Exploiting Statistical and Relational Information on the Web and in Social Media: Applications, Techniques, and New Frontiers

Part II: Survey of statistical relational learning and inference techniques
Part II Road Map

- “Flat” Relational Models
  - Describe flat relational models
  - Illustrate with 2 case studies
  - Aside 1: Aggregation functions
  - Aside 2: Using first-order logic to define relational features

- Collective Classification Models

- Richer SRL Models
“Flat” Relational Models

Given:

Task: Predict attribute of some of the entities

Alternate task: Predict existence of relationship between entities

© Getoor & Mihalkova 2010
“Flat” Relational Models

- Relational features are pre-computed by aggregating over related entities
- Values are represented as a fixed-length feature vector
- Instances are treated independently of each other
- Any classification or regression model can be used for learning and prediction
Next we present two applications that use a “flat” relational model
- Focus is on types of relational features used

Case Study 1: Predicting click-through rate of search result ads
Case Study 2: Predicting friendships in a social network
Case Study 1: Predicting Ad Click-Through Rate

- Task: Predict the click-through rate (CTR) of an online ad, given that it is seen by the user, where the ad is described by:
  - URL to which user is sent when clicking on ad
  - Bid terms used to determine when to display ad
  - Title and text of ad

- Our description is based on approach by
  - [Richardson et al., WWW07]
Relational Features Used

- Based on [Richardson et al., WWW07]

CTR?

Ad

BT1 contains-bid-term Ad1

BT2 contains-bid-term (according to search engine) Ad2

BT3 contains-bid-term Ad3

BT4 related-bid-term (containing subsets or supersets of the term)

Ad1

Ad2

Ad3

Ad4

Ad5

Ad6

Average CTR

Count

© Getoor & Mihalkova 2010
Case Study II: Predicting Friendships

- Based on [Zheleva et al., SNAKDD08] who experimented with social networks of pets

- Task: Predict new friendships among users, based on their descriptive attributes, their existing friendships, and their family ties.
Based on [Zheleva et al., SNAKDD08] who experimented with social networks of pets.
Aside: Aggregation Functions

- Review of some common aggregation functions
Global Measures

- Summarize properties of entire graph

- Next we discuss:
  - Clustering coefficient
  - Bipolarity

© Getoor & Mihalkova 2010
Clustering Coefficient

- Measures cliquishness of an undirected, unweighted graph, or its tendency to form small clusters.
- Computed as the proportion of all incident edge pairs that are completed by a third one to form a triangle.

\[
CC(G) = \frac{1}{|V|} \sum_{v \in V} CC(v)
\]

\[
CC(v) = \frac{|\{(i, j) \in E | i \in N_v \land j \in N_v\}|}{\frac{1}{2}k_v(k_v - 1)} = \frac{\text{Num actual neighbor links}}{\text{Possible num neighbor links}}
\]
Extensions exist for

- directed graphs [Kunegis et al. WWW09]
- graphs with weighted edges [Kalna & Higham, AICommunic07]
- graphs with signed edges [Kunegis et al. WWW09]
Bipolarity

- Defined on a weighted directed graph
- Measures to what extent the nodes in the graph are organized in two opposing camps
  - i.e., how close is the graph to being bipartite

\[ \text{Bipolarity}(G) = \frac{w - c}{w + c} \]

- Value between -1 and +1

Max Cut

© Getoor & Mihalkova 2010
Node-specific Measures

- **Attribute-based:**
  - Count, proportion, mode, most common value
  - Values most commonly related to positive/negative training cases
    - See [Perlich & Provost, KDD03] for extended treatment

- **Structural:**
  - PageRank [Page et al., TR98] and others as measures of popularity
  - Measures from social networks literature
    - CC(v) – clustering coefficient at a node
    - Degree centrality
    - Betweenness Centrality
    - For more, see [Wasserman & Faust, Book94]
Degree Centrality

- A very simple but useful aggregation:
  - Degree centrality of a node = number of neighbors

- Sometimes normalized by the total number of nodes in the graph
Betweenness Centrality

- A node $a$ is more central if paths between other nodes must go through it; i.e. more node pairs need $a$ as a mediator

\[ BC(a) = \sum_{j<k} \frac{|SP_{\sim a}(j, k)|}{|SP(j, k)|} \]

