Similarities and differences between stochastic programming, dynamic programming and optimal control

Václav Kozmík

Faculty of Mathematics and Physics
Charles University in Prague

11 / 1 / 2012

Stochastic Optimization

- Different communities focus on special applications in mind
  - Therefore they build different models
  - Notation differs even for the terms that are in fact same in all communities
- The communities are starting to merge
  - Ideas and algorithms may be useful in all communities
- We will focus on:
  - Stochastic programming
  - Dynamic programming
  - Optimal control

Stochastic programming

- Basic model (Shapiro et al., 2009)

$$\min_{x_1 \in \mathcal{X}_1} f_1(x_1) + \mathbb{E}\left[\inf_{x_2 \in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \mathbb{E}\left[\inf_{x_3 \in \mathcal{X}_3(x_2,\xi_3)} f_3(x_3,\xi_3) + \cdots \right.\right.$$
$$\left.\left.\left. + \cdots \mathbb{E}\left[\inf_{x_T \in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)\right]\right]\right]$$

- Decisions x_t are typically real-valued vectors
  - Integer values are possible, but significantly harder to solve
- Decisions x_t do not influence probability distributions of $\xi_{t'}$ $\forall t'$
- We require nonanticipativity: x_t is measurable w.r.t. $\sigma(\xi_{[t]})$

Stochastic programming

- We can develop dynamic programming equations

$$\min_{x_1} \; f_1(x_1) + \mathbb{E}\left[Q_2(x_1, \xi_2)\right]$$
$$\text{s.t. } x_1 \in \mathcal{X}_1$$

$$Q_t(x_{t-1}, \xi_t) = \inf_{x_t} \; f_t(x_t, \xi_t) + \mathbb{E}\left[Q_{t+1}(x_t, \xi_{[t+1]}) | \xi_{[t]}\right]$$
$$\text{s.t. } x_t \in \mathcal{X}_t(x_{t-1}, \xi_t)$$

Dynamic programming

- Basic model (Puterman, 1994)
  - Decision epochs $t = 1, \dots, N$ or $t = 1, 2, \dots$
  - Set of possible system states: S
  - Set of possible actions in the state $s \in S$: A_s
  - Reward function for choosing an action $a \in A_s$ in the state s: $r_t(s, a)$
  - Transition probabilities for the next state of the system $p_t(\cdot|s, a)$
  - We maximize the expected value of all rewards
- Set of states S is usually finite
- Sets of actions A_s are usually finite
- Extensions to countable, compact or complete spaces S and A_s are possible
- We usually seek Markov decision rules $d_t : S \to A_s$
  - Decisions can be also random and history dependent
- Rewards and transition probabilities are typically stationary

Dynamic programming

- Denote random sequence of states X_t
  - X_1 deterministic or specified by a probability distribution
- Following a decision rule d_t we select sequence of actions $Y_t = d_t(X_t)$
- Decisions affect the transition probabilities for following period
- We seek policy π consisting of decision rules d_t:

$$max_\pi \ \mathbb{E}\left[\sum_{1}^{\infty} \lambda^{t-1} r_t(X_t, Y_t)\right]$$

  - Discount factor $\lambda_t \in (0, 1]$
  - In the finite case we have salvage value $r_N(s)$ and maximize $\mathbb{E}\left[\sum_1^N r_t(X_t, Y_t) + r_N(X_N)\right]$

Optimal control

- Initial state $X(0) = x_0$
- State evolves according to stochastic differential equation:

$$dX(t) = f(t, X(t), u(t))\mathrm{d}t + \sigma(t, X(t), u(t))\mathrm{d}W(t)$$

- Set of possible controls U
- Basic model (Fleming, Soner (2006))

$$\min_{u \in U} \mathbb{E} \int_0^T L(t, X(t), u(t))\mathrm{d}t + \psi(X(T))$$

or infinite horizon discounted cost problem $\beta \geq 0$

$$\min_{u \in U} \mathbb{E} \int_0^\infty \exp^{-\beta t} L(X(t), u(t))\mathrm{d}t$$

- Discontinuous control u can be also admitted

Decision epochs

- Stochastic programming
  - Discrete time steps
  - Two-stage problems or problems with modest number of stages (hundreds) are usual
- Dynamic programming
  - Discrete time steps
  - Usually infinite horizon problems with discount
  - Also finite horizon problems with large number of stages can be solved
- Optimal control
  - Continuous time
  - Both finite horizon and infinite horizon problems
    - Random horizon (Markov time) also possible = Optimal stopping

State variable

- Usually models resource state, information state or knowledge about unknown parameters

Definition (Powell (2011))

A state variable s is the minimally dimensioned function of history that is necessary to compute the decision function, the transition function and the contribution function.

