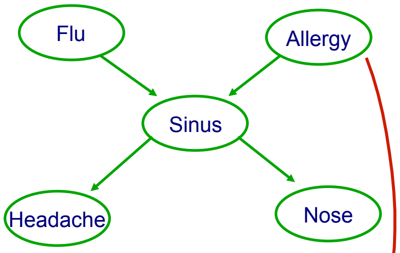
Bayesian Networks – (Structure) Learning

Machine Learning – CSE446
Carlos Guestrin
University of Washington

May 31, 2013

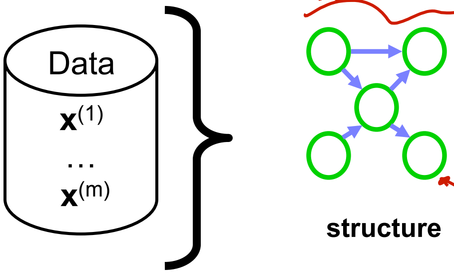
Review

- 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



Learning Bayes nets

learn



structure

+

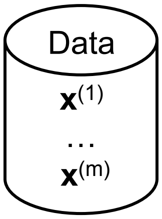
CPTs –
P(X_i| $\mathbf{Pa}_{X_i}$)

parameters

maximizing likelihood of data

MLE $P(D \mid \theta_G, G)$

Learning the CPTs

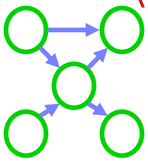


For each discrete variable X_i

$$P(S=t \mid A=t) \overset{MLE}{=} \frac{\text{Count}(S=t, A=t)}{\text{Count}(A=t)}$$

given we know params

$$P(X_i=x_i \mid Pa_{X_i}=u) \overset{MLE}{=} \frac{\text{Count}(X=x_i, Pa_{X_i}=u)}{\text{Count}(u)}$$

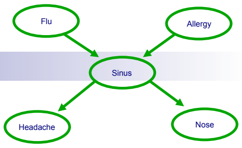


Small subtlety: Count(u)=0, or small …

need Smoothing / AKA regularization / AKA Bayesian learning

MLE: $P(X_i = x_i \mid X_j = x_j) = \dfrac{\text{Count}(X_i = x_i, X_j = x_j)}{\text{Count}(X_j = x_j)}$

Information-theoretic interpretation of maximum likelihood 1



m data points , n variables

- Given structure, log likelihood of data:

$$\log \overset{\text{maximize}_G}{P}(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) \overset{\text{iid}}{=} \log \prod_{j=1}^{m} P(x_1^{(j)} \cdots x_n^{(j)} \mid \theta_G, G)$$

params structure

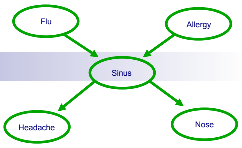
$$= \log \prod_{j=1}^{m} \prod_{i=1}^{n} P(x_i^{(j)} \mid Pa_{x_i,G} = u_i^{(j)})$$

$$= \underbrace{\sum_{j=1}^{m} \sum_{i=1}^{n} \log P(x_i^{(j)} \mid Pa_{x_i,G} = u_i^{(j)})}_{\text{max over } G}$$

$x_i^{(j)} \leftarrow$ data points $(F=t, A=f, S=t \ldots)$

$i \leftarrow$ variables F,A,S,H,N

Information-theoretic interpretation of maximum likelihood 2



- Given structure, log likelihood of data:

$$\log \overset{\max_G}{P}(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = \overset{\text{flip}}{\sum_{j=1}^{m} \sum_{i=1}^{n}} \log P\left(X_i = x_i^{(j)} \mid \mathbf{Pa}_{X_{i,G}} = \overset{u_i^{(j)}}{\cancel{\mathbf{x}^{(j)}[\mathbf{Pa}_{X_i}]}}\right)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} \log P(x_i^{(j)} \mid Pa_{x_i,G} = u_i^{(j)}) \xleftarrow{\text{e.g.}} \sum_{j=1}^{m} \log P(h^{(j)} \mid s^{(j)})$$

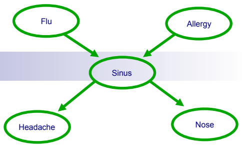
$$= \text{Count}(H=t, S=t) \log P(H=t \mid S=t) + \text{Count}(H=t, S=f) \log P(H=t \mid S=f) + \text{Count}(H=f, S=t) \log P(H=f \mid S=t) + \text{Count}(H=f, S=f) \log P(H=f \mid S=f)$$

$$= \sum_{i=1}^{n} \sum_{x_i} \sum_{u_i} \text{Count}(X_i = x_i, Pa_{x_i,G} = u_i) \log P(x_i \mid Pa_{x_i,G} = u_i)$$

$$= m \sum_{i=1}^{n} \underbrace{\sum_{x_i} \sum_{u_i} P(x_i, Pa_{x_i,G} = u_i) \log P(x_i \mid Pa_{x_i,G} = u_i)}_{-H(X_i \mid Pa_{x_i,G})}$$

$$P(x_i, u_i) \overset{\text{MLE}}{=} \frac{\text{Count}(X_i = x_i, Pa_{x_i,G} = u_i)}{m}$$

