Probabilistic
Graphical
Models

Learning

BN Structure

Structure Learning

Why Structure Learning

- To learn model for new queries, when domain expertise is not perfect

- For structure discovery, when inferring network structure is goal in itself

Importance of Accurate Structure

G^* perfect map for P^*

Missing an arc

Adding an arc

- Incorrect independencies
- Correct distribution P* cannot be learned
- But could generalize better

- Spurious dependencies
- Can correctly learn P*
- Increases # of parameters
- Worse generalization

Daphne Koller

Score-Based Learning

optimization

Define scoring function that evaluates how well a structure matches the data

Search for a structure that maximizes the score

Daphne Koller

Probabilistic
Graphical
Models

Learning

BN Structurds

Likelihood Structure Score

Likelihood Score

- Find (G,θ) that maximize the likelihood

$$\text{score}_L(\mathcal{G} \ : \ \mathcal{D}) = \ell((\hat{\boldsymbol{\theta}}, \mathcal{G}) : \mathcal{D})$$

$\hat{\theta}$ = MLE of params. given G and D

Example

$\mathcal{G}_0$ X Y

$\mathcal{G}_1$ X → Y

$$\text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_m (\log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m]})$$

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) = \sum_m (\log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m]|x[m]})$$

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_m (\log \hat{\theta}_{y[m]|x[m]} - \log \hat{\theta}_{y[m]})$$

$$= \sum_{x,y} M[x,y] \log \hat{\theta}_{y|x} - \sum_y M[y] \log \hat{\theta}_y$$

$\hat{P}$ = empirical distribution

$M[x,y] = M\hat{P}[x,y]$

$\sum_x \hat{P}(x,y) = \hat{P}(y)$

$$= M \sum_{x,y} \hat{P}(x,y) \log \hat{P}(y \mid x) - M \sum_y \hat{P}(y) \log \hat{P}(y)$$

$= \frac{\hat{P}(x,y)}{\hat{P}(x)}$

$$= M \left(\sum_{x,y} \hat{P}(x,y) \log \hat{P}(y \mid x) - \sum_{x,y} \hat{P}(x,y) \log \hat{P}(y) \right)$$

$$= M \left(\sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)} \right) = M \cdot \mathbf{I}_{\hat{P}}(X;Y)$$

mutual information

Daphne Koller

General Decomposition

- The Likelihood score decomposes as:

mutual information

independent of G

$$\underbrace{\mathrm{score}_L(\mathcal{G} \ : \ \mathcal{D})} = \underbrace{M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}^G_{X_i})} - \underbrace{M \sum_{i} \boldsymbol{H}_{\hat{P}}(X_i)}$$

Score is higher if X_i is correlated with parents

$$\mathbf{I}_P(\boldsymbol{X}; \boldsymbol{Y}) = \sum_{\boldsymbol{x},\boldsymbol{y}} P(\boldsymbol{x}, \boldsymbol{y}) \log \frac{P(\boldsymbol{x}, \boldsymbol{y})}{P(\boldsymbol{x})P(\boldsymbol{y})}$$

$$\boldsymbol{H}_P(\boldsymbol{X}) = -\sum_{\boldsymbol{x}} P(\boldsymbol{x}) \log P(\boldsymbol{x})$$

Limitations of Likelihood Score

$\mathcal{G}_0$ X Y

$\mathcal{G}_1$ X → Y

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = M\mathbf{I}_{\hat{P}}(X;Y)$$

- Mutual information is always ≥ 0
- Equals 0 iff X, Y are independent
  - In empirical distribution $\hat{P}$ $I_{\hat{P}}(X;Y) > 0$ almost always
- Adding edges can't hurt, and almost always helps
- Score maximized for fully connected network

Avoiding Overfitting

- Restricting the hypothesis space
  - restrict # of parents or # of parameters
- Scores that penalize complexity:
  - Explicitly
  - Bayesian score averages over all possible parameter values

Summary

- Likelihood score computes log-likelihood of D relative to G, using MLE parameters for G
  - Parameters optimized for D
- Nice information-theoretic interpretation in terms of (in)dependencies in G
- Guaranteed to overfit the training data (if we don't impose constraints)

Probabilistic Graphical Models

Learning

BN Structure

BIC Score and Asymptotic Consistency

Penalizing Complexity

Bayesian information criterion

$$\text{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\text{Dim}[\mathcal{G}]$$

training instances

$\text{Score}_L(G : D)$

independent params

- Tradeoff between fit to data and model complexity

(MDL criterion)

