v}^2(TX) = $800 \]

  How do we merge old with new?
  \[ \bar{V}^2(TX) = (1 - \alpha)\bar{V}^1(TX) + (\alpha)\hat{v}^2(TX) \]
  \[ = (0.90)$450 + (0.10)$800 \]
  \[ = $485 \]
Nomadic trucker illustration

- An updated value of being in TX

\[ S_{t+3} = \left( \frac{TX}{t+3}, \hat{D}_{t+3} \right) \]
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
The states of our system

Now let’s take a look at what we just did:

\[ a_t = \text{Attribute of our nomadic trucker at time } t \text{ before the decision is made.} \]

\[ \hat{D}_t = \text{Vector of demands that are revealed at time } t. \]

\[ S_t = (a_t, \hat{D}_t) \quad \text{Pre-decision state variable.} \]

\[ a_t^x = \text{Attribute of our nomadic trucker at time } t \text{ after the decision is made.} \]

\[ S_t^x = a_t^x \quad \text{Post-decision state variable.} \]
A single, complex entity

- Pre- and post-decision attributes for our nomadic truck driver:

<table>
<thead>
<tr>
<th>Time</th>
<th>City</th>
<th>ETA</th>
<th>Equip</th>
<th>Decision</th>
<th>Post-decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 40$</td>
<td>Dallas</td>
<td>41.2</td>
<td>Good</td>
<td>-</td>
<td>Chicago 54.7</td>
</tr>
<tr>
<td>$t = 40$</td>
<td>Chicago</td>
<td>-</td>
<td>-</td>
<td>Good</td>
<td>...</td>
</tr>
<tr>
<td>$t = 50$</td>
<td>Chicago</td>
<td>-</td>
<td>-</td>
<td>Repair</td>
<td>-</td>
</tr>
</tbody>
</table>
Multiple, complex entities

Notation for multiple entities:

» The truck state vector:

\[ a = \text{The attributes of the truck} \]

\[ a \in \mathcal{A} \quad \text{The attribute space} \]

\[ R_{ta}^{\text{truck}} = \text{The number of trucks with attribute } a \]

\[ R_t^{\text{truck}} = \left( R_{ta}^{\text{truck}} \right)_{a \in \mathcal{A}} \quad \text{The truck state vector} \]

» The information process:

\[ \hat{R}_{ta}^{\text{truck}} = \text{The change in the number of trucks with attribute } a. \]
Multiple, complex entities

- Modeling the fleet management problem:
  - The load state vector:
    \[ b = \text{The attributes of a load to be moved.} \]
    \[ b \in \mathcal{B} \quad \text{The attribute space} \]
    \[ R_{tb}^{load} = \text{The number of tasks with attribute } b \]
    \[ R_t^{load} = \left( R_{tb}^{load} \right)_{b \in \mathcal{B}} \quad \text{The load state vector} \]
  - The information process:
    \[ \dot{R}_{tb}^{load} = \text{The change in the number of loads with} \]
    \[ \text{attribute } b. \]
Multiple, complex entities

- Modeling the fleet management problem:
  - The resource state vector (a.k.a. “physical state”)
    \[ R_t = \left( R_{t}^{\text{truck}}, R_{t}^{\text{load}} \right) \]
  - The information process:
    \[ \hat{R}_t = \text{The number of new arrivals (of drivers and loads) during time interval } t. \]
    \[ = \left( \hat{R}_{t}^{\text{truck}}, \hat{R}_{t}^{\text{load}} \right) \]
    \[ = W_t \]
The states of our system

- The state of a single, simple entity:

$$a = \text{[Location]}$$

$$a \in A \quad |A| \approx 100 - 10,000$$
The states of our system

- The state of a single, complex entity:

\[ a = \begin{bmatrix} 
\text{Time} \\
\text{Location} \\
\text{Equipment type} \\
\text{Home base} \\
\text{Operator attributes} \\
\text{Time in service} \\
\text{Maintenance status} 
\end{bmatrix} \]

\[ a \in \mathcal{A} \quad |\mathcal{A}| \approx 10^{10} - 10^{100} \]

The curse of dimensionality!
The states of our system

- Multiple, complex entities

\[ R_t = \begin{bmatrix} R_{ta_1} \\ R_{ta_2} \\ R_{ta_3} \\ \vdots \\ R_{ta_n} \end{bmatrix} \]

The number of dimensions of our state variable is equal to the size of the state space for a single entity problem.