- *Every dynamic program is Markovian provided that the state variable is complete*
- In stochastic programming, the decision vector x is the state variable
  - Decisions and states are coupled together
- State vector x in optimal control

Decisions / Actions / Controls

- Different notations:
  - Stochastic programming: decision x
  - Dynamic programming: action a
  - Optimal control: control u
- Typical shape differs (provided by different applications):
  - Decision x is usually high-dimensional vector
  - Action a refers to discrete (or discretized) actions
  - Control u is used for low-dimensional (continuous) vectors
- Stochastic programming puts focus on the first stage decision x_1
- Optimal control community develop controls for the complete horizon
- Both cases are present in dynamic programming

Exogenous information

- Stochastic programming
  - Modeled by scenarios ξ
  - Scenarios influence the constraints of the model
  - Usually ξ_t is assumed to be know at stage t
  - Scenario probabilities are not influenced by our decisions
    - Or: decisions determine when uncertainty is resolved (Grossman, 2006)
- Dynamic programming
  - Exogenous information is encoded in the transition function $p_t(\cdot|s, a)$
    - Called transition kernel in the continuous case
  - Direct observation of the exogenous inputs is possible by including them into the state variables
- Optimal control
  - Random variable W_t, usually Wiener process
    - Not known at time t
    - Natural due to the continuous nature of the problems
  - Not influenced by our decisions

Transition function

- Stochastic programming
  - Transition encoded into the program constraints
  - Usually linear equations of the form

$$B_t x_{t-1} + A_t x_t = b_t$$

- Dynamic programming
  - Model-based problems - the transition matrix is known
  - Model-free problems - complex systems
    - Transition function is known, but the probability law for the exogenous information is not known
- Optimal control
  - Generic transition functions
    - Too general to be used in stochastic programming
    - Usually in the form of stochastic differential equation

Objective function

- Stochastic programming
  - Objective function usually linear or convex

$$f_1(x_1)+\mathbb{E}\left[f_2(x_2,\xi_2)+\mathbb{E}\left[f_3(x_3,\xi_3)+\cdots+\mathbb{E}\left[f_T(x_T,\xi_T)|\xi_{[T-1]}\right]\cdots|\xi_{[2]}\right]\right]$$

  - Does not have to be additive or linear
- Dynamic programming & Optimal Control
  - Usually infinite horizon discounted problem

$$\mathbb{E}\left[\sum_{1}^{\infty}\lambda^{t-1}r_t(X_t,Y_t)\right] \text{ or } \int_0^{\infty}\exp^{-\beta t}L(X(t),u(t))\mathrm{d}t$$

  - Alternatively finite horizon with a terminal cost
  - Additivity is important

Stochastic programming - solution approach

- We usually solve SAA versions of the continuous problems
  - Simple problems can be solved directly with simplex method
- Exploit the special problem structure
  - Recourse functions are polyhedral in the case of linear programs and finite number of scenarios
  - More generally, we rely on the convexity property
  - Lower bounding cuts of the recourse function are constructed to obtain approximate solution
    - Benders' decomposition, L-shaped method
    - Stochastic decomposition
- Decompose the problem by scenarios
  - We solve the problem scenario by scenario and iteratively find solution by penalizing anticipative solutions
  - Progressive hedging (Lagrangian relaxation)
  - Well suited for mixed integer stochastic programs (nonconvex)

Stochastic programming - solution approach

- For multistage programs we have extensions to the classic algorithms:
  - Nested Benders' decomposition
  - Multistage Stochastic decomposition
- But we usually hit the *curse of dimensionality*
  - Number of scenarios grows exponentially with the number of stages
  - Special algorithms usually rely on stage independence assumption
    - Exogeneous inputs are supposed independent
    - Stochastic Dual Dynamic Programming algorithm

Dynamic programming - solution approach

- Focus on deterministic Markov policies
  - They are optimal under various conditions
- Finite horizon problems
  - Backward induction algorithm
  - Enumerates all system states
- Infinite horizon problems
  - Bellmann's equation for value function v

$$v^*(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) v^*(s') \right\}$$

  - Optimal solution guaranteed by fixed-point theorems:

$$v = \max_{d \in D} \{ r_d + \lambda P_d v \} = Lv$$

Dynamic programming - solution approach

- Value iteration
  - Start with arbitrary v^0
  - Iterate while the value function improves significantly

$$v^{n+1}(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) v^n(s') \right\}$$

- Policy iteration
  - Start with arbitrary decision $d_0 \in D$
  - Policy evaluation - obtain v^n

$$(I - \lambda P_{d_n}) v = r_{d_n}$$

  - Policy improvement - find d_{n+1}

$$d_{n+1} \in \arg\max_{d \in D} \{ r_d + \lambda P_d v^n \}$$

- Combination of above - modified policy iteration

Dynamic programming - solution approach

- Generalized notation
  - Reward function $r(s, a, \omega)$
  - Transition function $f(s, a, \omega)$
  - For a given realization ω: $Y_t = d_t(X_t)$, $X_{t+1} = f(X_t, Y_t, \omega)$
- Q-factors
  - Bellman's equation with Q^* as the optimal Q-factor:

$$v^*(s) = \max_{a \in A_s} \{Q^*(s, a)\}$$

$$Q^*(s, a) = \mathbb{E}\left[r(s, a, \omega) + \lambda \max_{a' \in A_{s'}} Q^*(s', a')\right]$$

