Learning Bayesian Networks from Data
Nir Friedman
Hebrew U.
Daphne Koller
Stanford

Overview
- Introduction
- Parameter Estimation
- Model Selection
- Structure Discovery
- Incomplete Data
- Learning from Structured Data

Bayesian Networks
Compact representation of probability distributions via conditional independence

Qualitative part:
Directed acyclic graph (DAG)
- Nodes - random variables
- Edges - direct influence

Quantitative part:
Set of conditional probability distributions

Together:
Define a unique distribution in a factored form

Example: “ICU Alarm” network
Domain: Monitoring Intensive-Care Patients
- 37 variables
- 509 parameters

Inference
- Posterior probabilities
  - Probability of any event given any evidence
- Most likely explanation
  - Scenario that explains evidence
- Rational decision making
  - Maximize expected utility
  - Value of Information
- Effect of intervention

Why learning?
Knowledge acquisition bottleneck
- Knowledge acquisition is an expensive process
- Often we don’t have an expert

Data is cheap
- Amount of available information growing rapidly
- Learning allows us to construct models from raw data
Why Learn Bayesian Networks?

- Conditional independencies & graphical language capture structure of many real-world distributions
- Graph structure provides much insight into domain
  - Allows “knowledge discovery”
- Learned model can be used for many tasks
- Supports all the features of probabilistic learning
  - Model selection criteria
  - Dealing with missing data & hidden variables

Learning Bayesian networks

Known Structure, Complete Data

- Network structure is specified
  - Inducer needs to estimate parameters
  - Data does not contain missing values

Unknown Structure, Complete Data

- Network structure is not specified
  - Inducer needs to select arcs & estimate parameters
  - Data does not contain missing values

Known Structure, Incomplete Data

- Network structure is specified
  - Data contains missing values
  - Need to consider assignments to missing values

Unknown Structure, Incomplete Data

- Network structure is not specified
  - Data contains missing values
  - Need to consider assignments to missing values
Overview
- Introduction
- Parameter Estimation
  - Likelihood function
  - Bayesian estimation
- Model Selection
- Structure Discovery
- Incomplete Data
- Learning from Structured Data

Learning Parameters
- Training data has the form:

\[
D = \begin{bmatrix}
\vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

Likelihood Function
- Assume i.i.d. samples
- Likelihood function is

\[
L(\Theta : D) = \prod_i \prod_j \prod_k \prod_l \prod_m \prod_n \prod_o \prod_p \prod_q \prod_r \prod_s \prod_t \prod_u \prod_v \prod_w \prod_x \prod_y \prod_z \prod_{mL} \prod_{mP} \prod_{mD} \prod_{mE} \prod_{mC} : \Theta
\]

Rewriting terms, we get

\[
L(\Theta : D) = \prod_i \prod_j \prod_k \prod_l \prod_m \prod_n \prod_o \prod_p \prod_q \prod_r \prod_s \prod_t \prod_u \prod_v \prod_w \prod_x \prod_y \prod_z \prod_{mL} \prod_{mP} \prod_{mD} \prod_{mE} \prod_{mC} : \Theta
\]

Likelihood Function
- By definition of network, we get

\[
L(\Theta : D) = \prod_i \prod_j \prod_k \prod_l \prod_m \prod_n \prod_o \prod_p \prod_q \prod_r \prod_s \prod_t \prod_u \prod_v \prod_w \prod_x \prod_y \prod_z \prod_{mL} \prod_{mP} \prod_{mD} \prod_{mE} \prod_{mC} : \Theta
\]

General Bayesian Networks
- Generalizing for any Bayesian network:

\[
L(\Theta : D) = \prod_i \prod_j \prod_k \prod_l \prod_m \prod_n \prod_o \prod_p \prod_q \prod_r \prod_s \prod_t \prod_u \prod_v \prod_w \prod_x \prod_y \prod_z \prod_{mL} \prod_{mP} \prod_{mD} \prod_{mE} \prod_{mC} : \Theta
\]

Decomposition
\[
\Rightarrow \text{Independent estimation problems}
\]
**Likelihood Function: Multinomials**

\[ L(\theta : D) = P(D | \theta) = \prod_{i} P(x[m] | \theta) \]

- The likelihood for the sequence H, T, T, H, H is

\[ L(\theta : D) = \theta \cdot (1 - \theta) \cdot (1 - \theta) \cdot \theta \cdot \theta \]

