15-780: Grad AI
Lecture 21: Bayesian learning, MDPs

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman
Admin

- Reminder: project milestone reports due today
- Reminder: HW5 out
Review: numerical integration

- Parallel importance sampling
  - allows $Z\mathcal{R}(x)$ instead of $\mathcal{R}(x)$
  - biased, but asymptotically unbiased
- Sequential sampling (for chains, trees)
- Parallel IS + **resampling** for sequential problems = **particle filter**
Review: MCMC

- Metropolis-Hastings: randomized search procedure for high $R(x)$
- Leads to *stationary distribution* $= R(x)$
- Repeatedly tweak current $x$ to get $x'$
  - If $R(x') \geq R(x)$, move to $x'$
  - If $R(x') << R(x)$, stay at $x$
  - randomize in between

- Requires good one-step proposal $Q(x' | x)$ to get acceptable acceptance rate and mixing rate
Review: Gibbs

- Special case of MH for $X$ divided into blocks
- Proposal $Q$:
  - pick a block $i$ uniformly (or round robin, or any other fair schedule)
  - sample $X_{B(i)} \sim P(X_{B(i)} | X_{\neg B(i)})$
- Acceptance rate $= 100\%$
Review: Learning

\[ P(M | X) = \frac{P(X | M) P(M)}{P(X)} \]

\[ P(M | X, Y) = \frac{P(Y | X, M) P(X | M)}{P(Y | M)} \]

Example: framlings

Version space algorithm: when prior is uniform and likelihood is 0 or 1
Bayesian Learning
'Recall iris example

\[ \Phi \text{ specs} \]

\[ \Phi \text{ params} \]

- \( \mathcal{H} = \) factor graphs of given structure
- Need to specify entries of \( \phi \)s
Factors

<table>
<thead>
<tr>
<th>$\phi_0$</th>
<th>$\phi_1-\phi_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>setosa</td>
<td>$p$</td>
</tr>
<tr>
<td>versicolor</td>
<td>$q$</td>
</tr>
<tr>
<td>virginica</td>
<td>$1-p-q$</td>
</tr>
<tr>
<td></td>
<td>lo</td>
</tr>
<tr>
<td></td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>hi</td>
</tr>
<tr>
<td>set.</td>
<td>$p_i$</td>
</tr>
<tr>
<td></td>
<td>$q_i$</td>
</tr>
<tr>
<td></td>
<td>$1-p_i-q_i$</td>
</tr>
<tr>
<td>vers.</td>
<td>$r_i$</td>
</tr>
<tr>
<td></td>
<td>$s_i$</td>
</tr>
<tr>
<td></td>
<td>$1-r_i-s_i$</td>
</tr>
<tr>
<td>vir.</td>
<td>$u_i$</td>
</tr>
<tr>
<td></td>
<td>$v_i$</td>
</tr>
<tr>
<td></td>
<td>$1-u_i-v_i$</td>
</tr>
</tbody>
</table>
### Continuous factors

<table>
<thead>
<tr>
<th></th>
<th>lo</th>
<th>m</th>
<th>hi</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>$p_l$</td>
<td>$q_l$</td>
<td>$l-p_l-q_l$</td>
</tr>
<tr>
<td>vers</td>
<td>$r_l$</td>
<td>$s_l$</td>
<td>$l-r_l-s_l$</td>
</tr>
<tr>
<td>vir</td>
<td>$u_l$</td>
<td>$v_l$</td>
<td>$l-u_l-v_l$</td>
</tr>
</tbody>
</table>

\[
\Phi_1(\ell, s) = \exp\left(-\left(\ell - \ell_s\right)^2 / 2\sigma^2\right)
\]

parameters $\ell_{\text{set}}, \ell_{\text{vers}}, \ell_{\text{vir}}$; constant $\sigma^2$

- Discretized petal length
- Continuous petal length
Simpler example

<table>
<thead>
<tr>
<th>H</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>( 1-p )</td>
</tr>
</tbody>
</table>

Coin toss
Parametric model class

- $\mathcal{H}$ is a \textit{parametric} model class: each $H$ in $\mathcal{H}$ corresponds to a vector of parameters $\theta = (p)$ or $\theta = (p, q, p_1, q_1, r_1, s_1, \ldots)$