← minimum description length

Asymptotic Behavior

$$\ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]$$

$$\boxed{M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{G})} - \underbrace{M \sum_{i} \boldsymbol{H}_{\hat{P}}(X_i)}_{\text{independent of } \mathcal{G}} - \boxed{\frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]}$$

- Mutual information grows linearly with M while complexity grows logarithmically with M
  - As M grows, more emphasis is given to fit to data

$\hat{P} \to P^*$ G^*

Consistency

$$M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}^G_{X_i}) - M \sum_i \boldsymbol{H}_{\hat{P}}(X_i) - \frac{\log M}{2} \mathrm{Dim}[\mathcal{G}]$$

$\sim I_P(X_i; Pa^G_{X_i})$

- As $M \to \infty$, the true structure G^* (or any I-equivalent structure) maximizes the score
  - Asymptotically, spurious edges will not contribute to likelihood and will be penalized
  - Required edges will be added due to linear growth of likelihood term compared to logarithmic growth of model complexity

Summary

- BIC score explicitly penalizes model complexity (# of independent parameters)
  - Its negation often called MDL
- BIC is asymptotically consistent:
  - If data generated by G^*, networks I-equivalent to G^* will have highest score as M grows to ∞

Probabilistic Graphical Models

Learning

BN Structure

Bayesian Score

Bayesian Score

Marginal likelihood

Prior over structures

$$P(\mathcal{G} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathcal{G}) P(\mathcal{G})}{P(\mathcal{D})}$$

independent of $\mathcal{G}$

Marginal probability of Data

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

Daphne Koller

Marginal Likelihood of Data Given G

$$\text{score}_B(\mathcal{G} \ : \ \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

Likelihood

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

Prior over parameters

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}}) P(\boldsymbol{\theta}_{\mathcal{G}} \mid \mathcal{G}) d\boldsymbol{\theta}_{\mathcal{G}}$$

Marginal Likelihood Intuition

$$P(\mathcal{D} \mid \mathcal{G}) = P(\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M] \mid \mathcal{G})$$

$$P(\boldsymbol{x}[1] \mid \mathcal{G})$$
$$P(\boldsymbol{x}[2] \mid \boldsymbol{x}[1], \mathcal{G})$$
$$\ldots$$
$$P(\boldsymbol{x}[M] \mid \boldsymbol{x}[1], \ldots, \boldsymbol{x}[M-1], \mathcal{G})$$

$\hat{\theta}_g$ depends on all of D

Marginal Likelihood: BayesNets

variables

prior

suff stats

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i \left(\prod_{\boldsymbol{u}_i \in Val(\mathbf{Pa}^{\mathcal{G}}_{X_i})} \boxed{\frac{\Gamma(\alpha_{X_i|\boldsymbol{u}_i})}{\Gamma(\alpha_{X_i|\boldsymbol{u}_i} + M[\boldsymbol{u}_i])}} \prod_{x_i^j \in Val(X_i)} \left[\frac{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i} + M[x_i^j, \boldsymbol{u}_i])}{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i})} \right] \right)$$

suff stats

prior

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$$

$$\Gamma(x) = x \cdot \Gamma(x-1)$$

Daphne Koller

Marginal Likelihood Decomposition

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i \left(\prod_{\boldsymbol{u}_i \in Val(\mathbf{Pa}^{\mathcal{G}}_{X_i})} \frac{\Gamma(\alpha_{X_i|\boldsymbol{u}_i})}{\Gamma(\alpha_{X_i|\boldsymbol{u}_i} + M[\boldsymbol{u}_i])} \prod_{x_i^j \in Val(X_i)} \left[\frac{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i} + M[x_i^j, \boldsymbol{u}_i])}{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i})} \right] \right),$$

$$\log P(\mathcal{D} \mid \mathcal{G}) = \sum_i \mathrm{FamScore}_B(X_i \mid \mathbf{Pa}^{\mathcal{G}}_{X_i} \ : \ \mathcal{D})$$

Daphne Koller

Structure Priors

$$\text{score}_B(\mathcal{G} \ : \ \mathcal{D}) = \underline{\log P(\mathcal{D} \mid \mathcal{G})} + \underline{\log P(\mathcal{G})}$$

- Structure prior P(G)
  - Uniform prior: P(G) $\propto$ constant
  - Prior penalizing # of edges: P(G) $\propto$ $c^{|G|}$ (0<c<1)
  - Prior penalizing # of parameters
- Normalizing constant across networks is similar and can thus be ignored P(G)

