Stochastic Optimal Control – part 2

discrete time, Markov Decision Processes, Reinforcement Learning

Marc Toussaint

Machine Learning & Robotics Group – TU Berlin
mtoussai@cs.tu-berlin.de

ICML 2008, Helsinki, July 5th, 2008

- Why stochasticity?
- Markov Decision Processes
- Bellman optimality equation, Dynamic Programming, Value Iteration
- Reinforcement Learning: learning from experience
Why consider stochasticity?

1) system is inherently stochastic

2) true system is actually deterministic but
   a) system is described on level of abstraction, simplification which makes model approximate and stochastic
   b) sensors/observations are noisy, we never know the exact state
   c) we can handle only a part of the whole system
      – partial knowledge $\rightarrow$ uncertainty
      – decomposed planning; factored state representation

- probabilities are a tool to represent information and uncertainty
  – there are many sources of uncertainty
Machine Learning models of stochastic processes

- Markov Processes
defined by random variables $x_0, x_1, ..$ and transition probabilities

$$P(x_{t+1} \mid x_t)$$

- non-Markovian Processes
  - higher order Markov Processes, auto regression models
  - structured models (hierarchical, grammars, text models)
  - Gaussian processes (both, discrete and continuous time)
  - etc

- continuous time processes
  - stochastic differential equations
Markov Decision Processes

- Markov Process on the random variables of states $x_t$, actions $a_t$, and rewards $r_t$

![Diagram of Markov Decision Processes]

\[ P(x_{t+1} \mid a_t, x_t) \]  
transition probability  \hspace{1cm} (1)

\[ P(r_t \mid a_t, x_t) \]  
reward probability  \hspace{1cm} (2)

\[ P(a_t \mid x_t) = \pi(a_t \mid x_t) \]  
policy  \hspace{1cm} (3)

- we will assume stationarity, no explicit dependency on time
  - $P(x' \mid a, x)$ and $P(r \mid a, x)$ are invariable properties of the world
  - the policy $\pi$ is a property of the agent
optimal policies

- value (*expected* discounted return) of policy $\pi$ when started in $x$

$$V^\pi(x) = \mathbb{E}\left\{ r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid x_0 = x; \pi \right\}$$

(cf. cost function $C(x_0, a_0:T) = \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, a_t)$)

- optimal value function:

$$V^*(x) = \max_{\pi} V^\pi(x)$$

- policy $\pi^*$ if optimal iff

$$\forall x : V^{\pi^*}(x) = V^*(x)$$

(simultaneously maximizing the value in all states)

- *There always exists (at least one) optimal deterministic policy!*
Bellman optimality equation

\[ V^\pi(x) = E \{ r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | x_0 = x; \pi \} \]
\[ = E \{ r_0 | x_0 = x; \pi \} + \gamma E \{ r_1 + \gamma r_2 + \cdots | x_0 = x; \pi \} \]
\[ = R(\pi(x), x) + \gamma \sum_{x'} P(x' | \pi(x), x) E \{ r_1 + \gamma r_2 + \cdots | x_1 = x'; \pi \} \]
\[ = R(\pi(x), x) + \gamma \sum_{x'} P(x' | \pi(x), x) V^\pi(x') \]

• Bellman optimality equation

\[ V^*(x) = \max_a \left[ R(a, x) + \gamma \sum_{x'} P(x' | a, x) V^*(x') \right] \]
\[ \pi^*(x) = \arg\max_a \left[ R(a, x) + \gamma \sum_{x'} P(x' | a, x) V^*(x') \right] \]

(if \( \pi \) would select another action than \( \arg\max_a [\cdot] \), \( \pi \) wouldn’t be optimal: \( \pi' \) which = \( \pi \) everywhere except \( \pi'(x) = \arg\max_a [\cdot] \) would be better)

• this is the \textit{principle of optimality} in the stochastic case
  (related to Viterbi, max-product algorithm)
Dynamic Programming

- Bellman optimality equation

\[ V^*(x) = \max_a \left[ R(a, x) + \gamma \sum_{x'} P(x' | a, x) V^*(x') \right] \]