  - Once Q-factors are known optimization is *model-free*

Dynamic programming - solution approach

- Approximation in value space
  - Approximation architecture: consider only $v(s)$ from a parametric class $v(s, r)$
  - Training the architecture: determine optimal $r \in \mathbb{R}^m$
  - Context-dependent features (basis functions) $\phi(s)$
    - Polynomial approximation, kernels, interpolation, ...
    - Special features, for example in chess: material balance, safety, mobility
  - Linear architecture: $\phi(s)^\top r$
- Approximate Value iteration
  - Select small subset $S_n \subset S$ and compute $\forall s \in S_n$:

$$\tilde{v}^{n+1}(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) \tilde{v}^n(s') \right\}$$

  - Fit the function $\tilde{v}^{n+1}(s)$ $\forall s \in S$ to the set S_n

Dynamic programming - solution approach

- Approximate Policy iteration
  - Guess initial policy
  - Evaluate approximate cost using simulation, $\tilde{v}(s) = \phi(s)^\top r$
    - Cost samples obtained by simulation
    - Weights r optimized through least squares
  - Generate improved policy using linear approx. of the value function
  - Exploration issue - cost samples biased by current optimal policy
    - Randomization, mixture of policies
- Q-learning
  - Sampling: select pairs (s_k, a_k) and select s'_k according to $p(\cdot|s_k, a_k)$
  - Iteration: update just $Q(s_k, a_k)$ with $\gamma_k \sim 1/k$

$$Q(s_k, a_k) = (1 - \gamma_k) Q(s_k, a_k) + \gamma_k \left(r(s_k, a_k, s'_k) + \lambda \max_{a' \in A_{s'}} Q(s'_k, a') \right)$$

  - *model-free*: need only simulator to generate next state and cost

Optimal control - solution approach

- Define cost-to-go function $J(t, X(t))$

$$J(t, X(t)) = \min_{u(t) \in U} \mathbb{E} \int_t^T L(t, X(t), u(t)) \mathrm{d}t + \psi(X(T))$$

- Hamilton-Jacobi-Bellman equation:

$$\begin{aligned} \frac{\partial J(t,x)}{\partial t} + \min_{u(t) \in U} \Bigg\{ & L(t,x,u) + \frac{\partial J(t,x)}{\partial x} f(t,x,u) \\ & + \frac{1}{2} \operatorname{tr} \left\{ \sigma(t,x,u) \sigma^\top(t,x,u) \frac{\partial^2 J(t,x)}{\partial x^2} \right\} \Bigg\} = 0 \end{aligned}$$

$$J(T, X(T)) = \psi(X(T))$$

- Explicit solutions are rarely found
  - Numerical solutions for differential equations

Optimal control - solution approach

- Differential of $J(t, X(t))$ is important only in values along the optimal path
$$p(t) = J_x^*(t, x^*(t))$$
- Define Hamiltonian function $H(t, x, u, p, p_x)$:
$$H(t, x, u, p, p_x) = L(t, x, u) + f(t, x, u)^\top p + \frac{1}{2} \operatorname{tr}\left\{ p_x \sigma(t, x, u) \sigma^\top(t, x, u) \right\}$$
- Pontryagin principle:
$$\begin{cases} \mathrm{d}x^* = H_p^* \mathrm{d}t + \sigma \mathrm{d}W \\ \mathrm{d}p^* = -H_x^* \mathrm{d}t + p_x \sigma \mathrm{d}W \\ x^*(0) = x_0 \\ p^*(T) = \psi_x(T, X(T)) \\ H^*(t, X(t), u(t), p(t), p_x(t)) = \min_u H(t, X(t), u, p(t), p_x(t)) \end{cases}$$
- We usually need to prove optimality

References

- Bertsekas, D. P. (2012): *Dynamic Programming and Optimal Control*, Vol. II, 4th Edition: Approximate Dynamic Programming. Athena Scientific, ISBN 1-886529-44-2.
- Fleming, W. H., Soner, H. M. (2006): *Controlled Markov Processes and Viscosity Solutions*
- Goel, V., Grossmann, I. (2006): A Class of Stochastic Programs with Decision Dependent Uncertainty
- Powell, W. B. (2012): *AI, OR and Control Theory: A Rosetta Stone for Stochastic Optimization*
- Puterman, M. L. (1994): *Markov Decision Processes: Discrete Stochastic Dynamic Programming*
- Shapiro, A., Dentcheva, D., Ruszczynski A. (2009): *Lectures on Stochastic Programming: Modeling and Theory*

Conclusion

Thank you for your attention!

Václav Kozmík
vkozmik@gmail.com