The curse of curses.
The states of our system

■ What is a state variable?

» A minimally dimensioned function of history that necessary and sufficient to compute the decision function, the transition function and the contribution function.
The states of our system

The three states of our system

» The state of a single resource/entity

\[
a_t = \begin{bmatrix}
a_{t1} \\
a_{t2} \\
a_{t3}
\end{bmatrix}
\]

» The state of all our resources

\[
R_t = \begin{bmatrix}
R_{ta_1} \\
R_{ta_2} \\
R_{ta_3}
\end{bmatrix}
\]

» The state of knowledge

\[
S_t = \left( R_t, \bar{\theta}_t \right) \quad \bar{\theta}_t = \text{Estimates of "other parameters"}
\]
The states of our system

- The state variable

\[ S_t = (R_t, \bar{\theta}_t) \]

- The resource (physical) state
- “Additional information”
The states of our system

- The state variable

\[ S_t = (R_t, \bar{\theta}_t) \]

The “state of knowledge”
Nomadic trucker illustration

What is missing from our state variable?

\[ S_{t+3} = \left( \frac{TX}{t+3}, \hat{D}_{t+3} \right) \]
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Blood management

Managing blood inventories
Blood management

Managing blood inventories over time

Week 1

$S_0 \xrightarrow{\hat{R}_1, \hat{D}_1} S_1 \xrightarrow{x_1} S_1^x \xrightarrow{\hat{R}_2, \hat{D}_2} S_2 \xrightarrow{x_2} S_2^x \xrightarrow{\hat{R}_3, \hat{D}_3} S_3 \xrightarrow{x_3} S_3^x$

$t=0$ $t=1$ $t=2$ $t=3$
\[ S_t = \begin{pmatrix} R_t, & \hat{\mathcal{D}}_t \end{pmatrix} \]

\[ R_t^{x} \]

- \( R_{t,(AB+,0)} \) ➔ \( AB+,0 \)
- \( R_{t,(AB+,1)} \) ➔ \( AB+,1 \)
- \( R_{t,(AB+,2)} \) ➔ \( AB+,2 \)
- \( R_{t,(AB-,0)} \) ➔ \( AB-,0 \)
- \( R_{t,(AB-,1)} \) ➔ \( AB-,1 \)
- \( R_{t,(AB-,2)} \) ➔ \( AB-,2 \)

\( \hat{\mathcal{D}}_{t,AB+} \)
\( \hat{\mathcal{D}}_{t,AB-} \)
\( \hat{\mathcal{D}}_{t,A+} \)
\( \hat{\mathcal{D}}_{t,A-} \)
\( \hat{\mathcal{D}}_{t,B+} \)
\( \hat{\mathcal{D}}_{t,B-} \)
\( \hat{\mathcal{D}}_{t,O+} \)
\( \hat{\mathcal{D}}_{t,O-} \)

Satisfy a demand ➔ Hold
Solve this as a linear program.
Dual variables give value additional unit of blood.
Updating the value function approximation

- Estimate the gradient at $R_t^n$

![Graph showing the updating of the value function approximation]
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$
Updating the value function approximation

- The updated function may not be concave:

\[
R_{t-1}^{x,n}
\]
Maintaining concavity

A concave function...

... has monotonically decreasing slopes. But updating the function with a stochastic gradient may violate this property.
Maintaining concavity
Maintaining concavity
Maintaining concavity

$$\tilde{v}_{ta}^n = (1 - \alpha)\tilde{v}_{ta}^{n-1} + \alpha\tilde{v}_{ta}^n$$
A projection algorithm (SPAR)

\[
\tilde{v}_{ta}^n = (1 - \alpha)\tilde{v}_{ta}^{n-1} + \alpha\hat{v}_{ta}^n
\]
A projection algorithm (SPAR)

\[ \tilde{v}_{ta}^n = (1 - \alpha)\tilde{v}_{ta}^{n-1} + \alpha \tilde{v}_{ta}^n \]
A projection algorithm (SPAR)

\[ \tilde{v}_{ta}^n = (1 - \alpha)\tilde{v}_{ta}^{n-1} + \alpha\hat{v}_{ta}^n \]
Blood management
Blood management
Blood management
Blood management
Blood management
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
**Multiattribute resources**

