Theoretical Issues in Probabilistic Artificial Intelligence

Michael Kearns

AT&T Labs

Road Map

- Overview of classical logic-based AI
- The move towards probabilistic frameworks
- **Graphical Models/Bayes Nets/Probabilistic Inference**: Representing knowledge as probability distributions
- **Markov Decision Processes/Reinforcement Learning**: Planning under uncertainty

Subfields of “Core” AI

- **Knowledge Representation and Reasoning**:
  - Representations of facts or assertions about the world
  - Rules of inference
- **Planning**:
  - Representations of the effects and applicability of actions
  - Methods for finding sequences of actions achieving goals
- **Learning**
  - Has always favored probabilistic frameworks

All three: expressiveness-tractability trade-off

“Classical” (logic-based) AI

- **Knowledge Representation and Reasoning**:
  - Assert father(bill,ray), father(ray,joe), father(X,Y)&father(Y,Z) → grandfather(X,Z)
  - Query: grandfather(joe,bill)?
  - Develop logics and (tractable) inference algorithms
- **Planning**:
  - Operator with Preconditions: clear(X),clear(Y); Effects: remove clear(X), add on(X,Y)
  - Goal: stack red block on green block on blue block
  - Develop logics and (tractable) planning algorithms

Probabilistic AI

- **Knowledge Representation and Reasoning**:
  - Logical assertions → probability distribution
  - Logical inference → conditional distribution
- **Planning**:
  - Logical operators → Markov decision process
  - Operator sequence → policy

Some Feature of Probabilistic AI

- Unification of reasoning, planning, and learning
- Emphasis on approximation for hard problems
- Increased attention to algorithmic issues
- The actual **results** achieved so far!

Part I: Graphical Models and Probabilistic Inference

Representing Distributions by Directed Graphs

- Joint distribution $P(X_1, \ldots, X_n)$ on boolean variables
- Conditional factorization:

$$\begin{aligned} P(X_1, \ldots, X_n) &= P(X_1)P(X_2|X_1) \cdots P(X_n|X_1, \ldots, X_{n-1}) \\ &= \prod P(X_i|X_1, \ldots, X_{i-1}) \end{aligned}$$

- Hope for simplifications through conditional independences:

$$P(X_5|X_1, \ldots, X_4) = P(X_5|X_3)$$

Example: Burglar Alarm Model

- Variables A(larm), B(urglar), E(arthquake), J(ohn), M(ary)
- Joint distribution $P(A, B, E, J, M)$
- Exploit **causality** to choose ordering
- **Assert** factorization

$$P(A, B, E, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A)$$

- Associated **directed graph**: if factorization contains $P(X|pa(X))$, have directed edges from $pa(X)$ to X
- No directed loops, but may have undirected loops
- Full model = directed graph (factorization) + CPT's

Advantages of Bayes Nets

- Dimensionality reduction: $O(2^n) \to O(2^k n)$ parameters, k = max in-degree ($31 \to 10$ parameters in burglar alarm)
- Separate **causality** (qualitative) from CPT's (quantitative)
- Hidden variables can simplify model
- Graph-theoretic algorithms for natural problems

Caveats About Bayes Nets

- **Order** of decomposition can be crucial
- Generally want to reduce in-degree, undirected loops
- Basic problems still notoriously hard;
  must find special cases of interest

Basic Problems on Bayes Nets

- **Inference:**
  - Set S of instantitated **evidence variables** (e.g., $S = \{X_2 = 0, X_7 = 1\}$)
  - **Query variable(s)** X
  - **Goal**: compute $P(X|S)$
  - Query types: diagnostic, predictive,. . .
- **Learning:**
  - **Parameter estimation**: given directed graph; must learn CPT's from sample data
  - **Structure learning**: learn directed graph (and CPT's) from sample data

Complexity of the Basic Problems

- **Inference:**
  - $\#P$-complete in the worst case;
    many intractable restrictions
  - Interesting algorithms for several special cases
- **Learning:**
  - Efficient parameter estimation from fully observed data,
    good heuristics for partially observable
  - Structure learning: intractable

Subtleties of Conditional Independence: "Explaining Away"

- Two variables that are **independent** with **no** evidence may become **dependent** in the presence of evidence
- Burglar alarm example: B and E are independent, but if we observe $A = 1$ then they are dependent
- If we learn there was an earthquake, less likely to believe there was a burglary
- What determines when X and Y are independent given S?

