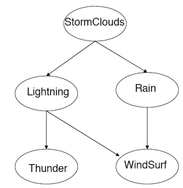
Bayes Nets: Learning Parameters and Structure

Machine Learning 10-701

Anna Goldenberg

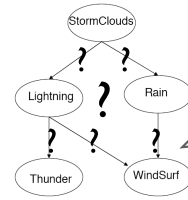
Learning in Bayes Nets

1. Parameter Learning/Estimation: infer Θ from data, given G



Parents	P(W|Pa)	P(~W|Pa)
~L,~R	$\theta_1 =?$	$1-\theta_1$
~L,R	$\theta_2 =?$	$1-\theta_2$
L,~R	$\theta_3 =?$	$1-\theta_3$
L,R	$\theta_4 =?$	$1-\theta_4$

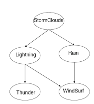
2. Structure Learning: inferring G and Θ from data



Parents	P(W|Pa)	P(~W|Pa)
~L,~R	$\theta_1 =?$	$1-\theta_1$
~L,R	$\theta_2 =?$	$1-\theta_2$
L,~R	$\theta_3 =?$	$1-\theta_3$
L,R	$\theta_4 =?$	$1-\theta_4$

Parameter Learning

Parents	P(W|Pa)	P(~W|Pa)
~L,~R	$\theta_1 =?$	$1-\theta_1$
~L,R	$\theta_2 =?$	$1-\theta_2$
L,~R	$\theta_3 =?$	$1-\theta_3$
L,R	$\theta_4 =?$	$1-\theta_4$



- G is a given DAG over N variables
- Goal: Estimate θ from iid data $D = (x^1, \ldots, x^M)$, where M is the number of records
  - Each record $x^m = \{x_1^m, \ldots, x_N^m\}$
- Complete Observability (no missing values)

Parameter Estimation Outline

- Frequentist Parameter Estimation
  - MLE
    - example of estimation with discrete data
  - MAP
    - estimate for discrete data
- Bayesian Parameter Estimation
  - How it's different from Frequentist

Maximum Likelihood Estimator

- Likelihood (for iid data): $p(D|\theta) = \prod_m \prod_i p(x_i^m | x_{\pi_i}^m, \theta)$
- Log likelihood $l(\theta; D) = \log\, p(D|\theta) = \sum_m \sum_i \log p(x_i^m | x_{\pi_i}^m, \theta)$
- **MLE** $\hat{\theta}_{ML} = \arg\max_\theta l(\theta; D)$
- advantages: has nice statistical properties
- disadvantages: can overfit

Example: MLE for one variable

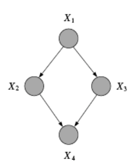
- Variable X ~ Multinomial with K values (K-sided die)
- Observe M rolls: 1, 4, K, 2, ...
- model $p(X = k) = \theta_k$, $\sum_k \theta_k = 1$ (2)

$$l(\theta; D) = \sum_m log \prod I(x^m = k)\theta_k = \sum_k \sum_m I(x^m = k) \log(\theta_k) = \sum_k N_k log(\theta_k) \quad (1)$$

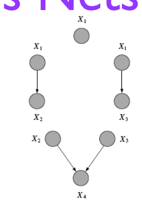
Maximizing (1) subject to constraint (2):

$\hat{\theta}_{k,ML} = \frac{N_k}{M}$ the fraction of times k occurs

Discrete Bayes Nets



<=>



$$p(\mathbf{x}|\theta) = p(\mathbf{x}_1|\theta_1)p(\mathbf{x}_2|\mathbf{x}_1, \theta_2)p(\mathbf{x}_3|\mathbf{x}_1, \theta_3)p(\mathbf{x}_4|\mathbf{x}_2, \mathbf{x}_3, \theta_4)$$

- Assume each CPD is represented as a table

  $\theta_{ijk} \stackrel{\text{def}}{=} P(X_i = j | X_{\pi_i} = k)$
- Loglikelihood: $\ell = \log \prod_m \prod_{ijk} \theta_{ijk}^{N_{ijk}}$
- Parameter Estimator: $\hat{\theta}_{ijk}^{ML} = \frac{N_{ijk}}{\sum_{j'} N_{ij'k}}$

Continuous Variables

Example: Gaussian Variables

One variable: $X \sim N(\mu, \sigma)$

ML estimates: $\hat{\mu}_{ML} = \frac{\sum_m x_m}{M}$

$\hat{\sigma^2}_{ML} = \frac{\sum_m (x_m - \hat{\mu}_{ML})^2}{M}$

Similarly for several Continuous Variables

Another option to estimate parameters: $X_i \sim f(Pa_i, \theta)$

Maximum A Posteriori estimate (MAP)

- MLE is obtained by maximizing loglikelihood
  - sensitive to small sample sizes
- **MAP** comes from maximizing posterior

$$p(\theta|D) \sim p(D|\theta)p(\theta) = likelihood \times prior$$

  - prior acts as a smoothing factor

Example: MAP for Multinomial

Multinomial likelihood: $P(D|\theta) = \prod_m \prod_{ijk} \theta_{ijk}^{N_{ijk}}$