Daphne Koller

Parameter Priors

- Parameter prior $P(\theta|G)$ is usually the BDe prior
  - α: equivalent sample size
  - B_0: network representing prior probability of events
  - Set $\alpha(x_i, pa_i^G) = \alpha P(x_i, pa_i^G | B_0)$
    - Note: pa_i^G are not the same as parents of X_i in B_0
- A single network provides priors for all candidate networks
- Unique prior with the property that I-equivalent networks have the same Bayesian score

Daphne Koller

BDe and BIC

- As $M \to \infty$, a network G with Dirichlet priors satisfies

#instances

#independent parameters

$$\log P(\mathcal{D} \mid \mathcal{G}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}] + O(1)$$

likelihood score

log likelihood of $\mathcal{D}$ given MLE $\hat{\theta}_G$

constant relative to m — doesn't grow with m

$$\underbrace{\mathrm{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]}_{\text{as } M\to\infty \text{ score is consistent}}$$

Summary

- Bayesian score averages over parameters to avoid overfitting
- Most often instantiated as BDe
  - BDe requires assessing prior network
  - Can naturally incorporate prior knowledge
  - I-equivalent networks have same score
- Bayesian score
  - Asymptotically equivalent to BIC (as $m \to \infty$)
  - Asymptotically consistent learns correct network as $m \to \infty$
  - But for small M, BIC tends to underfit

Probabilistic Graphical Models

Learning

BN Structure

Structure Learning In Trees

Score-Based Learning

Define scoring function that evaluates how well a structure matches the data

Search for a structure that maximizes the score

Daphne Koller

Optimization Problem

Input:

- Training data
- Scoring function (including priors, if needed)
- Set of possible structures

Output: A network that maximizes the score

Key Property: Decomposability

$$\text{score}(\mathcal{G} \ : \ \mathcal{D}) = \sum_i \text{score}(X_i \mid \mathbf{Pa}_{X_i}^{\mathcal{G}} \ : \ \mathcal{D})$$

Daphne Koller

Learning Trees/Forests

- Forests
  - At most one parent per variable

- Why trees?
  - Elegant math
  - Efficient optimization
  - Sparse parameterization ⟹ overfit less

    M is small relative to n

Learning Forests

- $p(i)$ = parent of X_i, or 0 if X_i has no parent

$$\text{score}(\mathcal{G} : \mathcal{D}) = \sum_i \text{score}(X_i \mid \mathbf{Pa}^{\mathcal{G}}_{X_i} : \mathcal{D})$$

$$= \underbrace{\sum_{i:p(i)>0} \text{score}(X_i \mid X_{p(i)} : \mathcal{D})}_{\text{have parent}} + \underbrace{\sum_{i:p(i)=0} \text{score}(X_i : \mathcal{D})}_{\text{no parent}}$$

same for all trees

$$= \sum_{i:p(i)>0} \left(\text{score}(X_i \mid X_{p(i)} : \mathcal{D}) - \text{score}(X_i : \mathcal{D})\right) + \sum_{i=1}^{n} \text{score}(X_i : \mathcal{D})$$

$p(i) \to i$

Improvement over "empty" network

Score of "empty" network

- Score = sum of edge scores + constant

Daphne Koller

Learning Forests I

- Set $w(i \rightarrow j) = Score(X_j \mid X_i) - Score(X_j)$
- For likelihood score, $w(i \rightarrow j) = \overset{\text{mutual information}}{M\,I_{\hat{P}}(X_i; X_j)}$, and all edge weights are nonnegative
  - ⇨ Optimal structure is always a tree
- For BIC or BDe, weights can be negative
  - ⇨ Optimal structure might be a forest

Daphne Koller

Learning Forests II

- A score satisfies score equivalence if I-equivalent structures have the same score
  - Such scores include likelihood, BIC, and BDe

- For such a score, we can show $w(i \rightarrow j) = w(j \rightarrow i)$, and use an undirected graph

Learning Forests III
(for score-equivalent scores)

- Define undirected graph with nodes {1,…,n}
- Set $w(i,j) = \max[Score(X_j \mid X_i) - Score(X_j), 0]$
- Find forest with maximal weight
  - Standard algorithms for max-weight spanning trees (e.g., Prim's or Kruskal's) in $O(n^2)$ time
  - Remove all edges of weight 0 to produce a forest

Daphne Koller

Learning Forests: Example

Tree learned from data of Alarm network

Correct edges

Spurious edges

- Not every edge in tree is in the original network
- Inferred edges are undirected – can't determine direction

Daphne Koller

Summary

- Structure learning is an optimization over the combinatorial space of graph structures
- Decomposability ⇨ network score is a sum of terms for different families
- Optimal tree-structured network can be found using standard MST algorithms
- Computation takes quadratic time