- $H_{\theta}: \mathbf{X} \sim P(\mathbf{X} \mid \theta)$ (or, $Y \sim P(Y \mid \mathbf{X}, \theta)$)

- Contrast to \textit{discrete} $\mathcal{H}$, as in version space

- Could also have \textit{mixed} $\mathcal{H}$: discrete choice among parametric (sub)classes
Continuous prior

- E.g., for coin toss, \( p \sim \text{Beta}(a, b) \):

\[
P(p \mid a, b) = \frac{1}{B(a, b)} p^{a-1} (1 - p)^{b-1}
\]

- Specifying, e.g., \( a = 2, b = 2 \):

\[
P(p) = 6p(1 - p)
\]
Prior for $p$
Coin toss, cont’d

- Joint dist’n of parameter $p$ and data $x_i$:

$$P(p, x) = P(p) \prod_{i} P(x_i | p)$$

$$= 6p(1 - p) \prod_{i} p^{x_i} (1 - p)^{1-x_i}$$
Coin flip posterior

\[
P(p \mid x) = \frac{P(p) \prod_i P(x_i \mid p)}{P(x)}
= \frac{1}{Z} p(1 - p) \prod_i p^{x_i} (1 - p)^{1-x_i}
= \frac{1}{Z} p^{1+\sum_i x_i} (1 - p)^{1+\sum_i (1-x_i)}
= \text{Beta}(2 + \sum_i x_i, 2 + \sum_i (1 - x_i))
\]
Prior for $p$
Posterior after 4 H, 7 T
Posterior after 10 H, 19 T
Predictive distribution

- Posterior is nice, but doesn’t tell us directly what we need to know
- We care more about $P(x_{N+1} \mid x_1, \ldots, x_N)$
- By law of total probability, conditional independence:

$$P(x_{N+1} \mid D) = \int P(x_{N+1}, \theta \mid D) d\theta$$

$$= \int P(x_{N+1} \mid \theta)P(\theta \mid D) d\theta$$
Coin flip example

- After 10 H, 19 T: $p \sim \text{Beta}(12, 21)$
- $E(x_{N+1} \mid p) = p$
- $E(x_{N+1} \mid \theta) = E(p \mid \theta) = a/(a+b) = 12/33$
- So, predict 36.4% chance of H on next flip
Approximate Bayes
Approximate Bayes

- Coin flip example was easy
- In general, computing posterior (or predictive distribution) may be hard
- Solution: use the approximate integration techniques we’ve studied!
Bayes as numerical integration

- Parameters $\theta$, data $\mathbf{D}$
- $P(\theta \mid \mathbf{D}) = P(\mathbf{D} \mid \theta) P(\theta) / P(\mathbf{D})$
- Usually, $P(\theta)$ is simple; so is $\mathcal{Z} P(\mathbf{D} \mid \theta)$
- So, $P(\theta \mid \mathbf{D}) \propto P(\mathbf{D} \mid \theta) P(\theta)$
  - similarly for conditional model: if $\mathbf{X} \perp \theta$,
  - $P(\theta \mid \mathbf{X}, \mathbf{Y}) \propto P(\mathbf{Y} \mid \theta, \mathbf{X}) P(\theta)$
- Perfect for MH
\[ P(y \mid x) = \sigma(ax + b) \]
\[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]
Posterior

\[ P(a, b \mid x_i, y_i) = \]
\[ Z P(a, b) \prod_{i} \sigma(ax_i + b)^{y_i} \sigma(-ax_i - b)^{1-y_i} \]

\[ P(a, b) = N(0, I) \]
Sample from posterior
Predictive distribution

- For each $\theta$ in sample, predict $P(X)$ or $P(Y \mid X)$
- Average predictions over all $\theta$ in sample
Cheaper approximations
Getting cheaper

- Maximum a posteriori (MAP)
- Maximum likelihood (MLE)
- Conditional MLE / MAP

- Instead of true posterior, just use single most probable hypothesis
MAP

\[ \arg \max_{\theta} P(D \mid \theta)P(\theta) \]

- Summarize entire posterior density using the maximum
MLE

\[ \arg \max_{\theta} P(D \mid \theta) \]

- Like MAP, but ignore prior term
  - often prior is overwhelmed if we have enough data
Conditional MLE, MAP

\[
\arg \max_\theta P(y \mid x, \theta)
\]

\[
\arg \max_\theta P(y \mid x, \theta) P(\theta)
\]