**Bayesian Inference**

- Represent uncertainty about parameters using a probability distribution over parameters, data
- Learning using Bayes rule

\[ P(\theta | x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M] | \theta) P(\theta)}{P(x[1], \ldots, x[M])} \]

**Example: Binomial Data**

- Prior: uniform for \( \theta \) in \([0, 1]\)
- \( \Rightarrow P(\theta | D) \propto \text{the likelihood } L(\theta : D) \)
- \( P(\theta | x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M] | \theta) P(\theta)}{P(x[1], \ldots, x[M])} \)
- \( (N_H, N_T) = (4, 1) \)
- MLE for \( P(X = H) \) is \( 4/5 = 0.8 \)
- Bayesian prediction is

\[ P(x[M + 1] = H | D) = \int \theta P(\theta | D) d\theta = \frac{5}{7} = 0.7142... \]

**Dirichlet Priors**

- Recall that the likelihood function is

\[ L(\theta : D) = \prod_{i} \theta_{i}^{x_{i}} \]

- **Dirichlet** prior with hyperparameters \( \alpha_1, \ldots, \alpha_K \)

\[ P(\theta) = \prod_{i=1}^{K} \theta_{i}^{\alpha_{i} - 1} \]

- \( \Rightarrow \) the posterior has the same form, with

hyperparameters \( \alpha_1 + N_H, \ldots, \alpha_K + N_T \)

\[ P(\theta | D) = P(\theta) P(D | \theta) = \prod_{i=1}^{K} \theta_{i}^{\alpha_{i} + x_{i}} \cdot \prod_{i=1}^{K} \theta_{i}^{\alpha_{i} - 1} \]

**Dirichlet Priors - Example**
Dirichlet Priors (cont.)

- If $P(\theta)$ is Dirichlet with hyperparameters $\alpha_1, \ldots, \alpha_K$
  \[ P(X[1] = k) = \int \theta_k \frac{\alpha_k}{\sum_l \alpha_l} \]
- Since the posterior is also Dirichlet, we get
  \[ P(X[M+1] = k | D) = \int \theta_k \frac{\alpha_k + N_k}{\sum_l (\alpha_l + N_l)} \]

Bayesian Nets & Bayesian Prediction

- Priors for each parameter group are independent
- Data instances are independent given the unknown parameters

Learning Parameters: Summary

- Estimation relies on sufficient statistics
  - For multinomials: counts $N(x_i, p_a)$
  - Parameter estimation
    \[ \hat{\theta}_{MLE} = \frac{N(x_i, p_a)}{N(p_a)} \quad \hat{\theta}_{B} = \frac{N(x_i, p_a) + N'(x_i, p_a)}{N(p_a) + N'(p_a)} \]
- Both are asymptotically equivalent and consistent
- Both can be implemented in an on-line manner by accumulating sufficient statistics

Overview

- Introduction
- Parameter Learning
- Model Selection
  - Scoring function
  - Structure search
  - Structure Discovery
- Incomplete Data
- Learning from Structured Data
Why Struggle for Accurate Structure?

- Cannot be compensated for by fitting parameters
- Wrong assumptions about domain structure

Score-based Learning

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

Search for a structure that maximizes the score

Likelihood Score for Structure

\[ l(G : D) = \log L(G : D) = M \sum_i (I(X_i; Pa_i) - k(X_i)) \]

- Larger dependence of \( X \) on \( Pa \) \( \Rightarrow \) higher score
- Adding arcs always helps
  - \( I(X; Y) \leq I(X; \{Y, Z\}) \)
  - Max score attained by fully connected network
  - Overfitting: A bad idea...

Bayesian Score

Likelihood score: \( L(G : D) = P(D \mid \hat{G}, \hat{\theta}_G) \)

Bayesian approach:
- Deal with uncertainty by assigning probability to all possibilities
  \[ P(D \mid G) = \int P(D \mid G, \theta) P(\theta \mid G) d\theta \]
  \[ P(G \mid D) = \frac{P(D \mid G) P(G)}{P(D)} \]

Marginal Likelihood: Multinomials

Fortunately, in many cases integral has closed form

- \( P(\theta) \) is Dirichlet with hyperparameters \( \alpha_1, \ldots, \alpha_K \)
- \( D \) is a dataset with sufficient statistics \( N_1, \ldots, N_K \)