Dirichlet Prior: $P(\theta|\alpha) = \frac{\prod_{ijk} \theta_{ijk}^{(\alpha_{ijk}-1)}}{Z(\alpha)}$

Posterior: $P(\theta|D, \alpha) \propto \prod_{ijk} \theta_{ijk}^{N_{ijk}+\alpha_{ijk}-1}$

MAP $\hat{\theta}_{ijk}^{MAP} = \frac{N_{ijk} + \alpha_{ijk}}{\sum_{j'}(N_{ij'k} + \alpha_{ij'k})}$

α can be thought of as virtual pseudo counts

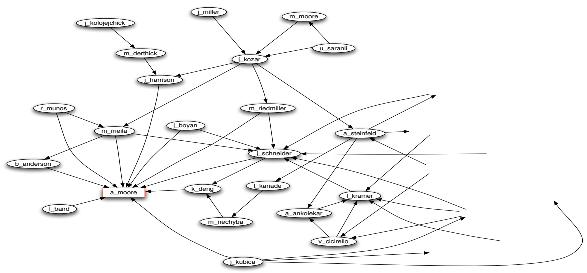
Bayesian vs Frequentist

- Frequentist:
  - θ are unknown **constants**
  - MLE is a very common frequentist estimator
- Bayesian
  - unknown θ are **random variables**
  - estimates differ based on a prior

Questions on Parameter Learning?

What if G is not given?

- When?
  - Scientific discovery (protein networks, data mining)
  - Need a good model for compression, prediction...



Structural Learning

- Constraint Based
  - Test independencies
  - Add edges according to the tests
- Search and Score
  - Define a selection criterion that measures goodness of a model
  - Search in the space of all models (or orders)
- Mix models (recent)
  - Test for almost all independencies
  - Search and score according to possible

Constraint Based Learning

- Define Conditional Independence Test *Ind*($X_i;X_j|S$)
  - e.g. $\chi^2 : \sum_{x_i,x_j} \frac{(O_{x_i,x_j|s} - E_{x_i,x_j|s})^2}{E_{x_i,x_j|s}}$,
    G^2, conditional entropy, etc.
  - if *Ind*($X_i;X_j|S$)<p, *then independence*
  - Choose p with care!
- Construct model consistent with the set of independencies

Constraint Based Learning

- Cons:
  - Independence tests are less reliable on small samples
  - One incorrect independence test might propagate far (not robust to noise)
- Pros:
  - More global decisions => doesn't get stuck in local minima as much
  - Works well on sparse nets (small markov blankets, sufficient data)

Score Based Search Outline

- Select the highest scoring model!
- What should the score be?
- Specialized structures (trees, TANs)
- Selection operators - how to navigate the space of models?

Theorem: maximizing Bayesian Score for $d \geq 2$ (not a tree) is NP-hard (Chickering, 2002)

Maximum likelihood in Information Theoretic terms

$$\log P(D|\theta_G, G) = M \sum_i \hat{I}(X_i|Pa_{X_i}) - M \sum_i \hat{H}(X_i)$$

- The entropy does not depend on the current model
- Thus, it's enough to maximize mutual information!
- General case:
  - Same as constraint search!
- Special case (trees):
  - have to consider only all pairs (tree => only one parent): $O(N^2)$

Chow Liu tree algorithm

- Compute empirical distribution:

$$\hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{M}$$

- Mutual Information:

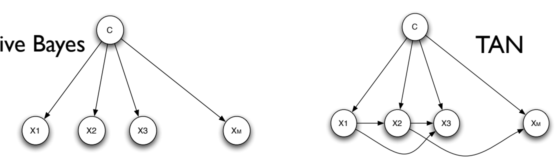
$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

- Set $\hat{I}(X_i, X_j)$ as weight per edge between X_i *and* X_j
- Find *Optimal* tree BN by getting the maximum spanning tree
  for direction: pick a random node as root
  direct in BFS order

Tree Augmented Naive Bayes

TAN (Friedman et al, 1997) is an extension of Chow Liu

Naive Bayes



TAN

TAN: $\hat{I}(X_i, X_j \mid C) = \sum_{c, x_i, x_j} \hat{P}(c, x_i, x_j) \log \frac{\hat{P}(x_i, x_j \mid c)}{\hat{P}(x_i \mid c)\hat{P}(x_j \mid c)}$

Score(TAN): $\sum_i \hat{I}(X_i, C) + \sum_j \hat{I}(X_j, \{Pa_{X_j}, C\})$

MI Problem

- Doesn't penalize complexity: $I(A,B) \leq I(A,\{B,C\})$
- Adding a parent always increases the score!
- Model will overfit, since the completely connected graph would be favored

Penalized Likelihood Score

- BIC (Bayesian Information Criterion)
$$logP(D) \sim logP(D|\hat{\theta}_{ML}) - \frac{d}{2}\log(N)$$
, where d is the number of free parameters
- AIC (Akaike Information Criterion)
$$logP(D) \sim logP(D|\hat{\theta}_{ML}) - d$$
- BIC penalizes complexity more than AIC

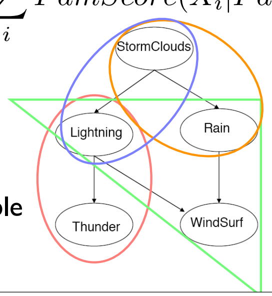
Minimum Description Length

- Total number of bits needed to describe data is $-log_2 P(x)$
- Instead - send the model and then residuals:

  $-L(D,H) = -logP(H) - log\,P(D|H) = -log\,P(H|D) + const$
- The best is the one with the shortest message!