- Assets can have a number of attributes:

<table>
<thead>
<tr>
<th>Location</th>
<th>ETA</th>
<th>Equipment type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home shop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>ETA</th>
<th>Equipment type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due for maint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home shop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>ETA</th>
<th>Equipment type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/C type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuel level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home shop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crew</th>
<th>Eqpt1</th>
<th>Eqpt100</th>
</tr>
</thead>
</table>

\[
a = \begin{bmatrix}
\text{Location} \\
\text{Equipment type} \\
\text{ETA} \\
\text{Train priority} \\
\text{Pool} \\
\text{Due for maint} \\
\text{Home shop}
\end{bmatrix}
\]
Boxcars
Multiattribute resources

- The evolution of attributes:

\[ a = \begin{bmatrix}
\text{Time} \\
\text{Location} \\
\text{Boxcar type} \\
\text{Time to dest.}
\end{bmatrix} \begin{bmatrix}
\text{Time} \\
\text{Location} \\
\text{Boxcar type} \\
\text{Time to dest.}
\end{bmatrix} \begin{bmatrix}
\text{Time} \\
\text{Location} \\
\text{Boxcar type} \\
\text{Repair status}
\end{bmatrix} \begin{bmatrix}
\text{Time} \\
\text{Location} \\
\text{Boxcar type} \\
\text{Shipper pool}
\end{bmatrix} \]

\[ |A| \approx 4,000 \quad 40,000 \quad 1,680,000 \quad 5,040,000 \quad 50,400,000 \]
The states of our system

- Multiple, complex entities

\[ R_t = \begin{bmatrix} R_{ta_1} \\ R_{ta_2} \\ \vdots \\ R_{ta_n} \end{bmatrix} \]

The number of *dimensions* of our state variable is equal to the size of the *state space* for a single entity problem.

The curse of curses.
Aggregation for table lookup
Aggregation for table lookup

\[ v_{PA} \approx v_{NE} \]

\[ v_{NE} \]

NE region

PA
Aggregation for table lookup

- Different levels of aggregation

\[
a = \begin{bmatrix}
\text{Time} \\
\text{Location} \\
\text{Fleet} \\
\text{Domicile} \\
\text{Days from home}
\end{bmatrix}
\]

\[
|A| \approx 12,000,000 \quad 600,000 \quad 6,000 \quad 2,000
\]
Aggregation

- Aggregation:
  - Exact methods
    - We have to use the same level of aggregation throughout (in particular, the transition matrix and value function).

\[
V_t(S_t) = \max_x \left( C_t(S_t, x_t) + \sum_{s'} p(s'|S_t, x_t) V_{t+1}(s') \right)
\]
Aggregation

Approximate DP

» We only need to discretize the value function. We can capture the full state variable in the transition function:

- Decision function:

\[ x_t = \arg \max_x \left( C_t(S_t, x_t) + \overline{V}_t(S_t^x) \right) \]

- Transition functions

\[
\begin{align*}
S_t &= S^{M,W}(S_{t-1}^x, W_t(\omega)) \\
S_t^x &= S^{M,x}(S_t, x_t)
\end{align*}
\]
Aggregation for table lookup

- Updating the value of a driver:

\[
\bar{v}^n (\begin{bmatrix} Location \\ Fleet \\ Domicile \end{bmatrix}) = (1 - \alpha)\bar{v}^{n-1} (\begin{bmatrix} Location \\ Fleet \\ Domicile \end{bmatrix}) + \alpha \hat{\nu} (\begin{bmatrix} Location \\ Fleet \\ Domicile \\ DOThrs \\ DaysFromHome \end{bmatrix})
\]

\[
\$2050 = (1 - 0.10) \times \$2000 + (0.10) \times \$2500
\]

Value function
Approximation may have fewer attributes than driver.