Then

\[ P(D) = \frac{\Gamma\left(\sum \alpha_i\right)}{\Gamma\left(\sum \alpha + N\right)} \prod \frac{\Gamma(\alpha_i + N_i)}{\Gamma(\alpha_i)} \]

Marginal Likelihood: Bayesian Networks

- Network structure determines form of marginal likelihood

Network 1: Two Dirichlet marginal likelihoods

\[ P(\cdot \mid \theta) \text{ integral over } \theta \]

\[ P(\cdot \mid \theta) \text{ integral over } \theta \]
Marginal Likelihood: Bayesian Networks

Network structure determines form of marginal likelihood

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Y</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
</tr>
</tbody>
</table>

Network 2:
Three Dirichlet marginal likelihoods

\[ P(\theta_x) \] Integral over \( \theta_x \)
\[ P(\theta_{y|x,H}) \] Integral over \( \theta_{y|x,H} \)
\[ P(\theta_{y|x,T}) \] Integral over \( \theta_{y|x,T} \)

Bayesian Score: Asymptotic Behavior

\[
\log P(D | G) = (G : D) - \frac{\log M \cdot \text{dim}(G)}{2} + O(1)
\]

- As \( M \) (amount of data) grows,
  - Increasing pressure to fit dependencies in distribution
  - Complexity penalty avoids fitting noise
  - Asymptotic equivalence to MDL score
  - Bayesian score is consistent
  - Observed data eventually overrides prior

Structure Search as Optimization

Input:
- Training data
- Scoring function
- Set of possible structures

Output:
- A network that maximizes the score

Key Computational Property: Decomposability:
\[
\text{score}(G) = \sum \text{score ( family of X in G )}
\]

Tree-Structured Networks

Trees:
- At most one parent per variable

Why trees?
- Elegant math
- We can solve the optimization problem
- Sparse parameterization
- Avoid overfitting

Learning Trees

- Let \( p(i) \) denote parent of \( X_i \)
- We can write the Bayesian score as

\[
\text{Score}(G : D) = \sum_i \text{Score}(X_i : p_i)
= \sum_i [\text{Score}(X_i : X_{p_i}) - \text{Score}(X_i)] + \sum \text{Score}(X)
\]

Score = sum of edge scores + constant
Learning Trees

- Set $w(j \rightarrow i) = \text{Score}(X_j \rightarrow X_i) - \text{Score}(X_i)$
- Find tree (or forest) with maximal weight
  - Standard max spanning tree algorithm $\sim O(n^2 \log n)$

Theorem: This procedure finds tree with max score

Beyond Trees

When we consider more complex network, the problem is not as easy
- Suppose we allow at most two parents per node
- A greedy algorithm is no longer guaranteed to find the optimal network
- In fact, no efficient algorithm exists

Theorem: Finding maximal scoring structure with at most $k$ parents per node is NP-hard for $k > 1$

Heuristic Search

- Define a search space:
  - search states are possible structures
  - operators make small changes to structure
- Traverse space looking for high-scoring structures
- Search techniques:
  - Greedy hill-climbing
  - Best first search
  - Simulated Annealing
  - ...

Local Search

- Start with a given network
  - empty network
  - best tree
  - a random network
- At each iteration
  - Evaluate all possible changes
  - Apply change based on score
- Stop when no modification improves score

Learning in Practice: Alarm domain

![Image of alarm domain diagram]

To update score after local change, only re-score families that changed

Structure known, fit params
Learn both structure & params

Typical operations:
- $S_{(C,E) \rightarrow D}$
- $S_{(E) \rightarrow D}$

To update score after local change, only re-score families that changed

KL Divergence to true distribution vs #samples
Local Search: Possible Pitfalls

- Local search can get stuck in:
  - Local Maxima:
    - All one-edge changes reduce the score
  - Plateaux:
    - Some one-edge changes leave the score unchanged
- Standard heuristics can escape both
  - Random restarts
  - TABU search
  - Simulated annealing

Improved Search: Weight Annealing

- Standard annealing process:
  - Take bad steps with probability $\propto \exp(-\Delta \text{score}/t)$
  - Probability increases with temperature
- Weight annealing:
  - Take uphill steps relative to perturbed score
  - Perturbation increases with temperature

Perturbing the Score

- Perturb the score by reweighting instances
- Each weight sampled from distribution:
  - Mean = 1
  - Variance $\propto$ temperature
- Instances sampled from "original" distribution
- … but perturbation changes emphasis