A Graph-Theoretic Characterization of Independence: d-Separation

Let P be an undirected path between X and Y. Say that P is **blocked** by S if:

- There is a node $Z \in S$ on P with an out-edge along P;
- There is a node $Z \notin S$ on P, with both edges along P directed in, and no descendant of Z is in S.

All paths blocked: d-separation, and

$$P(X, Y|S) = P(X|S)P(Y|S)$$

A Tractable Special Case for Inference: Polytrees

- **Polytree**: no **undirected** cycles
- Query node X, evidence set S, want to compute $P(X = x|S)$
- Let $S(X, Y) \subseteq S$ be the evidence **reachable** (undirected) from X **avoiding** Y
- Algorithm: for all nodes X, Y, if $X \to Y$:
  - X sends to Y: $P(X = x, S(X, Y))$ for each x
  - Y sends to X: $P(S(Y, X)|X = x)$ for each x
- S^+, S^-: evidence “upstream” and “downstream” from query node X

Analysis

- First write $P(X = x|S) = P(X = x, S)/P(S) = \alpha P(X = x, S)$
- By d-separation on X:

$$\begin{aligned} P(X = x, S) &= P(X = x, S^{+})P(S^{-}|X = x, S^{+}) \\ &= P(X = x, S^{+})P(S^{-}|X = x) \end{aligned}$$

- Compute $P(X = x, S^{+})$, $P(S^{-}|X = x)$ from messages to X
- Let $\vec{U}$ be parents of X, $\vec{V}$ be children of X

Analysis Continued. . .

- Computing $P(X = x, S^+)$: marginalize over parents

$$P(X = x, S^+) = \sum_{\vec{u}} P(\vec{U} = \vec{u}, S^+) P(X = x | \vec{U} = \vec{u}, S^+)$$

- $P(X = x | \vec{U} = \vec{u}, S^+) = P(X = x | \vec{U} = \vec{u})$, get from CPT
- $P(\vec{U} = \vec{u}, S^+) = \prod P(U_i = u_i, S(U_i, X))$ by d-separation on X; messages from U_i to X
- $P(S^- | X = x) = \prod P(S(V_i, X) | X = x)$ by d-separation on X
- Messages from children V_i to X

Wrapping Up

- If X has all but message from Y, can write to Y
- Tree fills up from the leaves
- Running time: linear in tree size and CPT size

Generalizations to Sparse Networks

Two basic approaches:

- **Cluster** nodes until a polytree is obtained
- **Instantiate** some nodes to yield a set of polytrees, take weighted average

Run time typically exponential in cluster size or number of instantiated variables.

Approximate Inference in Dense Networks

- Often assume a **parametric form** for CPT's
- Parametric form assures "randomness" or averaging behavior
- Sampling/simulation methods: Gibbs sampling
- Variational methods: rigorous upper and lower bounds

Some Common Parametric CPT's

- Node X, parents $U_1, \ldots, U_n$
- CPT specified by weight vector $\vec{\theta}$
- Look at forms $\Pr[X = 1 | \vec{U} = \vec{u}] = \sigma(\vec{\theta} \cdot \vec{u})$
- **Noisy-OR**: $\sigma(x) = 1 - e^{-x}$
- **Sigmoid**: $\sigma(x) = 1/(1 + e^{x})$