- Value iteration (initialize \( V_0(x) = 0 \), iterate \( k = 0, 1, .. \))

\[ \forall x : V_{k+1}(x) = \max_a \left[ R(a, x) + \gamma \sum_{x'} P(x' | a, x) V_k(x') \right] \]

- stopping criterion: \( \max_x |V_{k+1}(x) - V_k(x)| \leq \epsilon \)

(see script for proof of convergence)

- once it converged, choose the policy

\[ \pi_k(x) = \arg \max_a \left[ R(a, x) + \gamma \sum_{x'} P(x' | a, x) V_k(x') \right] \]
maze example

• typical example for a value function in navigation

[online demo – or switch to Terran Lane’s lecture...]

8/21
comments

- Bellman’s principle of optimality is the core of the methods

- it refers to the recursive thinking of what makes a path optimal
  - the recursive property of the optimal value function

- related to Viterbi, max-product algorithm
Learning from experience

- Reinforcement Learning problem: model $P(x' | a, x)$ and $P(r | a, x)$ are not known, only exploration is allowed.
Model learning

- trivial on direct discrete representation: use experience data to estimate model

\[ \hat{P}(x'|a, x) \propto \#(x' \leftarrow x|a) \]

- for non-direct representations: Machine Learning methods

- use DP to compute optimal policy for estimated model

- Exploration-Exploitation *is not a Dilemma*
  possible solutions: E$^3$ algorithm, Bayesian RL (see later)
Temporal Difference

- recall Value Iteration

\[ \forall x : V_{k+1}(x) = \max_a \left[ R(a, x) + \gamma \sum_{x'} P(x'|a, x) V_k(x') \right] \]

- Temporal Difference learning (TD): given experience \((x_t a_t r_t x_{t+1})\)

\[ V_{\text{new}}(x_t) = (1 - \alpha)V_{\text{old}}(x_t) + \alpha [r_t + \gamma V_{\text{old}}(x_{t+1})] \]

\[ = V_{\text{old}}(x_t) + \alpha [r_t + \gamma V_{\text{old}}(x_{t+1}) - V_{\text{old}}(x_t)] . \]

... is a stochastic variant of Dynamic Programming

→ one can prove convergence with probability 1 (see Q-learning in script)

- reinforcement:
  - more reward than expected \((r_t > \gamma V_{\text{old}}(x_{t+1}) - V_{\text{old}}(x_t))\)
    → increase \(V(x_t)\)
  - less reward than expected \((r_t < \gamma V_{\text{old}}(x_{t+1}) - V_{\text{old}}(x_t))\)
    → decrease \(V(x_t)\)
Q-learning convergence with prob 1

- Q-learning

\[ Q^\pi(a, x) = \mathbb{E}\{r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid x_0 = x, a_0 = u; \pi \} \]

\[ Q^*(a, x) = R(a, x) + \gamma \sum_{x'} P(x' \mid a, x) \max_{a'} Q^*(a', x') \]

\[ \forall a, x : Q_{k+1}(a, x) = R(a, x) + \gamma \sum_{x'} P(x' \mid a, x) \max_{a'} Q_k(a', x') \]

\[ Q_{\text{new}}(x_t, a_t) = (1 - \alpha)Q_{\text{old}}(x_t, a_t) + \alpha[r_t + \gamma \max_a Q_{\text{old}}(x_{t+1}, a)] \]

- Q-learning is a stochastic approximation of Q-VI:
  Q-VI is deterministic: \[ Q_{k+1} = T(Q_k) \]
  Q-learning is stochastic: \[ Q_{k+1} = (1 - \alpha)Q_k + \alpha[T(Q_k) + \eta_k] \]

\( \eta_k \) is zero mean!
Q-learning impact

• Q-Learning (Watkins, 1988) is the first provably convergent direct adaptive optimal control algorithm

• Great impact on the field of Reinforcement Learning
  – smaller representation than models
  – automatically focuses attention to where it is needed i.e., no sweeps through state space
  – though does not solve the exploration versus exploitation issue
  – epsilon-greedy, optimistic initialization, etc,...
Eligibility traces