Benefit:
- Allows global moves in the search space

Weight Annealing: ICU Alarm network

Cumulative performance of 100 runs of annealed structure search

Structure Search: Summary

- Discrete optimization problem
- In some cases, optimization problem is easy
  - Example: learning trees
- In general, NP-Hard
  - Need to resort to heuristic search
  - In practice, search is relatively fast (~100 vars in ~2.5 min):
    - Decomposability
    - Sufficient statistics
  - Adding randomness to search is critical

Overview

- Introduction
- Parameter Estimation
- Model Selection
- Structure Discovery
- Incomplete Data
- Learning from Structured Data
Structure Discovery

Task: Discover structural properties
- Is there a direct connection between X & Y
- Does X separate between two “subsystems”
- Does X causally effect Y

Example: scientific data mining
- Disease properties and symptoms
- Interactions between the expression of genes

Discovering Structure

Problem
- Small sample size ⇒ many high scoring models
- Answer based on one model often useless
- Want features common to many models

Bayesian Approach

Posterior distribution over structures
Estimate probability of features
- Edge $X \rightarrow Y$
- Path $X \rightarrow \ldots \rightarrow Y$
- $\ldots$

$$P(f \mid D) = \sum_{G} f(G)P(G \mid D)$$

Feature of $G$, e.g., $X \rightarrow Y$
Indicator function for feature $f$

MCMC over Networks

- Cannot enumerate structures, so sample structures
  $$P(f(G) \mid D) = \frac{1}{1} \sum_{G} f(G)$$
- MCMC Sampling
  - Define Markov chain over BNs
  - Run chain to get samples from posterior $P(G \mid D)$

Possible pitfalls:
- Huge (superexponential) number of networks
- Time for chain to converge to posterior is unknown
- Islands of high posterior, connected by low bridges

ICU Alarm BN: No Mixing

- 500 instances:
- The runs clearly do not mix
Effects of Non-Mixing
- Two MCMC runs over same 500 instances
- Probability estimates for edges for two runs

Probability estimates highly variable, nonrobust

Fixed Ordering
Suppose that
- We know the ordering of variables
  - say, $X_1 > X_2 > X_3 > X_4 > \ldots > X_n$
  - parents for $X_i$ must be in $X_1, \ldots, X_{i-1}$
- Limit number of parents per nodes to $k$

Intuition: Order decouples choice of parents
- Choice of $P_q(X_i)$ does not restrict choice of $P_q(X_{i+1})$

Upshot: Can compute efficiently in closed form
- Likelihood $P(D \mid \neg \cdot)$
- Feature probability $P(f \mid D, \neg \cdot)$

Our Approach: Sample Orderings
We can write

$$P(f \mid D) = \sum_i P(f \mid <_i D) P(<_i D)$$

Sample orderings and approximate

$$P(f \mid D) = \sum_{i \leq k} P(f \mid <_i D)$$

- MCMC Sampling
  - Define Markov chain over orderings
  - Run chain to get samples from posterior $P(\cdot \mid D)$

Mixing with MCMC-Orderings
- 4 runs on ICU-Alarm with 500 instances
  - fewer iterations than MCMC-Nets
  - approximately same amount of computation

Process appears to be mixing!

Mixing of MCMC runs
- Two MCMC runs over same instances
- Probability estimates for edges

Probability estimates very robust

Application: Gene expression
Input:
- Measurement of gene expression under different conditions
  - Thousands of genes
  - Hundreds of experiments

Output:
- Models of gene interaction
  - Uncover pathways
Overview

- Introduction
- Parameter Estimation
- Model Selection
- Structure Discovery
- Incomplete Data
  - Parameter estimation
  - Structure search
  - Learning from Structured Data

Incomplete Data

Data is often *incomplete*

- Some variables of interest are not assigned values

This phenomenon happens when we have

- **Missing values:**
  - Some variables unobserved in some instances
- **Hidden variables:**
  - Some variables are never observed
  - We might not even know they exist

Hidden (Latent) Variables

Why should we care about unobserved variables?