Number of shortest paths between $j$ and $k$ that go through $a$

Total number of shortest paths between $j$ and $k$
Node-Pairs Measures

- Measures defined on pairs of nodes
  - Attribute similarity measures to compare nodes based on their attributes
    - String similarity, Hamming distance, cosine
  - Set similarity measures to compare nodes based on set of nodes to which they are related, e.g.,
    - Average similarity between set members
    - Jaccard coefficient
    - Preferential attachment score
    - Adamic/Adar measure
    - SimRank
    - Katz score
  - For detailed treatment, see [Liben-Nowell & Kleinberg, JASIST07]
Jaccard Coefficient

- Compute overlap between two sets
  - e.g., compute overlap between sets of friends of two entities

\[ \text{Jaccard}(P_1.\text{Friends}, P_2.\text{Friends}) = \frac{|P_1.\text{Friends} \cap P_2.\text{Friends}|}{|P_1.\text{Friends} \cup P_2.\text{Friends}|} \]
Preferential Attachment Score

[Liben-Nowell & Kleinberg, JASIST07]

Based on studies, e.g. [Newman, PRL01], showing that people with a larger number of existing relations are more likely to initiate new ones.

\[ s(a, b) = |N_a| \cdot |N_b| \]

Set of a’s neighbors
Adamic/Adar Measure

- Two users are more similar if they share more items that are overall less frequent.

\[ s(a, b) = \sum_{i \in \text{Shared items}} \frac{1}{\log(\text{frequency}(i))} \]

Can be any kind of shared attributes or relationships to shared entities.

Overall frequency in the data.
SimRank

- “Two objects are similar if they are related to similar objects”
- Defined as the unique solution to:

\[
s(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{\text{Set of incoming edges into } a} \sum_{j=1}^{\text{Decay factor between 0 and 1}} s(I_i(a), I_j(b))
\]

- Computed by iterating to convergence
- Initialization to \( s(a, b) = 1 \) if \( a=b \) and 0 otherwise
Katz Score

- Two objects are similar if they are connected by shorter paths

\[ s(a, b) = \sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}^{(l)}(a, b)| \]

Set of paths between a and b of length exactly \( l \)

Decay factor between 0 and 1

© Getoor & Mihalkova 2010
End Aside on Aggregation Functions
Aside: Using First-Order Logic to Define Relational Features

- Review of an important class of representations
An especially important class of models draws on the expressivity of first order logic to define features.

Inductive logic programming (ILP)
- Learning: Induce a set of first-order logic rules
- Inference: Uses logical inference, e.g. resolution
- Algorithms: FOIL, ALEPH, and many others

“Upgraded” Learners
- Take an existing learner and “upgrade” it to use first-order logic features [Van Laer & De Raedt, BkChapter01]
- Systems: TILDE (decision trees), kFOIL (SVMs), structural logistic regression, and others

© Getoor & Mihalkova 2010
Relations and attributes are represented as predicates.

- **Predicate**: \( \text{Hyperlink}(A, B) \)
- **Literal**: \( \text{Hyperlink}(\text{aaai.org, westin.com}) \)
- **Variable**: \( \text{Entity} \)
Liketals are combined to form **clauses**:

\[ \text{HasCategory}(A, C1) \land \text{Hyperlink}(A, B) \Rightarrow \text{HasCategory}(B, C1) \]

... is logically equivalent to

\[ \neg\text{HasCategory}(A, C1) \lor \neg\text{Hyperlink}(A, B) \lor \text{HasCategory}(B, C1) \]

- **A Horn clause** has at most one positive literal in the disjunction

© Getoor & Mihalkova 2010
FOIL

- Given: background facts, positive and negative examples for a target relation
- Task: induce a set of Horn clauses that cover all of the positive and none of the negative examples
- The FOIL Algorithm:
  - While positive examples remain uncovered
    - New clause \( C = \text{target-rel}(V_1, V_2, \ldots, V_n) \Leftarrow \)
    - While \( C \) has unbound variables or is incorrect
      - Specialize:
        - Find appropriate literal \( L \) to add
        - Add \( L \) to the body of \( C \)
      - Remove unnecessary literals from \( C \)
    - Remove positive examples covered by the new clause

Quinlan, MLJ90

© Getoor & Mihalkova 2010
“Upgraded” Learners

- **TILDE**
  - [Blockeel & De Raedt, AI98]
  - Learn decision trees whose nodes contain logical queries rather than attribute-value tests