Inference in Two-Layer Noisy-OR Networks

- Input units $U_1, \ldots, U_n$, outputs $X_1, \ldots, X_m$
- CPT's for outputs given by weight vectors $\vec{\theta}^1, \ldots, \vec{\theta}^m$
- Inputs have biases $p_1, \ldots, p_n$, assume all 1/2
- Can reduce general queries to form

$$\Pr[X_1 = 1, \ldots, X_m = 1] = (1/2^n) \sum_{\vec{u}} \left(\prod_i \left(1 - e^{-\vec{\theta}^i \cdot \vec{u}} \right) \right)$$

- Suppose we choose λ_i, $i = 1, \ldots, m$, such that

$$e^{\lambda_i x} \geq 1 - e^{-x}$$

for **all** x

Closed-Form Computation of the Variational Upper Bound

$$\begin{aligned}
(1/2^n) \sum_{\vec{u}} \prod_i \left(e^{\lambda_i \vec{\theta^i} \cdot \vec{u}} \right) &= (1/2^n) \sum_{\vec{u}} \left(e^{\sum_i \lambda_i \vec{\theta^i} \cdot \vec{u}} \right) \\
&= (1/2^n) \sum_{\vec{u}} \left(e^{\sum_i \lambda_i \sum_j \theta^i_j u_j} \right) \\
&= (1/2^n) \sum_{\vec{u}} \left(e^{\sum_j u_j \sum_i \lambda_i \theta^i_j} \right) \\
&= (1/2^n) \sum_{\vec{u}} \left(\prod_j \left(e^{u_j \sum_i \lambda_i \theta^i_j} \right) \right) \\
&= \prod_j \mathbf{E} \left[e^{u_j \sum_i \lambda_i \theta^i_j} \right]
\end{aligned}$$

How Should We Choose the λ_i?

- Basic idea: over the distribution on the weighted sums, integrate an **upper bound** on transfer function
- Single unit: choose λ_i so upper bound approximates transfer function well near $\mu_i = \mathbf{E}[\vec{\theta^i} \cdot \vec{u}]$
- Many units: may do better than approximating near each μ_i
- The λ_i capture (limited) correlations between the X_i
- In practice: gradient descent on $\vec{\lambda}$

Analysis of Variational Methods

- Let P be true probability, $\widehat{P}_U(\vec{\lambda})$, $\widehat{P}_L(\vec{\lambda})$ variational upper and lower bounds
- Want to bound $\widehat{P}_U(\vec{\lambda}) - \widehat{P}_L(\vec{\lambda})$
- Intuition: for "most" input settings $\vec{u}$, all weighted sums are "near" their means

Large Deviation Methods

- Probability that $\vec{\theta^i} \cdot \vec{u}$ exceeds its mean μ_i by more than ϵ_i bounded by $e^{c_i \epsilon_i^2 n}$
- **Conditioned** on this event E_i, $\Pr[X_i = 1 | E_i] \leq \sigma(\mu_i + \epsilon_i)$
- Probability **some** E_i fails bounded by $\sum_i e^{c_i \epsilon_i^2 n}$
- Another parameterized upper bound:

$$\hat{P}_U(\vec{\epsilon}) = \left(1 - \sum_i e^{c_i \epsilon_i^2 n}\right) \prod_i \sigma(\mu_i + \epsilon_i) + \sum_i e^{c_i \epsilon_i^2 n}$$

- Lower bound:

$$\hat{P}_L(\vec{\epsilon}) = \left(1 - \sum_i e^{c_i \epsilon_i^2 n}\right) \prod_i \sigma(\mu_i - \epsilon_i)$$

Bounds for Large, Dense Networks

- Can get bounds of form

$$m/n^2 + \beta^m m\sqrt{\log(n)/n}$$

  for some $\beta < 1$ depending on network

- Larger γ yields larger ϵ_i

- Generalizes to variational methods, arbitrary transfer functions, multilayer networks,...