Information-theoretic interpretation of maximum likelihood 3



- Given structure, log likelihood of data:

$$\max_{G} \log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{x_i, \mathbf{Pa}_{x_i,\mathcal{G}}} \hat{P}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) \log \hat{P}(x_i \mid \mathbf{Pa}_{x_i,\mathcal{G}})$$

$$\max_{G} = -m \sum_i H(X_i \mid Pa_{X_i,G}) \equiv \min_{G} m \sum_{i=1}^{n} H(X_i \mid Pa_{X_i,G})$$

$$\equiv \max_{G} \underbrace{m \sum_{i=1}^{n} I(X_i, Pa_{X_i,G})}_{\text{information theoretic criteria}} \underbrace{- m \sum_{i=1}^{n} H(X_i)}_{\text{just a constant WRT } G}$$

information theoretic criteria
⇒ max over G mutual info between vars & parents

just a constant WRT G

also measures how dependent vars are

For DTs:

$H(A \mid B)$

$= -\sum_a \sum_b P(a,b) \log P(a \mid b)$

if X_i is highly "correlated" with parents

$H(X_i \mid Pa_{X_i,G})$ is low

⇒ good G

Mutual information

$= I(A,B)$

$= H(A) - H(A \mid B)$

Decomposable score

- Log data likelihood

$$\max_{G} \log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \hat{I}(X_i, \mathbf{Pa}_{X_i,\mathcal{G}}) \underbrace{- m \sum_i \hat{H}(X_i)}_{\text{constant}}$$

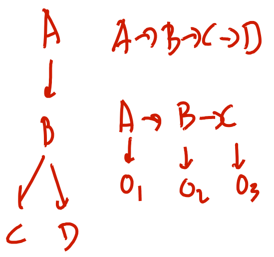
- example of a Decomposable score:
  - ☐ Decomposes over families in BN (node and its parents)
  - ☐ Will lead to significant computational efficiency!!!
  - ☐ Score(G : D) = $\sum_{i}^{n}$ FamScore($X_i|\mathbf{Pa}_{X_i}$: D)

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

but there are many others

How many trees are there?

Nonetheless – Efficient optimal algorithm finds best tree

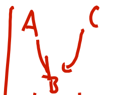


each var has at most 1 parent

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

exhaustive search hopeless

Scoring a tree 1: equivalent trees



not a tree, B has 2 parents

$$\arg\max_{G} \log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \hat{I}(X_i, \mathbf{Pa}_{X_i,\mathcal{G}}) - \underbrace{m \sum_i \hat{H}(X_i)}_{\text{doesn't depend on } G}$$

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

$A \to B \to C$ Same Score $\leftrightarrow$ $A \leftarrow B \leftarrow C$ Same Score $\leftrightarrow$ $A \leftarrow B \to C$

Score: $I(\cancel{A}, \emptyset)^{0} + I(B,A) + I(C,B)$

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

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

$I(A,B) = I(B,A)$ symmetric

pattern is sum of mutual info of edges

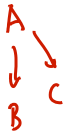
Scoring a tree 2: similar trees

$\hat{I}(A,B) \geq 0$

Argmax G

$$\log \hat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \hat{I}(X_i, \mathbf{Pa}_{X_i,\mathcal{G}}) - m \sum_i \hat{H}(X_i)$$

Score:
$\hat{I}(A,B) + \hat{I}(B,C)$



Score:
$\hat{I}(A,B) + \hat{I}(A,C)$

Shows that edge directions in BNs are not causal (in general)

$$\text{Score}(\text{Tree}) = \sum_{(i,j) \in \text{Tree}} \hat{I}(X_i, X_j)$$

Moral of the story

ignores edge directions

⇒ all Edge directions lead to same score.

⇒ pick arbitrary directions

Chow-Liu tree learning algorithm 1

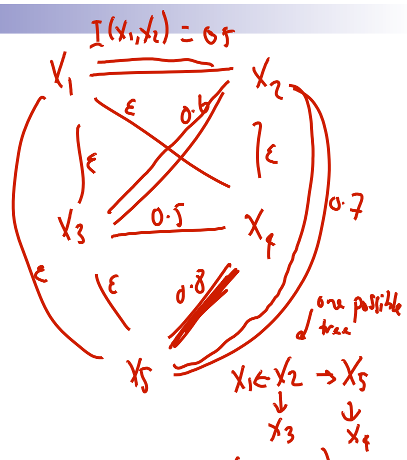
- For each pair of variables X_i,X_j
  - ☐ Compute empirical distribution:

$$\hat{P}(x_i, x_j) \overset{MLE}{=} \frac{\text{Count}(x_i, x_j)}{m}$$

  - ☐ Compute mutual information:

$$\hat{I}(X_i, X_j) = \sum_{x_i,x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