- **kFOIL**
  - [Landwehr et al., MLJ10]
  - First-order logic features are used to define kernels over relational objects

- **Structural logistic regression**
  - [Popescul & Ungar, MRDM-KDD-WkShop03]
  - Features of the logistic regression classifier are defined as logical queries that allow for aggregate functions
Case Study: WebKB

- One of the earliest applications of relational learning on the web
- Collected web pages from academic departments
- Link structure is exploited
  1. For web page classification into one of several categories:
     - Department, Faculty, Student, Research Project, Course, Other
  2. For predicting relations between entities represented by pages
     - Department of a person, members of a project, instructors of a class

[Craven et al., ECML98]

© Getoor & Mihalkova 2010
WebKB Predicting Class

- Evidence Predicates
  - Has_Word(Page)
  - LinkTo(Page, Page)

- Unknown (Predicted) Predicates
  - StudentPage(Page)
  - FacultyPage(Page)
  - etc.

\[
\text{Has\_Professor}(A) \land \text{Has\_ph}(A) \land \text{LinkTo}(B, A) \land \text{Has\_faculty}(B) \Rightarrow \text{FacultyPage}(A)
\]
Evidence Predicates
- Class(Page), for each of the classes, e.g. StudentPage(Page)
- LinkTo(Hyperlink, Page, Page)
- HasWord(Hyperlink)
- AllWordsCapitalized(Hyperlink)
- HasAlphaNumericWord(Hyperlink)
- HasNeighborhood_Word(Hyperlink)

Unknown Predicates
- DepartmentOfPerson(Page, Page)
- InstructorOfCourse(Page, Page)
- MemberOfProject(Page, Page)
Example Rule

\[ \text{ResearchProject}(A) \land \text{Person}(B) \]
\[ \land \text{LinkTo}(C, A, D) \land \text{LinkTo}(E, D, B) \]
\[ \land \text{HasNeighborhood\_People}(C) \]
\[ \Rightarrow \text{MembersOfProject}(A, B) \]
[Craven et al., ECML98] used relational pathfinding to discover rules.

Simple but effective technique also exploited in more recent work on structure learning in SRL.

$\text{Movie}(T, A) \land \text{Movie}(T, B) \Rightarrow \text{WorkedFor}(A, B)$

Director($B$)
End Aside on FOL Features

- Back to discussion of flat relational models
Flat Relational Models: Pros

- Efficient
  - Can handle large amounts of data
    - Features can often be pre-computed ahead of time
  - One of the most commonly-used ways of incorporating relational information

- Flexible
  - Can take advantage of well-understood classification/regression algorithms

© Getoor & Mihalkova 2010
Flat Relational Models: Cons

- Relational features cannot be based on attributes or relations that are being predicted
  - For example:
Example

CTR? → Ad → BT₁ → BT₂ → BT₃ → Ad₁ → Ad₂ → Ad₅ → Ad₆

Average CTR

CTR of these ads have to be observed
If $P_1$ and $P_2$ become friends, $P_7$ and $P_{11}$ are likely to also become friends.
Relational features cannot be based on attributes or relations that are being predicted

- For example:

... but a couple of caveats:

- This can be overcome by proceeding in two rounds:
  1. Make predictions using only observed features and relations
  2. Make predictions using observed features and relations and predictions of unobserved ones from round 1.

- Inductive Logic Programming techniques for learning “recursive” clauses exist that allow the model to prove further examples from previously proven ones.

We’ll see a general approach to doing this.
Flat Relational Models: Cons

- Relational features cannot be based on attributes or relations that are being predicted
  - For example:

- Cannot impose global constraints on joint assignments
  - For example, when inferring a hierarchy of individuals, we may want to enforce constraint that it is a tree
Part II Road Map

- “Flat” Relational Models
- Collective Classification Models
- Richer SRL Models
Collective Models

- Disadvantages of flat relational models can be addressed by making collective predictions
  - Can help correct errors
  - Can coordinate assignments to satisfy constraints

- Collective models have been widely studied. Here we present a derivation based on extending flat relational representations
Towards Collective Models

To simplify matters, suppose $y_i$ is binary.

If we use Logistic regression, let's make the features be a function of $y_i$.