Further Topics

- Handling "loopy" networks: connections with decoding turbocodes
- Object-oriented Bayesian networks

Part II: Markov Decision Processes, Probabilistic Planning, and Reinforcement Learning

Planning Under Uncertainty: Markov Decision Processes

- **State space** $\{1, \ldots, N\}$ (or infinite)
- **Actions** $a_1, \ldots, a_k$
- **Transition probabilities** P_{ij}^a
- **Rewards** R_i^a (assume deterministic)
- **Return** on reward sequence $R_0, \ldots, R_t$:
  - **Discounted:** $R_0 + \gamma R_1 + \cdots + \gamma^t R_t$,
    $0 < \gamma < 1$; ϵ-horizon time $H_\epsilon \approx (1/(1-\gamma)) \log(1/\epsilon)$
  - **Average:** $(1/(t+1))(R_0 + \cdots + R_t)$
    (finite or infinite horizon)

Assume **full observability** for now.

Basic Problems on MDP's

- **Planning**:
  - **Given** complete MDP as input, compute strategy with optimal expected return
- **Learning**:
  - Only have access to **experience** in the MDP
  - Learn a near-optimal strategy
  - What kind of experience?

Problems and their solutions are often blurred.

Policies and Value Functions

- **Policy**: (randomized) mapping π of states to actions
- **State value function** for π (discounted): expected asymptotic discounted return starting from i following π

$$V^\pi(i) = R_i^{\pi(i)} + \gamma \sum_j P_{ij}^{\pi(i)} V^\pi(j)$$

- **State-action value function**: value of immediately taking action a if we subsequently follow π

$$Q^\pi(i,a) = R_i^a + \gamma \sum_j P_{ij}^a V^\pi(j)$$

- **Optimal** value functions $V^*(i), Q^*(i,a)$

Approaches to Optimal Planning

- Linear programming: action variable for each state
- Policy iteration: being greedy w.r.t. $Q^{\pi}(i, a)$ improves π
- Value iteration

Optimal Planning via Value Iteration

- Begin with initial guess $\hat{Q}^*(i,a)$ for all state-action pairs (i,a); value function defines (greedy) policy
- Iterative updates: for all (i,a)

$$\hat{Q}^*(i,a) \leftarrow R_i^a + \gamma \sum_j P_{ij}^a \max_b \{\hat{Q}^*(j,b)\}$$

- $\hat{Q}^*(i,a) = Q^*(i,a)$ is only fixed point of mapping
- **Contraction property**: after t iterations,

$$\max_{i,a}\{|Q^*(i,a) - \hat{Q}^*(i,a)|\} \leq \gamma^t$$

- (Near) Optimal planning in time polynomial in N; large N?
- Advantages over linear programming

Learning in MDP's

- Continuous experience vs. reset to a start state vs. access to a simulator
- **Credit assignment** problem
- **Exploration-Exploitation** trade-off

An On-Line Version of Value Iteration: Q-Learning

- Again begin with initial guess $\widehat{Q}^*(i,a)$ for all (i,a)
- In response to observation $i \to^a j$:

$$\widehat{Q}^*(i,a) \leftarrow (1-\alpha)\widehat{Q}^*(i,a) + \alpha(R_i^a + \gamma \max_b \{\widehat{Q}^*(j,b)\})$$

- Adjustable **learning rate** α
- Typical choice is $\alpha = \alpha(t) = 1/t$ at observation t
- Note:

$$\mathbf{E}[\gamma \max_b \{\widehat{Q}^*(j,b)\}] = \gamma \sum_j P_{ij}^a \max_b \{\widehat{Q}^*(j,b)\}$$

- Q-Learning can be applied to **any** observations

Indirect Methods for Learning

- Q-Learning **directly** learns a value function
- **Indirect** methods
  - Use observations to learn a **model** $\widehat{P}_{ij}^{a}$
  - Run value iteration on model

Q-Learning vs. Indirect Algorithm

- Both algorithms known to converge to optimal policy **asymptotically** (infinite sampling at every (i, a))
- Number of parameters: $O(N)$ vs. $O(N^2)$
- Sample sizes? Memory?
- Multiple reward functions?