- $P(X)$ assumed to be known
- Likelihood function of: $\theta_{Y|X=X} , \theta_{Y|X=H}$
- Contour plots of log likelihood for different number of missing values of $X \ (M = 8)$:

  - $\theta_{Y|X=T}$
  - $\theta_{Y|X=T}$
  - $\theta_{Y|X=T}$

  - no missing values
  - 2 missing values
  - 3 missing values

In general: likelihood function has multiple modes
Incomplete Data

- In the presence of incomplete data, the likelihood can have multiple maxima.
- Example:
  - If we can rename the values of hidden variable \( H \)
  - If \( H \) has two values, likelihood has two maxima
- In practice, many local maxima

Expectation Maximization (EM)

- A general purpose method for learning from incomplete data.

Intuition:

- If we had true counts, we could estimate parameters.
- But with missing values, counts are unknown.
- We “complete” counts using probabilistic inference based on current parameter assignment.
- We use completed counts as if real to re-estimate parameters.

Expectation Maximization (EM)

- Use current point to construct “nice” alternative function.
- Max of new function scores ≥ than current point.

Expectation Maximization (EM)

Formal Guarantees:

- Each iteration improves the likelihood.
- If \( \theta_1 = \theta_0 \), then \( \theta_0 \) is a stationary point of \( L(\theta; D) \).
- Usually, this means a local maximum.
Expectation Maximization (EM)

Computational bottleneck:
- Computation of expected counts in E-Step
  - Need to compute posterior for each unobserved variable in each instance of training set
  - All posteriors for an instance can be derived from one pass of standard BN inference

Summary: Parameter Learning with Incomplete Data

- Incomplete data makes parameter estimation hard
- Likelihood function
  - Does not have closed form
  - Is multimodal
- Finding max likelihood parameters:
  - EM
  - Gradient ascent
- Both exploit inference procedures for Bayesian networks to compute expected sufficient statistics

Incomplete Data: Structure Scores

Recall, Bayesian score:
\[ P(\mathcal{G} \mid D) \propto P(\mathcal{G}) P(D \mid \mathcal{G}) \]
\[ = P(\mathcal{G}) \mathbb{E}[P(D \mid \mathcal{G}, \theta)] \]

With incomplete data:
- Cannot evaluate marginal likelihood in closed form
- We have to resort to approximations:
  - Evaluate score around MAP parameters
  - Need to find MAP parameters (e.g., EM)

Naive Approach

- Perform EM for each candidate graph
  - Perform EM for each candidate graph $G_1$, $G_2$, $G_3$, ...
  - Computational expensive:
    - Parameter optimization via EM — non-trivial
    - Need to perform EM for all candidate structures
    - Spend time even on poor candidates
    - In practice, considers only a few candidates

Structural EM

Recall, in complete data we had
- Decomposition $\Rightarrow$ efficient search

Idea:
- Instead of optimizing the real score…
- Find decomposable alternative score
- Such that maximizing new score $\Rightarrow$ improvement in real score

Idea:
- Use current model to help evaluate new structures

Outline:
- Perform search in (Structure, Parameters) space
- At each iteration, use current model for finding either:
  - Better scoring parameters: “parametric” EM step
  - Better scoring structure: “structural” EM step
**Example: Phylogenetic Reconstruction**

**Input:** Biological sequences
- Human: CGTTGC…
- Chimp: CCTAGG…
- Orang: CGAACG…

**Output:** a phylogeny

An "instance" of evolutionary process

Assumption: positions are independent

---

**Phylogenetic Model**

- Topology: bifurcating
  - Observed species – 1…N
  - Ancestral species – N+1…2N-2
- Lengths $l = (t_{i,j})$ for each branch $(i,j)$
- Evolutionary model:
  - $P(A \text{ changes to } T | 10 \text{ billion yrs})$

---

**Phylogenetic Tree as a Bayes Net**

- Variables: Letter at each position for each species
  - Current day species – observed
  - Ancestral species - hidden
- BN Structure: Tree topology
- BN Parameters: Branch lengths (time spans)

Main problem: Learn topology

If ancestral were observed
⇒ easy learning problem (learning trees)

---

**Algorithm Outline**

- Compute expected pairwise stats
- Weights: Branch scores

Original Tree ($T, \pi$)

---

**Algorithm Outline**

- Compute expected pairwise stats
- Weights: Branch scores
- Find: $T^* = \arg \max_{T} \sum_{i,j} w_{i,j}$

Pairwise weights

$O(N^2)$ pairwise statistics suffice to evaluate all trees
Algorithm Outline

→ Compute expected pairwise stats
→ Weights: Branch scores
→ Find: $\mathcal{T} = \arg \max_{\mathcal{T}} \sum_{ij} w_{ij}$
→ Construct bifurcation $\mathcal{T}_1$