Drivers may have very detailed attributes
Estimating value functions

Most disaggregate level

\[ \bar{v}^n \begin{bmatrix} \text{Location} \\ \text{Fleet} \\ \text{Domicile} \end{bmatrix} = (1 - \alpha)\bar{v}^{n-1} \begin{bmatrix} \text{Location} \\ \text{Fleet} \\ \text{Domicile} \end{bmatrix} + \alpha \hat{v} \begin{bmatrix} \text{Location} \\ \text{Fleet} \\ \text{Domicile} \\ \text{DOThrs} \\ \text{DaysFromHome} \end{bmatrix} \]
Aggregation for table lookup

- Estimating value functions
  » Middle level of aggregation

$$\overline{v}^n\left(\begin{bmatrix} Location \\ Fleet \end{bmatrix}\right) = (1-\alpha)\overline{v}^{n-1}\left(\begin{bmatrix} Location \\ Fleet \end{bmatrix}\right) + \alpha \hat{v}\left(\begin{bmatrix} Location \\ Fleet \\ Domicile \\ DOThrs \\ DaysFromHome \end{bmatrix}\right)$$
Estimating value functions

» Most aggregate level

\[
\bar{v}^n (\begin{bmatrix} Location \\ Fleet \\ Domicile \\ DOThrs \\ DaysFromHome \end{bmatrix}) = (1 - \alpha)\bar{v}^{n-1} (\begin{bmatrix} Location \end{bmatrix}) + \alpha \hat{v} (\begin{bmatrix} Location \\ Fleet \\ Domicile \\ DOThrs \\ DaysFromHome \end{bmatrix})
\]
Using different levels of aggregation:

» Pick the (single) level of aggregation that produces the best overall results.

» Pick the level of aggregation that produces the lowest variance for each state.

» Use a weighted sum of estimates at each level of aggregation (weight depends only on the level of aggregation):

\[
\overline{v}_a^n = \sum_g w^{(g,n)} \overline{v}_a^{(g,n)}
\]

» Use a weighted sum, but where the weights depend on the state (attribute):

\[
\overline{v}_a^n = \sum_g w^{(g,n)} \overline{v}_a^{(g,n)}
\]
State-dependent weighted aggregation:

There may be hundreds of thousands of weights, so these have to be easy to compute.

\[ \bar{v}_a = \sum_g w_a^{(g)} \bar{v}_a^{(g)} \quad \sum_g w_a^{(g)} = 1 \]

where

\[ w_a^{(g)} \propto \left( \text{Var} \left( \bar{v}_a^{(g)} \right) + \left( \beta_a^{(g)} \right)^2 \right)^{-1} \]

Estimate of variance  Estimate of bias

Both can be computed using simple recursive formulas.
Aggregation for table lookup

Weight on most disaggregate level

Weight on most aggregate levels

Aggregation level

© 2007 Warren D. Powell
Aggregation for table lookup

© 2007 Warren B. Powell
Aggregation for table lookup

- Weighted Combination
- Aggregate
- Disaggregate

© 2007 Warren B. Powell
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management

- Approximation methods
  - Lookup tables and aggregation
  - Basis functions

- Stepsizes
- Exploration vs. exploitation
- Applications
Basis functions

- Aggregation works well when we have a state (attribute) space with very little structure.

- But what if we have some structure? Consider our inventory problem:

\[
x_t^n = \arg \max_x \left( p x_t^D - c x_t^O + \overline{V}_t \left( R_t^x \right) \right)
\]
Basis functions

- Approximating the value function:
  » We have to exploit the structure of the value function (e.g. concavity).
  » We might approximate the value function using a simple polynomial

\[
\overline{V}_t \left( R_t \mid \theta \right) = \theta_0 + \theta_1 R_t + \theta_2 R_t^2
\]

 » .. or a complicated one:

\[
\overline{V}_t \left( R_t \mid \theta \right) = \theta_0 + \theta_1 R_t + \theta_2 R_t^2 + \theta_3 \ln (R_t) + \theta_4 \sin(R_t)
\]