- Temporal Difference:

\[ V_{\text{new}}(x_0) = V_{\text{old}}(x_0) + \alpha [r_0 + \gamma V_{\text{old}}(x_1) - V_{\text{old}}(x_0)] \]

- longer reward sequence: \( r_0 \ r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \)

temporal credit assignment, think further backwards, receiving \( r_3 \) also
tells us something about \( V(x_0) \)

\[ V_{\text{new}}(x_0) = V_{\text{old}}(x_0) + \alpha [r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V_{\text{old}}(x_3) - V_{\text{old}}(x_0)] \]

- online implementation: remember where you’ve been recently
  (“eligibility trace”) and update those values as well:

\[ e(x_t) \leftarrow e(x_t) + 1 \]
\[ \forall x : V_{\text{new}}(x) = V_{\text{old}}(x) + \alpha e(x) [r_t + \gamma V_{\text{old}}(x_{t+1}) - V_{\text{old}}(x_t)] \]
\[ \forall x : e(x) \leftarrow \gamma \lambda e(x) \]

- core topic of Sutton & Barto book
  – great improvement
again, Bellman’s principle of optimality is the core of the methods

\(TD(\lambda)\), Q-learning, eligibilities, are all methods to converge to a function obeying the Bellman optimality equation
E³ : Explicit Explore or Exploit

• (John Langford) from observed data construct two MDPs:
  
  (1) $\text{MDP}_{\text{known}}$ includes sufficiently often visited states and executed actions with (rather exact) estimates of $P$ and $R$.  
  (model which captures what you know)
  
  (2) $\text{MDP}_{\text{unknown}} = \text{MDP}_{\text{known}}$ except the reward is 1 for all actions which leave the known states and 0 otherwise.  
  (model which captures optimism of exploration)

• the algorithm:
  
  (1) If last $x$ not in Known: choose the least previously used action
  
  (2) Else:
    
    (a) [seek exploration] If $V_{\text{unknown}} > \epsilon$ then act according to $V_{\text{unknown}}$ until state is unknown (or $t \mod T = 0$) then goto (1)
    
    (b) [exploit] else act according to $V_{\text{known}}$
E³ – Theory

• for any (unknown) MDP:
  – total number of actions and computation time required by E³ are
    \( \text{poly}(|X|, |A|, T^*, \frac{1}{\epsilon}, \ln \frac{1}{\delta}) \)
  – performance guarantee: with probability at least \((1 - \delta)\) exp. return of E³ will exceed \(V^* - \epsilon\)

• details
  – actual return: \(\frac{1}{T} \sum_{t=1}^{T} r_t\)
  – let \(T^*\) denote the (unknown) mixing time of the MDP
  – one key insight: even the optimal policy will take time \(O(T^*)\) to achieve actual return that is near-optimal

• straight-forward & intuitive approach!
  – the exploration-exploitation dilemma is not a dilemma!
  – cf. active learning, information seeking, curiosity, variance analysis
Bayesian Reinforcement Learning

• initially, we don’t know the MDP
  – but based on experience we can estimate the MDP

• parametrize MDP by a parameter $\theta$
  (e.g., direct parametrization: $P(x' \mid a, x) = \theta_{x', ax}$)
  – given experience $x_0a_0 x_1a_1 x_0a_0 \cdots$ we can estimate the posterior

\[
b(\theta) = P(\theta \mid x_{0:t}, a_{0:t})
\]

• given a “posterior belief $b$ about the world”, plan to maximize policy value in this distribution of worlds

\[
V^*(x, b) = \max_a \left[ R(a, x) + \gamma \sum_{x'} \int \theta P(x' \mid a, x; \theta) b(\theta) d\theta \: V^*(x', b') \right]
\]

(see last year’s ICML tutorial, old theory from Operations Research)
comments

- there is no fundamentally unsolved exploration-exploitation dilemma!
  - but an efficiency issue
further topics