Probabilistic Graphical Models

Learning

BN Structure

General Graphs: Search

Optimization Problem

Input:

- Training data
- Scoring function
- Set of possible structures

Output: A network that maximizes the score

Beyond Trees

- Problem is not obvious for general networks
  - Example: Allowing two parents, greedy algorithm is no longer guaranteed to find the optimal network

- Theorem:
  - Finding maximal scoring network structure with at most *k* parents for each variable is NP-hard for *k>1*

Heuristic Search

Daphne Koller

Heuristic Search

- Search operators:
  - local steps: edge addition, deletion, reversal
  - global steps

- Search techniques:
  - Greedy hill-climbing
  - Best first search
  - Simulated Annealing
  - ...

Search: Greedy Hill Climbing

- Start with a given network
  - empty network
  - best tree
  - a random network
  - prior knowledge
- At each iteration
  - Consider score for all possible changes
  - Apply change that most improves the score greedy
- Stop when no modification improves score local max.

Greedy Hill Climbing Pitfalls

- Greedy hill-climbing can get stuck in:
  - Local maxima

  - Plateaux

    - Typically because equivalent networks are often neighbors in the search space

Why Edge Reversal

G^*

local maximum

A Pretty Good, Simple Algorithm

- Greedy hill-climbing, augmented with:
- Random restarts:
  - When we get stuck, take some number of random steps and then start climbing again
- Tabu list:

  → add (A→B)
  del (C→B)

  del (A→B)
  - Keep a list of K steps most recently taken
  - Search cannot reverse any of these steps

Example: ICU-Alarm

Daphne Koller

JamBayes

Horvitz, Apacible, Sarin, & Liao, UAI 2005

Predicting Surprises

Horvitz, Apacible, Sarin, & Liao, UAI 2005

Learned Model

Horvitz, Apacible, Sarin, & Liao, UAI 2005

Daphne Koller

Influences in Learned Model

Horvitz, Apacible, Sarin, & Liao, UAI 2005

This figure may be used for non-commercial and classroom purposes only.
Any other uses require the prior written permission from AAAS

Biological Network Reconstruction

Phospho-Proteins
Phospho-Lipids
Perturbed in data

Known	———	15/17
Supported	———	2/17
Reversed	———	1
Missed	- - -	3

Subsequently validated in wetlab

From "Causal protein-signaling networks derived from multiparameter single-cell data"
Sachs et al., *Science* 308:523, 2005. **Reprinted with permission from AAAS.**

Daphne Koller

Summary

- Useful for building better predictive models:
  - when domain experts don't know the structure
  - for knowledge discovery
- Finding highest-scoring structure is NP-hard
- Typically solved using simple heuristic search
  - local steps: edge addition, deletion, reversal
  - hill-climbing with tabu lists and random restarts
- But there are better algorithms

Probabilistic Graphical Models

Learning

BN Structure

General Graphs: Decomposability

Heuristic Search

Daphne Koller

Naïve Computational Analysis

- Operators per search step: delete, reverse — absent – add
  n(n-1) possible edges — present — 2 — 1
  $O(n^2)$
- Cost per network evaluation:
  - Components in score — n components
  - Compute sufficient statistics — O(M)
  - Acyclicity check — O(m) ← # edges

  O(n.M)
- Total: $O(\underline{n^2}\ (\underline{Mn} + \underline{m}))$ per search step

Daphne Koller

Exploiting Decomposability

Δscore(D) =
Score(D | {B,C})
- Score(D | {C})

score = Score(A | {}) + Score(B | {}) + Score(C | {A,B}) + Score(D | {C})

score = Score(A | {}) + Score(B | {}) + Score(C | {A,B}) + Score(D | {B,C})

O(n) savings

Daphne Koller

Exploiting Decomposability

Daphne Koller

Exploiting Decomposability

To recompute scores,
only need to re-score families
that changed in the last move

1 family addition
deletion
2 in reversal
O(n)

Daphne Koller

Computational Cost

- Cost per move
  - Compute O(n) delta-scores damaged by move
  - Each one takes O(M) time

  $O(nM)$
- Keep priority queue of operators sorted by delta-score – O(n log n)

  $O(nM + n\log n)$

More Computational Efficiency

- Reuse and adapt previously computed sufficient statistics
- Restrict in advance the set of operators considered in the search

A, B, C

$M[A,B,C]$

$M[A,B] = \sum_c M[A,B,C]$

Summary

- Even heuristic structure search can get expensive for large n
- Can exploit decomposability to get orders of magnitude reduction in cost
- Other tricks are also used for scaling