Same thing with new features.
Towards Collective Models 2

This trivial transformation makes it easy to generalize to features that are functions of more than one $y_i$, thus forcing the model to make collective decisions...

$$P(y_i; \theta) = \frac{\exp(\theta \cdot f_i(y_i))}{\sum_{y_i' \in \{0,1\}} \exp(\theta \cdot f_i(y_i'))}$$

$y = [y_1, y_2, \ldots, y_m]$

$$P(y; \theta_L, \theta_G) = \frac{\exp \left( \sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i, y_j) \right)}{\sum_{y'} \exp \left( \sum_i \theta_L \cdot f_i(y_i') + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i', y_j') \right)}$$

- Local and global parameter vectors
- Compute probability over joint assignment to all instances
- Sum local features, as before
- Sum features that are functions of more than one $y_i$; in general, could be more than 2

normalize
Towards Collective Models 2

This trivial transformation makes it easy to generalize to features that are functions of more than one $y_i$, thus forcing the model to make collective decisions...

$$P(y_i; \theta) = \frac{\exp(\theta \cdot f_i(y_i))}{\sum_{y'_i \in \{0,1\}} \exp(\theta \cdot f_i(y'_i))}$$

becomes

$$i, j \in E$$

$$P(y; \theta_L, \theta_G) = \frac{\exp \left( \sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i, y_j) \right)}{\sum_{y'} \exp \left( \sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i, y'_j) \right)}$$

Often not all possible pairs are considered, but just ones that are related.
Good news:
- Now we have a way of coordinating the assignments to the query attributes/relationships

Bad news:
- Looks like we have to enumerate over all possible joint assignments

$$P(y; \theta_L, \theta_G) = \frac{\exp \left( \sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i, y_j) \right)}{\sum_{y'} \exp \left( \sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i, y'_j) \right)}$$

... but there are ways around this
Collective Models

- **Iterative Classification Algorithm (ICA)**
  - Make coherent joint assignments by iterating over individual decision points, changing them based on current assignment to related decision points

- **Graphical Models**
  - Define a joint probability model over all decision points

- **Statistical Relational Models**
  - Generalize graphical models by tying some of the parameters based on “lifted” representations.
Iterative Classification Algo. (ICA)

[Neville & Jensen, SRL00; Lu & Getoor, ICML03]

- Extends flat relational models by allowing relational features to be functions of predicted attributes/relations of neighbors
- At training time, these features are computed based on observed values in the training set
- At inference time, the algorithm iterates, computing relational features based on the current prediction for any unobserved attributes
  - In the first, bootstrap, iteration, only local features are used
ICA: Learning

- **label set:**

![Diagram with nodes labeled P1, P2, P3, P4, P5, P6, P7, P8, P9, and P10 connected by arrows.]

Learn models (local and relational) from fully labeled training set
ICA: Inference (1)

Step 1: Bootstrap using entity attributes only
Step 2: Iteratively update the category of each entity, based on related entities’ categories
ICA Summary

- Simple approach for coupled classification
- Builds on classic techniques such as iterated conditional modes (ICM)

Related Work:
- Cautious Inference
  - [McDowell et al., JMLR09]
- Weighted neighbor
  - [Macskassy, AAAI07]
- Active Learning
  - [Bilgic et al., TKDD09, ICML10, AAAI10]
An alternative is to use joint inference in graphical models.

Here we cannot possibly describe all there is to graphical models.

Instead, we will define the most general formalism, that of factor graphs.
Let’s go back to our joint probability distribution:

\[ y = [y_1, y_2, \ldots, y_m] \]

\[
P(y; \theta_L, \theta_G) = \frac{\exp \left( \sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i, y_j) \right)}{\sum_{y'} \exp \left( \sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i, y'_j) \right)}
\]

The factor graph representation of this is:
Factor Graphs

- Each light green square represents $\exp(\theta_L \cdot f_i(y_i))$.
- Each dark square represents $\exp(\theta_G \cdot f_{i,j}(y_i, y_j))$. 
More Generally...

- Factors can be functions of any number of variables

However, to keep the model compact, we want to keep factors small. In the worst case, the number of parameters needed by a factor is exponential in the number of variables of which it is a function.

- Not all pairs of variables have to share a factor

In fact, we want to avoid having variables share factors unless there truly is a dependence between them.

