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 $c^T x$

subject to $Ax \leq b$

- 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

  $Ex - f \leq 0$ and $-Ex + f \leq 0$

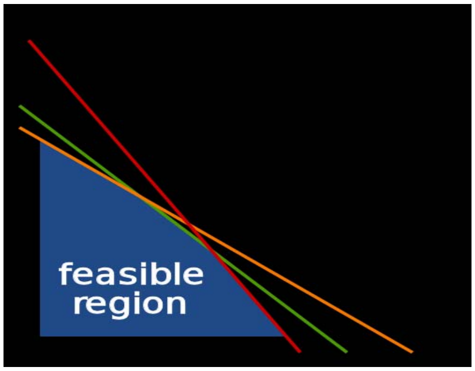
Solvers typically accept this form

min $c^T x$

subject to $Ax \leq b$

$Ex = f$

$lb \leq x \leq ub$

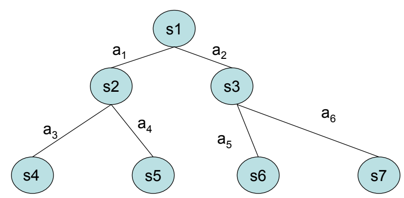


MDP as LP#1

- 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.
- <u>Policy representation #1</u>: for every state s and every action at that state a, x(s,a) = probability of doing a at s. Occupancy measure

MDP as LP#1

- Objective function:
  max $\Sigma_s \Sigma_a x(s,a).r(s,a)$
- Constraints:
  - For start state s: prob going out of s = 1
    $\Sigma_a x(s,a) = 1$
  - For every non-leaf state s:
    prob going out of s = prob going into s
    $\Sigma_a x(s,a) = \Sigma_{s'} \Sigma_{a'} x(s',a')P(s|s',a')$
  - For every state, action: x(s,a) ≥ 0



r(s1,a1) = 5
r(s1,a2) = 2
r(s2,a3) = 3
r(s2,a4) = 4
r(s3,a5) = 3
r(s3,a6) = 5

max cx subject to Ax = b
where 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:
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)
Thus A = [1 1 0 0 0 0;
-1 0 1 1 0 0;
0 -1 0 0 1 1]
b = [1 0 0]^T

Sequence Form

- Sequence: a path through the MDP, starting at the root $[st_1,a_1,st_2,a_2,\ldots st_n,a_n]$
- Complete seq: ends at a leaf
- Information set ψ: a decision-making point $[st_1,a_1,st_2,a_2,\ldots st_n]$
- (ψ.a) is the sequence obtained from doing action a at info set ψ
- 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

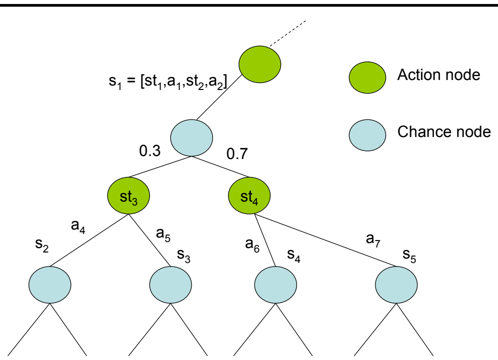
MDP as LP#2

- Obj fun: max $\Sigma_{s \text{ in } C}\, x(s).r(s)$

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
  - $x(s) \geq 0$

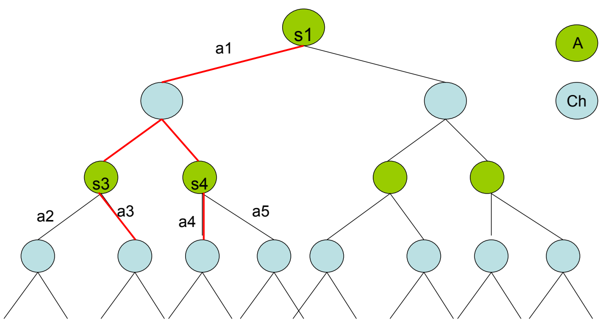


$s_2 + s_3 = 0.3s_1$

$s_4 + s_5 = 0.7s_1$

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 a tuple <S, A, P,R, Ω,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 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