- function approximation:
  - Laplacian eigen functions as value function representation (Mahadevan)
  - Gaussian processes (Carl Rasmussen, Yaakov Engel)

- representations:
  - macro (hierarchical) policies, abstractions, options (Precup, Sutton, Singh, Dietterich, Parr)
  - predictive state representations (PSR, Littman, Sutton, Singh)

- partial observability (POMDPs)
we conclude with a demo from Andrew Ng
  – helicopter flight
  – actions: standard remote control
  – reward functions: hand-designed for specific tasks

[rolls] [inverted flight]
appendix: discrete time continuous state control

- same framework as MDPs, different conventional notation:

<table>
<thead>
<tr>
<th>control</th>
<th>MDPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_t$ control</td>
<td>$a_t$ action</td>
</tr>
<tr>
<td>$x_{t+1} = x_t + f(t, x_t, u_t) + \xi_t$ (discrete)</td>
<td>$P(x_{t+1}</td>
</tr>
<tr>
<td>$dx = f(t, x, u) , dt + d\xi$ (continuous system)</td>
<td></td>
</tr>
</tbody>
</table>

| $\phi(x_T)$ final cost | $P(r_t | a_t, x_t)$ reward prob |
| $R(t, x_t, u_t)$ local cost | |
| $C(x_0, u_0:T)$ exp. trajectory cost | $V^\pi(x)$ policy value |
| $J(t, x)$ optimal cost-to-go | $V^*(x)$ optimal value function |

- discrete time stochastic controlled system

\[
x_{t+1} = f(x_t, u_t) + \xi, \quad \xi \sim \mathcal{N}(0, Q)
\]

\[
P(x_{t+1} | u_t, x_t) = \mathcal{N}(x_{t+1} | f(x_t, u_t), Q)
\]

- objective is to minimize the expectation of the cost

\[
C(x_0:T, u_0:T) = \sum_{t=0}^{T} R(t, x_t, u_t).
\]
appendix: discrete time continuous state control

• just as we had in the MDP case, the value function obeys the Bellman optimality equation

\[ J_t(x) = \min_u \left[ R(t, x, u) + \int P(x' \mid u, x) J_{t+1}(x') \right] \]

• 2 types of optimal control problems

open-loop: find control sequence \( u_{1:T}^* \) that minimizes the expected cost

closed-loop: find a control law \( \pi^* : (t, x) \mapsto u_t \) (that exploits the true state observation in each time step and maps it to a feedback control signal) that minimizes the expected cost
appendix: Linear-quadratic-Gaussian (LQG) case

- consider a linear control process with Gaussian noise and quadratic costs,

\[ P(x_t \mid x_{t-1}, u_t) = \mathcal{N}(x_t \mid Ax_{t-1} + Bu_t, Q), \]

\[ C(x_{1:T}, u_{1:T}) = \sum_{t=1}^{T} x_t^T Rx_t + u_t^T Hu_t. \]

- assume we know the exact cost-to-go \( J_t(x) \) at time \( t \) and that it has the form \( J_t(x) = x^T V_t x \). Then,

\[
J_{t-1}(x) = \min_{a} \left[ x^T Rx + u^T Hu + \int_y \mathcal{N}(y \mid Ax + Bu, Q) y^T V_t y \, dx \right]
\]

\[
= \min_{a} \left[ x^T Rx + u^T Hu + (Ax + Bu)^T V_t (Ax + Bu) + \text{tr}(V_t Q) \right]
\]

\[
= \min_{a} \left[ x^T Rx + u^T (H + B^T V_t B)u + 2u^T B^T V_t Ax + x^T AV_t Ax + \text{tr}(V_t Q) \right]
\]

minimization yields

\[
0 = 2(H + B^T V_t B)u^* + 2B^T V_t Ax \quad \Rightarrow \quad u^* = -(H + B^T V_t B)^{-1} B^T V_t Ax
\]

\[
J_{t-1}(x) = x^T V_{t-1} x, \quad V_{t-1} = R + A^T V_t A - A^T V_t B (H + B^T V_t B)^{-1} B^T V_t A
\]

- this is called the Ricatti equation