- Factors can be computed by any function that returns a strictly positive value

The log-linear representation is convenient and has nice properties.
Now, $y_3$ is conditionally independent of $y_1$, given $y_2$ and $y_4$. 

... in the most basic representation:

```
<table>
<thead>
<tr>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_4$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$v_1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$v_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$v_8$</td>
</tr>
</tbody>
</table>
```
Markov Nets

- Markov networks (aka Markov random fields) can be viewed as special cases of factor graphs:

  Same Markov net could indicate that $y_2$, $y_3$, and $y_4$ share a single factor.

Equivalent expressivity. However, factor graphs are more explicit.
Factors are called potential functions.

Viewed as functions that ensure compatibility between assignments to the nodes.

For example, in the Ising Model the possible assignments are \{-1, +1\}, and one has:

\[ \phi_{i,j} = \exp(\theta_{i,j} y_i y_j) \]

Positive, or ferromagnetic, \(\theta_{i,j}\) encourages neighboring nodes to have the same assignment.

Negative, or anti-ferromagnetic, \(\theta_{i,j}\) encourages contrasting assignment.

Variables participating in shared potential functions form cliques in the graph.
Markov Nets: Transitivity

- How to encode transitivity?

  Want to say: If A is friends with B and B is friends with C, then A is friends with C. For all permutations of the letters.

- Model as a Markov net with a node for each decision, connecting dependent decisions in cliques

- Possible assignments: 1 (friends), 0 (not friends)
Quick Aside: Two Kinds of Graphs

In Part I we were drawing social networks like this:

**Relational Graph:**
- Nodes represent entities
- Edges represent relationships

... not to be confused with a Markov net:

**Markov Net:**
- Nodes represent decisions
- Edges represent dependencies between decisions

© Getoor & Mihalkova 2010
Quick Aside: Two Kinds of Graphs

In Part I we were drawing social networks like this:

Relational Graph:
- Nodes represent entities
- Edges represent relationships

... not to be confused with a Markov net:

Markov Net:
- Nodes represent decisions
- Edges represent dependencies between decisions

Since here we are trying to infer the presence of a relationship, our Markov Net has a node for each possible edge in the Relational graph.
Markov Nets: Transitivity

- How to encode transitivity?

  Want to say: If A is friends with B and B is friends with C, then A is friends with C. For all permutations of the letters.

- Model as a Markov net with a node for each decision, connecting dependent decisions in cliques

- Possible assignments: 1 (friends), 0 (not friends)
### Markov Nets: Transitivity

**Equations:**
- $y_1 = (A \rightarrow B)$
- $y_2 = (B \rightarrow C)$
- $y_3 = (A \rightarrow C)$

<table>
<thead>
<tr>
<th></th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Graphical Representation:**
- $\phi_{1,2,3}$
- Edges: $e^0$, $e^{-\theta}$

**One Possibility:**
- $\checkmark$
If A and B are enemies and B and C are enemies, then A and C are friends. For all permutations of the letters.

<table>
<thead>
<tr>
<th></th>
<th>$y_1=(A \leftrightarrow B)$</th>
<th>$y_2=(B \leftrightarrow C)$</th>
<th>$y_3=(A \leftrightarrow C)$</th>
<th>$\phi_{1,2,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>e$^{-\theta}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>e$^\theta$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>e$^\theta$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>e$^\theta$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>e$^{0}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>e$^{0}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>e$^{0}$</td>
</tr>
</tbody>
</table>

© Getoor & Mihalkova 2010
To cast a Bayesian net as a factor graph, include a factor as a function of each node and its parents.

Going the other way requires ensuring acyclicity.
To cast a Bayesian net as a factor graph, include a factor as a function of each node and its parents.

Here the factors take the shape of conditional probability tables, giving, for each configuration of assignments to the parents, the distribution over assignments to the child.

Automatically normalized!
Next we consider an application that uses a Markov network to perform keyword generation.

Based on [Fuxman et al., WWW08]
Keyword Generation

- Given: A set of concepts, each represented by a set of URLs that exemplify it
- Goal: For each query and each remaining URL, infer what concept they belong to
  - [Fuxman et al., WWW08] modeled this as a Markov network, including a node for each query and each URL and an edge between any query/URL pair that is related by a click
  - This Markov net essentially has the structure of the query log graph we considered in Part I
Keyword Generation

\[ \psi(i, j) = \exp\left(-\text{freq}_{i,j}(l_i - l_j)^2\right) \]

Number of times query for \( i \) is followed by click to \( j \)

Label of node \( j \)

© Getoor & Mihalkova 2010
How to introduce the evidence provided by the seed URLs?