 » Sometimes, they get really messy:
\[ \bar{V}_t(R | \theta) = \theta^{(0)} + \sum_{s} \sum_{t'=t}^{t+2} \theta_{t,st}^{(1)} R_{t,st} + \sum_{w} \sum_{t'=t}^{t+3} \theta_{t,wt'}^{(1)} R_{t,wt'} + \sum_{w} \sum_{t'=t}^{t+2} \theta_{t,wt}^{(2)} R_{t,wt}^2 + \sum_{s} \sum_{t'} \theta_{t,st}^{(2)} R_{t,st}^2 \]

\[ + \sum_{s} \theta_{ts}^{(3)} \left( R_{t,st} - \frac{1}{S} \sum_{s'} R_{t,s't} \right)^2 \]

\[ + \sum_{s} \theta_{ts}^{(4)} \left( \sum_{t'=t}^{t+1} R_{t,st'} \right) - \frac{1}{2S} \sum_{s'} \sum_{t'=t}^{t+1} R_{t,s't'}^2 \]

\[ + \sum_{s} \theta_{ts}^{(5)} \left( \sum_{t'=t}^{t+2} R_{t,st'} \right) - \frac{1}{3S} \sum_{s'} \sum_{t'=t}^{t+2} R_{t,s't'}^2 \]

\[ + \sum_{w} \sum_{s} \theta_{t,ws}^{(ws,1)} R_{t,wt} R_{t,st} \]

\[ + \sum_{w} \sum_{s} \theta_{t,ws}^{(ws,2)} R_{t,wt} \left( \sum_{t'=t}^{t+2} R_{t,st'} \right) \]

\[ + \sum_{w} \sum_{s} \theta_{t,ws}^{(ws,3)} \left( \sum_{t'=t}^{t+3} R_{t,wt'} \right) \left( \sum_{t'=t}^{t+2} R_{t,st'} \right) \]

\[ + \sum_{w} \sum_{s} \theta_{t,ws}^{(ws,4)} \left( \sum_{t'=t}^{t+2} R_{t,wt'} \right) \left( \sum_{t'=t}^{t+2} R_{t,st'} \right) \]

\[ + \sum_{w} \sum_{s} \theta_{t,sw}^{(ws,5)} R_{t,s,t+2} R_{t,wt} R_{t,w,t+2} \]
Basis functions

- We can write a model of the observed value of being in a state as:

\[ \hat{v} = \theta_0 + \theta_1 R_t + \theta_2 R_t^2 + \theta_3 \ln(R_t) + \theta_4 \sin(R_t) + \varepsilon \]

- This is often written as a generic regression model:

\[ Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \theta_4 X_4 \]

- The ADP community refers to the independent variables as \textit{basis functions}:

\[ Y = \theta_0 \varphi_0 (R) + \theta_1 \varphi_1 (R) + \theta_2 \varphi_2 (R) + \theta_3 \varphi_3 (R) + \theta_4 \varphi_4 (R) \]

\[ = \sum_{f \in \mathcal{F}} \theta_f \varphi_f (R) \quad \varphi_f (R) \text{ are also known as features.} \]
Basis functions

Methods for estimating $\theta$

» Generate observations $\hat{v}^1, \hat{v}^2, \ldots, \hat{v}^N$, and use traditional regression methods to fit $\theta$.

» Use recursive statistics - update $\theta^n$ after each iteration:

$$
\theta^n = \theta^{n-1} - \alpha_{n-1} \left( \nabla \bar{V}^{n-1}(S^n | \theta^{n-1}) - \hat{v}^n \right) \bar{V}^{n-1}(S^n | \theta^{n-1})
$$

$$
= \theta^{n-1} - \alpha_{n-1} \left( \nabla \bar{V}^{n-1}(S^n | \theta^{n-1}) - \hat{v}^n \right) \begin{pmatrix}
\phi_1(S) \\
\phi_2(S) \\
\phi_F(S)
\end{pmatrix}
$$

Error

Basis functions
Basis functions

Notes:

» When using basis functions, we are basically drawing on the entire field of statistics.

» Designing basis functions (independent variables) is mostly art.

» In special cases, the resulting algorithm can produce optimal solutions.

» Most of the time, we are hoping for “good” solutions.

» In some cases, it can work terribly.

