Review

- Bayesian Networks
  - Compact representation for probability distributions
  - Exponential reduction in number of parameters
- Fast probabilistic inference
  - As shown in demo examples
  - Compute $P(X|e)$
- Today
  - Learn BN structure

Bayesian Networks – (Structure) Learning

Machine Learning – CSE446
Carlos Guestrin
University of Washington

May 31, 2013

©Carlos Guestrin 2005-2013
Learning Bayes nets

Data structure
P(X_i | Pa_{X_i})

Learning the CPTs

For each discrete variable X_i

\[ P(s=t | a=t) \overset{MLE}{=} \frac{\text{count}(s=t, a=t)}{\text{count}(a=t)} \]

\[ P(x_i=x_i | pa_{X_i}=w) \overset{MLE}{=} \frac{\text{count}(x_i=x_i, pa_{X_i}=w)}{\text{count}(w)} \]

Small subtlety: \( \text{count}(w) = 0 \) or small ...

MLE: \[ P(X_i = x_i | X_j = x_j) = \frac{\text{Count}(X_i = x_i, X_j = x_j)}{\text{Count}(X_j = x_j)} \]
Information-theoretic interpretation of maximum likelihood 1

Given structure, log likelihood of data:
\[
\log P(\mathcal{D} | \theta, \mathcal{G}) = \log \prod_{j=1}^n P(x_j | \theta_c, \theta) = \sum_{j=1}^n \log P(x_j | \theta_c, \theta)
\]

Information-theoretic interpretation of maximum likelihood 2

Given structure, log likelihood of data:
\[
\log P(\mathcal{D} | \theta, \mathcal{G}) = \sum_{i=1}^m \sum_{j=1}^n \log P(x_i = x^{(i)}_j | \text{Pa}_{x_i} = \omega^{(i)}_j)
\]

\[
P(x_i = x^{(i)}_j | \text{Pa}_{x_i} = \omega^{(i)}_j) = \frac{\text{count}(x_i = x^{(i)}_j, \text{Pa}_{x_i} = \omega^{(i)}_j)}{\text{count}(\text{Pa}_{x_i} = \omega^{(i)}_j)}
\]
Information-theoretic interpretation of maximum likelihood 3

Given structure, log likelihood of data:

$$\log P(D | \theta, G) = m \sum_i \sum_{x_i, Pa_{x_i}, G} \bar{P}(x_i, Pa_{x_i}, G) \log \bar{P}(x_i | Pa_{x_i}, G)$$

Decomposable score

- Log data likelihood
  $$\log \bar{P}(D | \theta, G) = m \sum_i \bar{T}(X_i, Pa_{X_i}, G) - m \sum_i \bar{H}(X_i)$$

- Decomposable score:
  - Decomposes over families in BN (node and its parents)
  - Will lead to significant computational efficiency!!!
  - Score($G : D$) = $\sum_i$ FamScore($X_i | Pa_{x_i} : D$)

Decomposes over families in BN (node and its parents):
How many trees are there?

Nonetheless – Efficient optimal algorithm finds best tree

A → B → C → D

A → B → C

01 02 03

Each var has at most 2 parents

About $2^{O(n \log n)}$ trees

Exhaustive search hopeless

Scoring a tree 1: equivalent trees

$\log P(D | \theta, G) = m \sum_i I(X_i, Pa_{X_i}, G) - m \sum_i H(X_i)$

$\sum_{i=1}^{n} I(X_i, Pa_{X_i}, G)$

Scores:

$I(A, B) + I(B, C)$

$I(A, C)$

$A \rightarrow B \rightarrow C$

$A \rightarrow B \leftarrow C$

Same score

Score

$\sum_{i=1}^{n} I(X_i, Pa_{X_i}, G)$

doesn't depend on $\theta$.