- Split \(D = (x, y)\)
- Condition on \(x\), try to explain only \(y\)
Iris example: MAP vs. posterior
Irises: MAP vs. posterior
Too certain

- This behavior of MAP (or MLE) is typical: we are too sure of ourselves
- But, often gets better with more data
- Thm: MAP and MLE are consistent estimates of true $\theta$, if “data per parameter” $\rightarrow \infty$
Sequential Decisions
Markov decision process: influence diagram

- States, actions, initial state $s_1$, (expected) costs $C(s,a) \in [C_{\text{min}}, C_{\text{max}}]$, transitions $T(s' \mid s, a)$
Influence diagrams

- Like a Bayes net, except:
  - diamond nodes are costs/rewards
    - must have no children
  - square nodes are decisions
    - we pick the CPTs (before seeing anything)
    - minimize expected cost
- Circles are ordinary r.v.s as before
Markov decision process: state space diagram

- States, actions, costs $C(s,a) \in [C_{\text{min}}, C_{\text{max}}]$
- Transitions $T(s' \mid s, a)$, initial state $s_1$

Goal: all costs = 0, self-transition 100%
Choosing actions

- Execution trace: \( \tau = (s_1, a_1, c_1, s_2, a_2, c_2, \ldots) \)
  - \( c_1 = C(s_1, a_1), c_2 = C(s_2, a_2), \text{etc.} \)
  - \( s_2 \sim T(s \mid s_1, a_1), s_3 \sim T(s \mid s_2, a_2), \text{etc.} \)

- Policy \( \pi: S \rightarrow A \)
  - or randomized, \( \pi(a \mid s) \)

- Trace from \( \pi: a_1 \sim \pi(a \mid s_1), \text{etc.} \)
  - \( \tau \) is then an r.v. with known distribution
  - we’ll write \( \tau \sim \pi \) (rest of MDP implicit)
Choosing **good** actions

- **Value of a policy:**

  \[
  J^\pi = \frac{1 - \gamma}{\gamma} \mathbb{E} \left[ \sum_t \gamma^t c_t \mid \tau \sim \pi \right]
  \]

- **Objective:**

  \[
  J^* = \min_\pi J^\pi
  \]

  \[
  \pi^* \in \text{arg min}_\pi J^\pi
  \]
Why a discount factor?
Why a discount factor?

- AI: to make the sums finite
Why a discount factor?

- A1: to make the sums finite
- A2: interest rate $1/\gamma - 1$ per period
Why a discount factor?

- A1: to make the sums finite
- A2: interest rate $1/\gamma - 1$ per period
- A3: model mismatch
  - probability $(1-\gamma)$ that something unexpected happens on each step and my plan goes out the window
Recursive expression

\[ J^\pi = \mathbb{E} \left[ \frac{1 - \gamma}{\gamma} \sum_t \gamma^t c_t \mid \tau \sim \pi \right] \]

\[ = \mathbb{E}[J(\tau) \mid \tau \sim \pi] \]

\[ J(\tau) = \frac{1 - \gamma}{\gamma} [\gamma c_1 + \gamma^2 c_2 + \gamma^3 c_3 + \ldots] \]

\[ = (1 - \gamma)c_1 + \gamma \left[ \frac{1 - \gamma}{\gamma} (\gamma c_2 + \gamma^2 c_3 + \ldots) \right] \]

\[ = (1 - \gamma)c_1 + \gamma J(\tau^+) \]

\((1 - \gamma) \times \text{immediate cost} + \gamma \times \text{future cost}\)
Tree search

\( \gamma = 0.5 \)

\( \text{transitions} = 0.5 \)

- Root node = current state
- Alternating levels: action and outcome
  - min and expectation
- Build out tree until goal or until \( \gamma^t \) small enough
Interpreting the result

- Number at each \( \bigcirc \) node: optimal cost if starting from state \( s \) instead of \( s_1 \)
  - call this \( J^*(s) \)—so, \( J^* = J^*(s_1) \)
  - \textit{state-value} function

- Number at each \( \cdot \) node: optimal cost if starting from parent’s \( s \), choosing incoming \( a \)
  - call this \( Q^*(s,a) \)
  - \textit{action-value} function