What should the score be?

- Consistent : for all G' I-equivalent to the true G and all G^* not equivalent to G
  $Score(G)=Score(G')$ and $Score(G^*)<Score(G')$
- Decomposable : can be locally computed (for efficiency)
$$Score(G;D) = \sum_i FamScore(X_i|Pa_{X_i};D)$$



Example: BIC and AIC are consistent and decomposable

Bayesian Scoring
Parameter Prior

- Parameter Prior - important for small datasets!

- Dirichlet Parameters (from a few slides before)
- For each possible family define a prior distribution
- Can encode it as a Bayes Net
- (Usually Independent - product of marginals)

Bayesian Scoring
Parameter Prior

- Bayes Dirichlet equivalent scoring (BDe) :

$$\alpha_{X_i|Pa_{X_i}} = MP'(X_i, Pa(X_i))$$

- Is consistent (and decomposable)

  Theorem: If P(G) assigns the same prior to I-equivalent structures and Parameter prior is Dirichlet then Bayesian score satisfies score equivalence, if and only if prior is of BDe form!

The BDeu (uniform) prior is $P'(X_i, X_{\pi_i}) = \frac{1}{|X_i||X_{\pi_i}|}$

Bayesian Scoring
Structure Prior

- Structure Prior - should satisfy prior modularity
- Parameter Modularity: if X has the same set of parents in two different structures, then parameters should be the same.

- Typically set to uniform.
- Can be a function of prior counts: $\frac{1}{\alpha + 1}$

Structure search algorithms

- Order in known
- Order is unknown
  - Search in the space of orderings
  - Search in the space of DAGs
  - Search in the space of equivalence classes

Order is known

- Suppose the total ordering is $X_1 \prec X_2 \ldots \prec X_n$
- Then for each node X_i can find an optimal set of parents in $Pa_i \subseteq \{X_1, \ldots, X_{i-1}\}$
- Choice of parents for X_j doesn't depend on previous X_i
- Need to search among all $\binom{i-1}{d}$ choices (where d is the maximum number of parents) for the highest local score
- Greedy search with known order, aka K2 algorithm is $O(d\binom{n}{d})$

Order is unknown
Search space of orderings

- Select an order according to some heuristic
- Use K2 to learn a BN corresponding to the ordering and score it
- Maybe do multiple restarts
- Most recent research: Tessier and Koller (2005)

Order is unknown
Search space of DAGs

- Typical search operators
  - Add an edge
  - Remove an edge
  - Reverse an edge
- At most $O(n^2)$ steps to get from any graph to any graph
- Moves are reversible
- Simplest search is Greedy Hillclimbing
- Move to proposed new graph if it satisfies constraints

Exploiting Decomposable Score

- If the operator for edge (X,Y) is valid, then we need only to look at the families of X and Y
- e.g. for addition operator *o*

$$\delta_G(o) = \text{FamScore}(Y, Pa(Y,G) \cup X | D) - \text{FamScore}(Y, Pa(Y,G) | D)$$

Evaluating costs of moves

- Total *O(N^2)* operators
- For each operator need to check for acyclicity *O(e)*
  For local moves check acyclicity in amortized O(1) using ancestor matrix
- If new graph is acyclic, need to score it (amortized) O(M)
- K steps to convergence
- Total time O(K *N^2 M)*
- For large M can use AD Trees to compute counts in sub linear time

Suboptimality

- Hillclimbing might get stuck in local maxima
- Local maxima are common because of equivalent classes
- Solutions
  - Random restarts
  - TABU: do not undo up to L latest steps
  - Data perturbation
  - Simulated Annealing (slow!)

Other operators

Optimal Reinsertion (Moore and Wong, 2003)

- Start with an arbitrary DAG
- At every step sever all the edges of a given node
- Reinsert it optimally
  (find best set of parents and children)
- Random restart if necessary

Pros: works much faster and is less prone to get stuck in local minima

Searching in space of equivalent classes (GES)

- Pros: Space of equivalent classes is smaller
- Cons: Operators are more complicated
  Harder to implement
- Empirically shown to have outperformed Greedy Hillclimbing
- Proved to find an optimal BN as $M \rightarrow \infty$

Constraint+Score Algorithms

Tsamardinos et al, 2005

- ➡ Find edges via independence tests
- ➡ Find final structures from the pool of edges using hill-climbing

Claims to be faster than most of the algorithms described above!!!

Current problems with Structural Search

1. Scalability
2. Scalability
3. Scalability
4. Assumption that data samples are iid

Note: there are special purpose algorithms that scale...

Questions on Structural Learning?