Convergence Rates for Q-Learning and Indirect Algorithm

- After only $O((\log(1/\epsilon)/\epsilon^2)\log(N/\epsilon))$ trials **per state-action pair**, both algorithms will have an ϵ-good policy with probability at least $1-\delta$
- **Sparse sampling**: only $O(\log(N))$ samples per next-state distribution
- Memory $O(N\log(N))$ vs. $O(N^2)$ for indirect algorithm
- Proof appeals to uniform convergence methods on $O(N)$ random variables per iteration, plus contraction property
- Exploration: account for **mixing time** of an **arbitrary** "exploration policy", but ...

Towards Near-Optimal Exploration

- Full learning problem: **choose** actions during training phase
- Discounted: effectively finite-horizon, given by ϵ-horizon time $H_\epsilon = (1/(1-\gamma))\log(1/\epsilon)$
- Undiscounted: **must** depend on mixing time of **optimal policy**
- More refined: **compete** against all policies with mixing time T, in time polynomial in T
- **Anytime** algorithm?

The Explicit Explore or Exploit (E^3) Algorithm

- Assume given mixing time T, optimal expected return V_T
- Learning algorithm:
  - Wander randomly, estimate next-state distributions
  - Let $\hat{M}$ be **known** sub-MDP
  - Offline: compute optimal T-step return in $\hat{M}$
  - If near V_T, execute it!
  - Else appeal to **Explore or Exploit Lemma**
- Key idea: any time we are not gaining V_T, we improve our statistics at an unknown state

Performance Guarantee

For **any** MDP on N states, and **any** T, ϵ, δ, if we run E^3 for poly$(N, T, 1/\epsilon, 1/\delta)$ steps, then with probability at least $1 - \delta$ the total return will exceed $V_T - \epsilon$.

Handling Large or Infinite State Spaces

- Typically have $N = 2^n$ (games) or N infinite (control problems)
- Even explicitly specifying a policy is infeasible
- Cannot run directly on the P_{ij}^a, value iteration doomed
- More realistic: assume we are given a **generative model** for the MDP
- On input (i, a), receive R_i^a and a random j drawn from P_{ij}^a
- How can we use a generative model to plan optimally?

Near-Optimal Planning in Large MDP's via Sparse Sampling

- Given access to a generative model for large MDP
- Instead of outputing a **complete** policy, give algorithm taking (current) state i as input
- Output: a near-optimal action from i
- Algorithm builds **sparse tree** rooted at i to approximate $Q^*(i,a)$ for each action a
- Claim: with a tree of size only $O((1/\epsilon)^{H_\epsilon})$, get ϵ-good approximation

Near-optimal planning with **no** dependence on state space size.

Handling Partial Observability

- Many applications: do not see "full state"
- Example: elevator controller can detect pushed buttons, but Markovian only if we know distribution of waiting passengers
- Example: learning finite-state automata
- Formal model: to MDP P^a_{ij}, R^a_i, POMDP adds **observation distributions** Q_i on observation set for each state i

What Changes?

- Move from policies to **strategies**: optimal action may depend on entire **history**
- Optimal **planning** (given P_{ij}^a, Q_i, R_i^a) intractable
- What's the optimal planning algorithm?

The Belief-State MDP of a POMDP

- Assume known initial distribution P_0 on the N states of given POMDP
- States of belief-state MDP: all possible **distributions** on states of POMDP
- From distribution P, action a with observation o causes transtion to P' according to Bayesian posterior update
- Generalization of value iteration runs on belief-state MDP in time exponential in N

Further Topics

- Learning constrained strategies in POMDP's
- Function approximation in large state spaces