Decision Making As An Optimization Problem

Hala Mostafa

683 Lecture 14
Wed Oct 27th 2010
DEC-MDP

• Formulation as a math’al program
  – Often requires good insight into the problem to get a compact ‘well-behaved’ formulation
  – Can be very un-intuitive
  – Math’al programming may turn out to be ill-suited for the problem

Wait! Why bother?

Premise:
There are great, industrial-strength, highly optimized solvers out there. Use them!
Outline

• Linear Programming
  – MDP policy finding as LP #1
  – Sequence form
  – MDP policy finding as LP #2
• DEC-POMDP and DEC-MDP
• Quadratic & Bilinear Programming
  – 2-Agent DEC-MDP as Bilinear Program
• Nonlinear Programming
• Mixed Integer LP
Linear Programming

max \quad c^T x
subject to \quad Ax \leq b
• Both objective function and constraints are linear (x is only multiplied by constants)
• Can model equality constraints using the above. Ex = f can be expressed as
  \quad Ex - f \leq 0 \quad \text{and} \quad -Ex + f \leq 0
Solvers typically accept this form

min \quad c^T x
subject to \quad Ax \leq b
\quad Ex = f
\quad lb \leq x \leq ub
MDP as LP#1

• MDP goal: find the policy that maximizes reward over the T steps of the problem
• Policy maps states to distributions over actions
• Pure policy assigns all probability at a state to a single action. Deterministic.
• **Policy representation #1**: for every state s and every action at that state a, \( x(s,a) = \text{probability of doing a at s. Occupancy measure} \)
MDP as LP#1

• **Objective function:**
  \[ \max \sum_s \sum_a x(s, a) \cdot r(s, a) \]

• **Constraints:**
  – For start state \( s \): prob going out of \( s \) = 1
    \[ \sum_a x(s, a) = 1 \]
  – For every non-leaf state \( s \):
    prob going out of \( s \) = prob going into \( s \)
    \[ \sum_a x(s, a) = \sum_{s'} \sum_{a'} x(s', a') P(s|s', a') \]
  – For every state, action: \( x(s, a) \geq 0 \)
max \ c^T x \quad \text{subject to} \quad A x = b \\
\text{where} \quad x = [x(s1,a1), x(s1,a2), x(s2,a3), x(s2,a4), x(s3,a5), x(s3,a6)] \\
c = [5,2,4,4,3,5] \\
Constraints:
\begin{align*}
  x(s1,a1) + x(s1,a2) &= 1 \\
  x(s2,a3) + x(s2,a4) &= x(s1,a1) \\
  x(s3,a5) + x(s3,a6) &= x(s1,a2)
\end{align*}

Thus \quad A = [\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 1
\end{bmatrix}]

b = [1 \ 0 \ 0]^T
Sequence Form

- Sequence: a path through the MDP, starting at the root \([s_{t_1}, a_1, s_{t_2}, a_2, \ldots, s_{t_n}, a_n]\)
- Complete seq: ends at a leaf
- Information set \(\psi\): a decision-making point \([s_{t_1}, a_1, s_{t_2}, a_2, \ldots, s_{t_n}]\)
- \((\psi.a)\) is the sequence obtained from doing action \(a\) at info set \(\psi\)
- A policy can be characterized by the weight it assigns each sequence
- **Policy representation #2**: \(x(s) = \) realization weight of sequence \(s\). Product of action probabilities on the sequence
MDP as LP#2

- **Obj fun:** max $\sum_{s \in C} x(s).r(s)$
  - $r(s)$ is the reward for the complete seq $s$
- **Constraints:**
  - $\Sigma_a x(st_0.a) = 1$
  - sum of child seqs = weighted parent seq
    - $\Sigma_a x(s.st.a) = x(s)P(st|s)$ for every seq $s$ and next state $st$
  - $x(s) \geq 0$
\[ s_1 = [st_1, a_1, st_2, a_2] \]

\[ s_2 + s_3 = 0.3s_1 \]

\[ s_4 + s_5 = 0.7s_1 \]
Note

• W/ probabilistic outcomes, multiple full sequences can have non-zero weights under a pure policy
  – It’s not like we’re choosing the one sequence to assign max weight to and setting others=0