Max. Spanning Tree

$\sum \in \mathcal{T}_{ji} \max \arg'$

Find:

Repeat until convergence…

$\mathcal{T}_1$

Algorithm Outline

→ Compute expected pairwise stats
→ Weights: Branch scores
→ Find: $\mathcal{T} = \arg \max_{\mathcal{T}} \sum_{ij} w_{ij}$
→ Construct bifurcation $\mathcal{T}_1$
→ Theorem: $L(\mathcal{T}_0, t_0) \geq L(\mathcal{T}_1, t_1)$

Real Life Data

<table>
<thead>
<tr>
<th></th>
<th>Lysozyme $c$</th>
<th>Mitochondrial genomes</th>
</tr>
</thead>
<tbody>
<tr>
<td># sequences</td>
<td>43</td>
<td>34</td>
</tr>
<tr>
<td># pos</td>
<td>122</td>
<td>3,578</td>
</tr>
<tr>
<td>Traditional approach</td>
<td>-2,916.2</td>
<td>-74,227.9</td>
</tr>
<tr>
<td>Structural EM Approach</td>
<td>-2,892.1</td>
<td>-70,533.5</td>
</tr>
<tr>
<td>Difference per position</td>
<td>0.19</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Each position twice as likely

Overview

- Introduction
- Parameter Estimation
- Model Selection
- Structure Discovery
- Incomplete Data
- Learning from Structured Data

Bayesian Networks: Problem

- Bayesian nets use propositional representation
- Real world has objects, related to each other

Bayesian Networks: Problem

- Bayesian nets use propositional representation
- Real world has objects, related to each other

These “instances” are not independent!
St. Nordaf University

Relational Schema
- Specifies types of objects in domain, attributes of each type of object, & types of links between objects

Representing the Distribution
- Many possible worlds for a given university
  - All possible assignments of all attributes of all objects
- Infinitely many potential universities
  - Each associated with a very different set of worlds

Possible Worlds
- World: assignment to all attributes of all objects in domain

Probabilistic Relational Models
Key ideas:
- Universals: Probabilistic patterns hold for all objects in class
- Locality: Represent direct probabilistic dependencies
  - Links give us potential interactions!

PRM Semantics
- Instantiated PRM = BN
  - variables: attributes of all objects
  - dependencies: links & PRM

Welcome to CS101
Welcome to Geo101
The Web of Influence

- Objects are all correlated
- Need to perform inference over entire model
- For large databases, use approximate inference:
  - Loopy belief propagation

PRM Learning: Complete Data

- Introduce prior over parameters
- Update prior with sufficient statistics:
  \[ \text{Count}(\text{Reg. Grade}=A, \text{Reg. Course. Difficulty}=\text{lo}, \text{Reg. Student. Intelligence}=\text{hi}) \]

PRM Learning: Incomplete Data

- Use expected sufficient statistics
- But, everything is correlated:
  \[ \text{E-step uses (approx) inference over entire model} \]

A Web of Data

- Tom Mitchell
  - Professor
- Sean Slattery
  - Student
- CMU CS Faculty

Standard Approach

What's in a Link

- From-Page
  - Category
  - Word
  - Word
- To-Page
  - Category
  - Word
  - Word
- Link
Discovering Hidden Concepts

Web of Influence, Yet Again

Movies
- The Lord of the Rings
- The Silence of the Lambs
- The Love Bug
- The Parent Trap
- Pulp Fiction
- The Dark Knight
- The Dark Knight Rises

Actors
- Anthony Hopkins
- Robert De Niro
- Morgan Freeman
- Gary Oldman
- Kevin Spacey
- Sean Connery

Directors
- Steven Spielberg
- Martin Scorsese
- Quentin Tarantino
- Christopher Nolan

Conclusion
- Many distributions have combinatorial dependency structure
- Utilizing this structure is good
- Discovering this structure has implications:
  - To density estimation
  - To knowledge discovery
- Many applications
  - Medicine
  - Biology
  - Web

The END

Thanks to
- Gal Elidan
- Lise Getoor
- Moises Goldszmidt
- Matan Ninio
- Dana Pe’er
- Eran Segal
- Ben Taskar

Slides will be available from:
- http://www.cs.huji.ac.il/~nir/
- http://robotics.stanford.edu/~koller/