» As a general rule – you have to use problem structure. Value function approximations have to capture the right structure. Blind use of polynomials will rarely be successful.
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Stepsizes

Stepsizes:

» Fundamental to ADP is an updating equation that looks like:

\[ V_{t-1}^n(S_{t-1}^x) = (1 - \alpha_{n-1})V_{t-1}^{n-1}(S_{t-1}^x) + \alpha_{n-1}\hat{y}_t^n \]

- Updated estimate
- Old estimate
- New observation

The stepsize

“Learning rate”

“Smoothing factor”
Stepsizes

Theory:
» Many convergence results require:
\[ \sum_{n=1}^{\infty} \alpha_{n-1} = \infty \]
\[ \sum_{n=1}^{\infty} \left( \alpha_{n-1} \right)^2 < \infty \]
» For example:
\[ \alpha_{n-1} = \frac{1}{n} \]

Practice
» 1/n “doesn’t work”
» Constant stepsizes
» Various stepsize rules
  • Deterministic
  • Stochastic
Rate of convergence

Smoothed estimate using $1/n$
Rate of convergence

- The challenge of stepsizes:
  - When have we converged?

We need to improve our understanding of adaptive stepsizes.
Stepsizes

- Deterministic stepsize rules:

\[
\alpha_n = \frac{1}{n} \quad \text{or} \quad \frac{1}{n^\beta}
\]

Basic averaging

\[
\alpha_n = \frac{a}{a + n - 1}
\]

Slows the rate of descent

\[
\alpha_n = \frac{b}{b + a + n^\beta}
\]

"Search then converge"

\[
\alpha_n = \frac{\alpha_n}{1 + \alpha_n - \bar{\alpha}}
\]

McClain's formula
Stepsizes

- The right stepsize rule depends on the rate of change in the value function.
- This varies widely for different parameters in the same problem:
Stepsizes

- The right stepsize rule depends on the rate of change in the value function.
- This varies widely for different parameters in the same problem:
Stepsizes

- The right stepsize rule depends on the rate of change in the value function.
- This varies widely for different parameters in the same problem:

![Graph showing different iterations and value slopes with a note for small stepsize.]
Stepsizes

Stochastic stepsize rules:

Let: \( \varepsilon^n = \text{New observation} - \text{old estimate} \)

Kesten's rule: Tunable parameters

\[
\alpha_n = \alpha^0 \frac{a}{a + K^n} \quad K^n = K^{n-1} + 1_{\{\varepsilon^n \varepsilon^{n-1} < 0\}}
\]

Stochastic gradient adaptive stepsize (Benveniste et al.)

\[
\alpha_n = \left[ \alpha^{n-1} + a \varepsilon^n \varphi^n \right]_{\alpha_{\min}}^{\alpha_{\max}}
\]

where:

\[
\varphi^n = \left( 1 - \alpha^{n-1} \right) \varphi^n + \varepsilon^n
\]
Stepsizes

- Bias-adjusted Kalman filter

\[ \alpha_n = 1 - \frac{\sigma^2}{\left(1 + \lambda^{n-1}\right) \sigma^2 + \beta^n} \]

where:

\[ \lambda^n = \left(1 - \alpha_n\right)^2 \lambda^{n-1} + \left(\alpha_n\right)^2 \]

As \( \sigma^2 \) increases, stepsize decreases.

As \( \beta^n \) increases, stepsize increases.
Stepsizes

- Bias-adjusted Kalman filter

  » Properties:

  \[ \alpha_n \to 1 \quad \text{as} \quad \sigma^2 \to 0 \]

  \[ \alpha_n \to 1/n \quad \text{as} \quad \beta \to 0 \quad \text{or} \quad \sigma^2 \to \infty \]
Stepsizes

- Deterministic data: predictions and stepsizes
Stepsizes

- Low noise stochastic data: predictions
Stepsizes

- Low noise stochastic data: stepsizes

![Graph showing stepsizes for different data sets]

- OSA
- $1/n$
- Constant
Stepsizes

- High noise stochastic data: predictions
Stepsizes

High noise stochastic data: stepsizes

OSA
1/n
Constant

© 2007 Warren B. Powell
Stepsizes

- Bias-adjusted Kalman filter learning rate vs. other

[Graph showing percentage error from optimal vs. average number of observations per state.]

Other stochastic stepsize rules

Bias-adjusted Kalman filter
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Exploration vs. exploitation
Exploration vs. exploitation

What decision do we make?