• How many sequences can be non-zero simultaneously? Depends on the horizon and # action outcomes
[s1, a1, s3, a3] = [s1, a1, s4, a4] = 1
Support size = 2
Multi-Agent Decision Making
Tiger Problem

- Want to open the door with the treasure
- Can listen or open a door, but listening
  - is noisy: may hear tiger on left when it’s right
  - has a cost
- Agent do not communicate their observations
- Need to reason over what the other agent may be hearing
DEC-POMDP

- DEC-POMDP is a tuple \(<S, A, P, R, \Omega, O>\)
  - \(S\) is a finite set of world states
  - \(A = A_1 \times A_2 \times \ldots \times A_n\) is a finite set of joint actions
  - \(P : S \times A \times S \rightarrow [0, 1]\) is the transition function
  - \(R : S \times A \times S \rightarrow \mathbb{R}\) is the reward function
  - \(\Omega = \Omega_1 \times \Omega_2 \times \ldots \Omega_n\) is a finite set of joint observations
  - \(O : S \times A \times \Omega \rightarrow [0, 1]\) is the observation function

- Observations of all agents do not necessarily determine the global state
- An agent’s policy maps each observation history to an action
- Joint policy: a tuple with one policy per agent
Tiger DEC-POMDP

\[ S = \{ \text{TigerL, TigerR} \} \]
\[ A_1 = A_2 = \{ \text{OpenL, OpenR, Listen} \} \]
\[ \Omega_1 = \Omega_2 = \{ \text{NoiseL, NoiseR} \} \]

<table>
<thead>
<tr>
<th>Joint Action</th>
<th>State</th>
<th>Joint Observation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Listen, Listen)</td>
<td>Left</td>
<td>(Noise Left, Noise Left)</td>
<td>0.7225</td>
</tr>
<tr>
<td>(Listen, Listen)</td>
<td>Left</td>
<td>(Noise Left, Noise Right)</td>
<td>0.1275</td>
</tr>
<tr>
<td>(Listen, Listen)</td>
<td>Left</td>
<td>(Noise Right, Noise Left)</td>
<td>0.1275</td>
</tr>
<tr>
<td>(Listen, Listen)</td>
<td>Left</td>
<td>(Noise Right, Noise Right)</td>
<td>0.0225</td>
</tr>
</tbody>
</table>
DEC-MDP

• Putting observations together determines global state
• We’ll consider DEC-MDPs with local observability. Each agent knows its own state
• Agent’s policy maps its local states to local actions
Mars Rovers

- Rovers collect samples from Mars.
- Each site has rock quality (reward) and probability of being easy (transition function) that depends on what *both* rovers do.
- A rover knows which site it is at, but not where the other is.
Mars Rovers DEC-MDP

- $S_i = <$site,outcome$>$ pairs visited by rover $i$
- $A_i = $ set of sites unvisited by rover $i$
- $r(s_1,s_2,a_1,a_2) = $ reward when rovers visit sites $a_1,a_2$ respectively

- Reward interactions: collecting samples can be redundant or complementary
- Transition interactions: rovers can get in each other’s way, or help each other finish a site faster
\begin{align*}
    r(*,*,a_1,a_1) &= 10 & r(*,*,a_1,a_{k\neq 1}) &= 3 + r(a_k) & \text{complementary} \\
    r(*,*,a_2,a_2) &= 3 & r(*,*,a_2,a_{k\neq 2}) &= 3 + r(a_k) & \text{redundant} \\
    p(\text{site2 fast} \mid a_3,a_3) &= 0.9 & \text{rovers help each other} \\
    p(\text{site2 fast} \mid a_3,a_{k\neq 3}) &= 0.2 & \text{rovers get in the way}
\end{align*}
QP

- Quadratic program has the variable raised to a max power of 2

Maximize \( f(x) = \frac{1}{2} x^T Q x + c^T x \)
subject to \( Ax \leq b \)
\( Ex = d \)

- If \( Q=0 \), we have an LP

\[
5x_1^2 - 2x_1x_2 - x_1x_3 + 2x_2^2 + 3x_2x_3 + 10x_3^2 + 2x_1 - 35x_2 - 47x_3
\]

- \( Q = \begin{bmatrix} 10 & -2 & -1 \\ -2 & 4 & 3 \\ -1 & 3 & 10 \end{bmatrix} \)