TigerL	OpenL	OpenR	Listen
Listen	-200	9	-2
OpenL	-50	-100	-200
OpenR	-100	20	9

S = {TigerL, TigerR}
A1 = A2 ={OpenL, OpenR, Listen}
Ω1 = Ω2 = {NoiseL, NoiseR}

Observation function

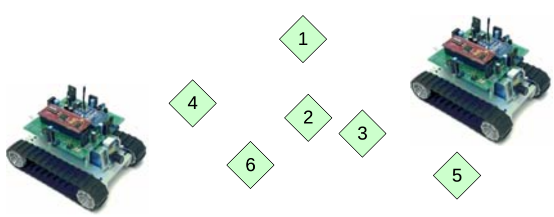
Joint Action	State	Joint Observation	Probability
(Listen, Listen)	Left	(Noise Left, Noise Left)	0.7225
(Listen, Listen)	Left	(Noise Left, Noise Right)	0.1275
(Listen, Listen)	Left	(Noise Right, Noise Left)	0.1275
(Listen, Listen)	Left	(Noise Right, Noise Right)	0.0225

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

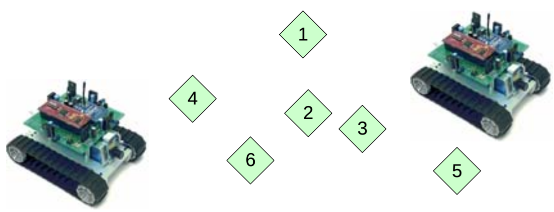
- S_i = <site,outcome> pairs visited by rover i
- A_i = set of sites unvisited by rover i
- $r(s_1,s_2,a_1,a_2)$ = reward when rovers visit sites a_1,a_2 respectively
- Reward interactions: collecting samples can be redundant or complementary
- Transition interactions: rovers can get in each other's way,or help each other finish a site faster

$r(*,*,a_1,a_1) = 10$ $r(*,*,a_1,a_{k\neq 1}) = 3 + r(a_k)$ complementary

$r(*,*,a_2,a_2) = 3$ $r(*,*,a_2,a_{k\neq 2}) = 3 + r(a_k)$ redundant

$p(\text{site2 fast} \mid a_3,a_3) = 0.9$ rovers help each other

$p(\text{site2 fast} \mid a_3,a_{k\neq 3}) = 0.2$ rovers get in the way



QP

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

Maximize $f(x) = \frac{1}{2} x^TQx + c^Tx$

subject to $Ax \le b$

$Ex = d$

- If Q=0, we have an LP
- $5x_1^2 - 2x_1x_2 - x_1x_3 + 2x_2^2 + 3x_2x_3 + 10x_3^2 + 2x_1 - 35x_2 - 47x_3$
- $Q = \begin{matrix} 10 & -2 & -1 \\ -2 & 4 & 3 \\ -1 & 3 & 10 \end{matrix}$
- $c = [2 \quad -35 \quad -47]$

BLP

- Special case of QP
- Separable bilinear program

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

subject to $A_1x = b_1$

$A_2y = b_2$

$x, y \ge 0$

DEC-MDP as BLP

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

subject to $\begin{cases} A_1x = b_1 \\ A_2y = b_2 \end{cases}$

$x, y \ge 0$

Rewards of *i*

Rewards of *j*

rewards that depend on both *i* and *j*.

Guarantee that realization weights represent a legal policy

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:

  Σ_a x(st_0.a) = 1

  Σ_a x(s.st.a) = x(s) for every seq s and next state st

DEC-MDP as BLP

- C(i,j) = r(i,j) * P(i|j) * 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 ≤ 0

NLP

Max	objFun(x)
Subject to	A_{eq} x = b_{eq}
	$A_{ineq}x \le b_{ineq}$
	C(x) ≤ 0
	LB ≤ x ≤ UB

solver(objFunPtr,A_{eq},b_{eq},A_{ineq},b_{ineq},CPtr, LB,UB,x0)

NLP Examples

- Min $x*\sin(3.14159x)$

subject to $0 \le x \le 6$

- Max $2x_1 + x_2 - 5\log_e(x_1)\sin(x_2)$

  subject to $x_1x_2 \le 10$

  $| x_1 - x_2 | \le 2$

  $0.1 \le x1 \le 5$

  $0.1 \le x2 \le 3$

NLP Examples

```
double objFun(x)
   return 2x[2] + x[2] – 5*log_e(x[1])* sin(x[2])
end
double constrFun(x)
   double ret[2]
   ret[1] = x[1]*x[2] – 10
   ret[2] = abs(x[1] – x[2]) – 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
- objFun returns the value of the given joint policy

  objFun = $\Sigma_i \Sigma_j \Sigma_k x_i x_j x_k r(i,j,k)$
- A_{ineq},b_{ineq},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 decsions. “Should we use the n^{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

- Formal Models and Algorithms for Decentralized Decision Making under Uncertainty. S. Seuken and S. Zilberstein. Autonomous Agents and Multi-Agent Systems, 2008.
- A Bilinear Programming Approach for Multiagent Planning. Marek Petrik and Shlomo Zilberstein. Journal of Artificial Intelligence Research, 2009.