» The one we think is best?
  • Exploitation

» Or do we make a decision just to try something and learn more about the result?
  • Exploration
Exploration vs. exploitation

- Exploration vs. exploitation with the nomadic trucker

  » Pure exploitation
Information collection

Pure exploitation
Exploration vs. exploitation

- The state variable:

\[ S_i = (a_t, (\bar{V}_t^{n-1}, \bar{\sigma}_t^{n-1})) \]

“Resource state”  Knowledge state
### Resource Allocation

The image contains a mathematical expression and a diagram of the United States. The expression is:

\[
\sum_{a} w_{a}^{0} \bar{V}^{0} \begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix} + w_{a}^{1} \bar{V}^{1} \begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix} + w_{a}^{2} \bar{V}^{2} \begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix}
\]

The diagram shows a map of the United States with different states shaded in various colors. The map includes several states and the numbers 1, 2, and 3, possibly indicating different regions or categories.

---

**Diagram Notes**

- The expression involves summation over states, \( a \), with weights \( w_{a}^{i} \) and matrices \( \bar{V}^{i} \) for different indices \( i \). Each matrix has two elements: \( a_{11}, a_{12} \).

---

**Map Notes**

- The map divides the United States into different regions, each possibly representing different resource allocations or scenarios.

---

**Mathematical Notes**

- This expression might be related to a model for resource allocation, where the goal is to optimize the distribution of resources across different states or regions.
Exploration vs. exploitation

- Strategies for overcoming the exploitation trap

  » Generalization:
    - Visit one state, learn something about other states
    - Exploitation with more general learning

\[
\max (\sigma^{-\Gamma} \sum M + d_t x) \quad \text{Std. dev. of } (\sigma^{-\Gamma} \sum M)
\]
Information collection

Pure exploration
Outline

- The languages of dynamic programming
- A resource allocation model
- The post-decision state variable
- Example: A discrete resource: the nomadic trucker
- The states of our system
- Example: A continuous resource: blood inventory management
- Approximation methods
  - Lookup tables and aggregation
  - Basis functions
- Stepsizes
- Exploration vs. exploitation
- Applications
Case study: truckload trucking

Revenue per WU

Utilization

Historical maximum
Simulation
Historical minimum

© 2007 Warren B. Powell
Case study: truckload trucking

Average LOH for Solos

Vanilla simulator
Using approximate dynamic programming
Acceptable region
Case study: truckload trucking

Simulation objective function

# of drivers with attribute a

© 2007 Warren B. Powell
Case study: truckload trucking

Simulation objective function

# of drivers with attribute a

© 2007 Warren B. Powell
Case study: truckload trucking
Case study: truckload trucking

Objective function vs. Iterations

© 2007 Warren B. Powell
Case study: truckload trucking

Mix of drivers changed based on value functions
Case study: truckload trucking

Mix of drivers changed based on value functions

© 2007 Warren B. Powell
Case study: truckload trucking
Case study: truckload trucking
Implementation metrics

- Results from the real world:

![Bar chart showing the comparison between History and Model for different categories: Setouts, Swaps, Nonpreferred consists, Underpowered, and Overpowered. The chart includes percentages for each category.](image)
The planning process
The flow of information

In practice, there are a number of parallel information processes taking place:

- Order is made
- Empty transit time to shipper becomes known.
- Customer inspects car and accepts or rejects.
- Customer loads car (we learn the release time)
- Destination of order becomes known

Time
Multiattribute resources

Information at time $t$

Actionable time

Locations

$1$ $2$ $3$ $4$ $5$

$1$ $2$ $3$ $4$ $5$

$\text{?}$ $\text{?}$ $\text{?}$ $\text{?}$ $\text{?}$
Multiattribute resources

Information at time $t+1$

Actionable time

1

2

3

4

5

© 2007 Warren B. Powell
Multiattribute resources

Information at time t+2

Actionable time

Locations

© 2007 Warren B. Powell
Engineering practice

Cars

Orders
A car distribution problem

Empty miles as a percent of total miles

History

Basic optimization model (engineering practice)
Engineering practice

Cars

Orders
Engineering practice

Assignments to booked orders.

Repositioning movements based on forecasts
A car distribution problem

Empty miles as a percent of total miles

- History
- Basic optimization model (engineering practice)
A car distribution problem

Empty miles as a percent of total miles

History

Basic optimization model (engineering practice)

“Optimized” with adaptive learning
Iteration 2
Tanker study

![Graph showing average tankers in flight per period over iterations. The graph plots iteration against the number of tankers, showing a decreasing trend.]