- Similarly, \( J^\pi(s) \) and \( Q^\pi(s,a) \)
The update equations

For • node

\[ Q^*(s, a) = (1 - \gamma)C(s, a) + \gamma \mathbb{E}[J^*(s') \mid s' \sim T(\cdot \mid s, a)] \]

For ○ node

\[ J^*(s) = \min_a Q^*(s, a) \]

\((1 - \gamma) \times \text{immediate cost} + \gamma \times \text{future cost}\)
Updates for a fixed policy

- For \( \cdot \) node

\[
Q^\pi(s, a) = (1 - \gamma)C(s, a) + \gamma \mathbb{E}[J^\pi(s') \mid s' \sim T(\cdot \mid s, a)]
\]

- For \( \circ \) node

\[
J^\pi(s) = \mathbb{E}[Q^\pi(s, a) \mid a \sim \pi(\cdot \mid s)]
\]

\((1 - \gamma) \times \text{immediate cost} + \gamma \times \text{future cost}\)
Speeding it up

- Can’t do DPLL-style pruning: outcome node depends on all children
- Can do some pruning: e.g., low-probability outcomes when branch is already clearly bad
- Or, use scenarios: subsample outcomes at each expectation node
  - with enough samples, good estimate of value of each expectation
Receding-horizon planning

- Stop building tree at $2k$ levels, evaluate leaf nodes with *heuristic* $h(s)$
  - or at $2k-1$ levels, evaluate with $h(s, a)$

- Minimal guarantees, but often works well in practice

- Can also use adaptive horizon

- Just as in deterministic search, a good heuristic is essential!
Good heuristic

- Good heuristic: $h(s) \approx J^*(s)$ or $h(s, a) \approx Q^*(s,a)$

- If we have $h(s) = J^*(s)$, only need to build first two levels of tree (action and outcome) to choose optimal action at $s_1$

- With $h(s, a) = Q^*(s,a)$, only need to build first (action) level

- Often try to use $h \approx J^\pi$ or $Q^\pi$ for some good $\pi$
Roll-outs

- Want $h(s) \approx J^\pi(s)$
- Starting from $s_1 = s$, sample $a_1 \sim \pi(a \mid s_1)$, set $c_1 = c(s_1,a_1)$, sample $s_2 \sim T(s' \mid s_1,a_1)$
- Repeat until goal (or until $\gamma^t$ small)
- Take $h(s) = \frac{(1-\gamma)}{\gamma} \sum_t \gamma^t c_t$
- Used in \textbf{UCT} (best algorithm for Go)
Dynamic programming

- If there are a small number of states and actions, makes sense to *memoize* tree search
  - compute an entire level of the tree at a time, working from bottom up
  - store only $S \times A$ numbers r.t. $b^d$
DP example: should I stay or should I go?

\[
Q(A, \text{stay}) \quad Q(A, \text{go}) \quad J(A)
\]

\[
\begin{aligned}
\frac{1}{3} &+ \frac{2}{3} \cdot \frac{2}{9} = \frac{19}{27} \\
\frac{1}{3} &+ \frac{2}{3} \cdot \frac{2}{3} = \frac{7}{9}
\end{aligned}
\]

\[
Q^*(A, s) \quad J^*(A)
\]
DP example 2

- each step costs 1
- discount 0.8
DP example 2—iteration 0
DP example 2—iteration 1

![Graphs showing Q(s,left) and Q(s,right) for states 0 to 15.](image)
DP example 2—iteration 3
DP example 2—iteration 4

![Graph showing Q(s, left) and Q(s, right) for different states.](image)
DP example 2—iteration 8

![Graphs showing Q(s, left) and Q(s, right) vs State.](image)
Discussion

- Terminology: backup, sweep, value iteration
- VI makes max error converge linearly to 0 at rate $\gamma$ per sweep
- Works well for up to 1,000,000s of states, as long as we can evaluate min and expectation efficiently (e.g., few actions, sparse outcomes)
  - tricks: replace $J(s)$ by backed up value immediately (not at end of sweep); schedule backups by *priority* = estimate of how much $J(s)$ will change
Curse of dimensionality

- Sadly, 1,000,000s of states don’t necessarily get us very far
- E.g., 10 state variables, each with 10 values: $10^{10}$ states
- See below for ways around the curse