- \( c = [2 \quad -35 \quad -47] \)
BLP

• Special case of QP
• Separable bilinear program

Maximize $f(x,y) = r_1^Tx + x^TCy + r_2^Ty$

subject to

$A_1x = b_1$

$A_2y = b_2$

$x, y \geq 0$
DEC-MDP as BLP

Maximize \( f(x,y) = \sum_i r_i^T x \) + \( \sum_j x^T C_j y \) + \( r_2^T y \)
subject to
\[
\begin{align*}
A_1 x &= b_1 \\
A_2 y &= b_2 \\
x, y &\geq 0
\end{align*}
\]

Rewards of \( i \)

Rewards of \( j \)

Guarantee that realization weights represent a legal policy

Rewards that depend on both \( i \) and \( j \).

This assumes realization weights of each agent are independent of the other
i.e. Transition independence
DEC-MDP as BLP

- C can’t contain raw rewards
- C is the only term that can capture interactions between agents

→ Slightly different formulation
- Move transition probabilities *from constraints to C*
- Constraints:
  \[ \Sigma_a x(s_{t0}.a) = 1 \]
  \[ \Sigma_a x(s.st.a) = x(s) \text{ for every seq s and next state st} \]
DEC-MDP as BLP

• $C(i,j) = r(i,j) \times P(i|j) \times P(j|i)$

• $P(i|j)$ = product of chance outcomes along seq i given actions of both agents along seqs i and j

• $C$ now captures all the interactions
NLP

- Variable appears in arbitrary expression (raised to any power, in trig functions..anything!)
- Objective function isn’t represented in matrix notation
- Solver takes pointer to function that takes variable vector and returns value
- Same for nonlinear constraints. Take variable vector and return value of constraint, assuming this value should be $\leq 0$
Max objFun(x)
Subject to
\[ A_{eq} x = b_{eq} \]
\[ A_{ineq} x \leq b_{ineq} \]
\[ C(x) \leq 0 \]
\[ LB \leq x \leq UB \]
solver(objFunPtr, A_{eq}, b_{eq}, A_{ineq}, b_{ineq}, CPtr, LB, UB, x0)
NLP Examples

• Min $x \cdot \sin(3.14159x)$
  subject to $0 \leq x \leq 6$

• Max $2x_1 + x_2 - 5 \log_e(x_1) \sin(x_2)$
  subject to $x_1x_2 \leq 10$
    $|x_1 - x_2| \leq 2$
    $0.1 \leq x_1 \leq 5$
    $0.1 \leq x_2 \leq 3$
NLP Examples

double objFun(x)
end
double constrFun(x)
    double ret[2]
    return ret
end
LB = [0.1 0.1]      UB = [5 3]
solver(@objFun,[],[],[],[],@constrFun, LB, UB)
DEC-MDP as NLP

- $x$ is vector of realization weights of all agents’ sequences, i.e. a joint policy
- $\text{objFun}$ returns the value of the given joint policy
  \[ \text{objFun} = \sum_i \sum_j \sum_k x_i x_j x_k r(i,j,k) \]
- $A_{\text{ineq}}, b_{\text{ineq}}, \text{CPtr} = []$
NLP

• With LP, QP and BP, solver can easily determine how obj fun & constraints vary as the components in x vary, i.e. first order derivatives
• Derivatives help follow the shape of the obj fun & constraints
• In NLP, obj fun is a black box!
  – Solver has no information how to move from one search point (a value of x) to the next
• Providing solver with first derivatives helps a LOT
MIP

• Mixed Integer programming has some continuous variables and some integer (or boolean) variables

• Why?
  – Integer: # persons assigned to a job, # airplanes manufactured
  – Boolean: indicator variables representing decisions. “Should we use the n\textsuperscript{th} machine?”

• MILP harder than LP
  – With LP, optimal solution is at corner of feasible region. Not so with MILP
  – Use as few integer variables as possible
  – Solve the problem w/o integrality constraints to get an initial upper bound (for max problem)
Software

• Mosek (free for academic purposes)
  – LP, convex QP (for which easy to get global opt.)
  – MIP

• Knitro (free for academic purposes)
  – LP, convex and non-convex QP
  – NLP
  – MINLP

• CPLEX (free under IBM Academic Initiative)
  – LP, convex QP
  – MILP

• All 3 have Matlab interfaces
Bibliography