Not a tree

B has 2 parents

I (A, B) = I (B, A)

Symmetric

Mutual info of edges

$I (A, B) + I (C, B)$

$I (A, B) + I (C, B)$
Scoring a tree 2: similar trees

\[ \log P(D | \theta, G) = m \sum_i \hat{I}(X_i, Pa_{X_i}, G) - m \sum_i H(X_i) \]

Chow-Liu tree learning algorithm 1

- For each pair of variables \( X_i, X_j \)
  - Compute empirical distribution:
    \[ \hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{m} \]
  - Compute mutual information:
    \[ \hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \]
- Define a graph
  - Nodes \( X_1, \ldots, X_n \)
  - Edge \( (i,j) \) gets weight \( \hat{I}(X_i, X_j) \)
- Maximum spanning tree: given a graph, fit subset of edges that maximize sum at weights \( \hat{I} \) form a tree

\[ I(A, B) > 0 \]

Scan:
- \( I(A, B) + I(B, C) \)
- ignore edge directions

\( (A, B), (A, C) \)

No edge directions

Scan:
- \( I(A, B) + I(A, C) \)

Causal (in general)

Scan:
- Edge directions in \( B_n \)
- not causal

\[ \sum_{(i,j) \in \text{Tree}} I(x_i, y_j) = \text{Marginal of the data} \]

\[ I(x_i, x_j) = 0 \text{ if } i \neq j \]
Chow-Liu tree learning algorithm 2

\[
\log P(D | \theta, G) = \sum_i \tilde{I}(X_i, \text{Pa}_{X_i, G}) - m \sum_i \tilde{H}(X_i)
\]

- Optimal tree BN
  - Compute maximum weight spanning tree
  - Directions in BN: pick any node as root, breadth-first-search defines directions

Structure learning for general graphs

- In a tree, a node only has one parent

- **Theorem:**
  - The problem of learning a BN structure with at most \( d \) parents is **NP-hard for any (fixed)** \( d > 1 \)

- Most structure learning approaches use heuristics
  - (Quickly) Describe the two simplest heuristic
Learn BN structure using local search

Starting from Chow-Liu tree

Local search, possible moves:
- Add edge
- Delete edge
- Invert edge

Score using BIC

Regularized objective: trade off complexity of model + log likelihood

Learn Graphical Model Structure using LASSO

Graph structure is about selecting parents:
\[ P(x_i | \text{Pa}(i)) \]
\[ \text{Pa}(i) \subseteq \{1, \ldots, x_i, x_{i+1}, \ldots x_n\} \]

If no independence assumptions, then CPTs depend on all parents:
\[ P(H | FASN) \]

With independence assumptions, depend on key variables:
\[ P(H | FASN) \leq P(H | S) \rightarrow \text{looking for sparse solutions} \]

One approach for structure learning, sparse logistic regression!

\[ LR \] for each variable: \[ P(x_i | x_2, x_n) \rightarrow \text{logistic regression} \]

Under certain conditions, LASSO optimal adds a sparse penalty on priors, such that priors for some \( x_2, x_n \) become 0 if they are ignored \( \rightarrow \) edge not there
What you need to know about learning BN structures

- Decomposable scores
  - Maximum likelihood
  - Information theoretic interpretation
- Best tree (Chow-Liu)
- Beyond tree-like models is NP-hard
- Use heuristics, such as:
  - Local search
  - LASSO
Reinforcement Learning

training by feedback

weak feedback / dynamical system
"good states", try to get there

Learning to act

- Reinforcement learning
- An agent
  - Makes sensor observations
  - Must select action
  - Receives rewards
    - positive for "good" states
    - negative for "bad" states

[Ng et al. '05]
Markov Decision Process (MDP) Representation

- State space:
  - Joint state $x$ of entire system

- Action space:
  - Joint action $a = \{a_1, \ldots, a_n\}$ for all agents

- Reward function:
  - Total reward $R(x, a)$

- Transition model:
  - Dynamics of the entire system $P(x' | x, a)$

Discount Factors $\gamma \in [0, 1)$

People in economics and probabilistic decision-making do this all the time.

The “Discounted sum of future rewards” using discount factor $\gamma$ is

\[
(\text{reward now}) + \gamma (\text{reward in 1 time step}) + \gamma^2 (\text{reward in 2 time steps}) + \gamma^3 (\text{reward in 3 time steps}) + \ldots \]

\(\text{(infinite sum)}\)
Define:

- $V_A = \text{Expected discounted future rewards starting in state A}$
- $V_B = \text{Expected discounted future rewards starting in state B}$
- $V_T = \text{Expected discounted future rewards starting in state T}$
- $V_S = \text{Expected discounted future rewards starting in state S}$
- $V_D = \text{Expected discounted future rewards starting in state D}$

How do we compute $V_A$, $V_B$, $V_T$, $V_S$, $V_D$?