- Define a graph
  - ☐ Nodes $X_1,\ldots,X_n$
  - ☐ Edge (i,j) gets weight $\hat{I}(X_i, X_j)$

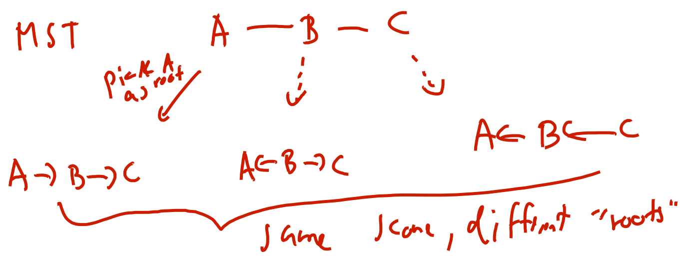


o Maximum Spanning tree:
given a graph, fit subset of edges that maximize sum of weights & form a tree ← complexity about $O(V \log V)$

Chow-Liu tree learning algorithm 2

$$\log \widehat{P}(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \widehat{I}(X_i, \mathbf{Pa}_{X_i,\mathcal{G}}) - m \sum_i \widehat{H}(X_i)$$

- Optimal tree BN
  - ☐ Compute maximum weight spanning tree
  - ☐ Directions in BN: pick any node as root, breadth-first-search defines directions ← all shakings have same score in this case, pick one



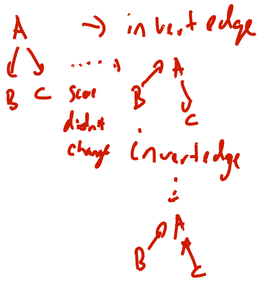
Structure learning for general graphs

- 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

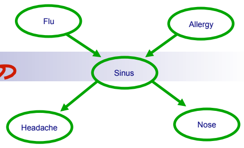
trade off

log likelihood + complexity of structure

e.g. # edges

converge to local optima

Learn Graphical Model Structure using LASSO



- Graph structure is about selecting parents:

$P(X_i | Pa_{X_i})$ defines

$Pa_{X_i} \subseteq \{1, \ldots X_{i-1}, X_{i+1}, \cdots X_n\}$

- If no independence assumptions, then CPTs depend on all parents:

$P(H | FASN)$

- With independence assumptions, depend on key variables:

$P(H|FASN) = P(H|S)$ ← looking for sparse solutions

- One approach for structure learning, sparse logistic regression!

LR for each variable: $P(X_1 | X_2 .. X_n)$ ← logistic regression

add a sparsity penalty on params, such that params for some $X_2 \ldots X_n$ become $\emptyset$ & they are ignored ⇒ edge not there

under certain conditions, learns optimal structure ⇒ not in general, otherwise P = NP

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

Markov Decision Processes (MDPs)

Machine Learning – CSE446
Carlos Guestrin
University of Washington

June 3, 2013

Classification : (Grade 446, GPA) → hire/not hired

supervised learning

direct feedback

unsupervised learning (Grade 446, GPA) → cluster

e.g. clustering

no feedback

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

$x = (x_1 \ldots x_n)$

position player

amount of gold

position of opponent

- State space:
  - Joint state **x** of entire system
- Action space:
  - Joint action **a**= {a_1,…, a_n} for all agents

build castle

walk to wood

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

build castle, negative reward

    - sometimes reward can depend on action

have wood

now

chop wood

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

want Policy: how to act in each state

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 γ" is

(reward now) +

γ (reward in 1 time step) +

γ^2 (reward in 2 time steps) +

γ^3 (reward in 3 time steps) +

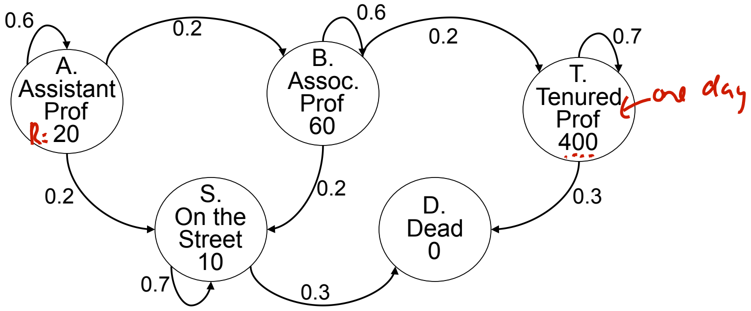
:

: (infinite sum)

maximize discounted future rewards

The Academic Life

Assume Discount Factor $\gamma = 0.9$



Define:

V_A = Expected discounted future rewards starting in state A

V_B = Expected discounted future rewards starting in state B

V_T = " " " " " " " T

V_S = " " " " " " " S

V_D = " " " " " " " D

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