As factors that evaluate to 1 for the given assignment and to 0 for all others!
Quick Aside: Two Kinds Of Graphs

- Here the factor graph and the relational graph look pretty much the same:

![Graph Diagram]
Quick Aside: Two Kinds Of Graphs

Here the factor graph and the relational graph look pretty much the same:

They differ in the same way as before:

- In Relational Graph, nodes represent queries and URLs; edges represent click-for relationships
- In Factor Graph, nodes represent decisions about the concept to which each query or URL belongs; edges represent dependencies between these decisions
Inference in Factor Graphs

- Goal of Inference: compute the marginal probability at each decision node in the factor graph

- We will briefly describe two of the most common algorithms
  - Gibbs sampling
  - Belief propagation

- Describing inference for factor graphs automatically makes it applicable to Markov and Bayesian nets
Gibbs Sampling

- Assign initial values to the variables
- While there is time:
  - For each variable i:
    - Sample a new value for i
    - Factor values are computed using current assignments to other participating variables

© Getoor & Mihalkova 2010
Gibbs Sampling

\[ x \in \{0, 1\} \]

\[
P(i = x) = \frac{\phi_i(x) \phi_{i,j,k}(x, 0, 1) \phi_{i,m}(x, 1) \phi_{i,n}(x, 0)}{\phi_i(x) \phi_{i,j,k}(0, 0, 1) \phi_{i,m}(0, 1) \phi_{i,n}(0, 0) + \phi_i(x) \phi_{i,j,k}(1, 0, 1) \phi_{i,m}(1, 1) \phi_{i,n}(1, 0)}
\]

Normalize by summing over every possible value of \( x \)
In general...

- $\mathcal{A}$: Current assignment of values to all variables in the model
- $\mathcal{A}[\mathbf{v}]$: Assignment of values to all variables in set $\mathbf{v}$
- $\Phi_i$: Set of factors in which variable $i$ participates
- $\mathbf{V}_\phi$: Set of variables used by factor $\phi$
- $\mathcal{X}$: Set of possible values

$$P(i = x) = \frac{\prod_{\phi \in \Phi_i} \phi(i = x, \mathbf{v}_\phi \setminus \{i\} = \mathcal{A}[\mathbf{v}_\phi \setminus \{i\}])}{\sum_{x' \in \mathcal{X}} \prod_{\phi \in \Phi_i} \phi(i = x', \mathbf{v}_\phi \setminus \{i\} = \mathcal{A}[\mathbf{v}_\phi \setminus \{i\}])}$$

Set other vars in the factor to their current value
Algorithmically, Gibbs sampling and ICA are very similar. There are several important distinctions.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>ICA</th>
<th>Gibbs Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End Goal</strong></td>
<td>Find most likely joint assignment to vars</td>
<td>Compute posterior marginal probabilities for vars</td>
</tr>
<tr>
<td><strong>In each iteration...</strong></td>
<td>Set the most probable value based on current assignments</td>
<td>Sample a new value based on current assignments</td>
</tr>
<tr>
<td><strong>Graph over which it is performed</strong></td>
<td>Relational graph</td>
<td>Factor graph</td>
</tr>
<tr>
<td><strong>Initialization</strong></td>
<td>Bootstrap from local features</td>
<td>Random/MAP state + burn-in</td>
</tr>
</tbody>
</table>

© Getoor & Mihalkova 2010
Belief Propagation

Based on messages sent from variables to their factors and vice versa.

At convergence:

$$P(i) \propto \mu_{\phi \rightarrow i(i)} \mu_{\psi \rightarrow i(i)} \mu_{\tau \rightarrow i(i)}$$
Parameter Learning

- Given: A data set $\mathcal{D}$, and a factor graph $\mathcal{F}$, whose factors are parameterized by a vector $\theta$
- Goal: Estimate values for $\theta$ to maximize prob. of the data

$$\theta^* = \arg\max_{\theta} P(\mathcal{D}|\mathcal{F}_\theta)$$

- This time, it will actually be helpful to consider Markov nets and Bayesian nets separately

This is one possible criterion for optimizing weights. More exist, e.g. Bayesian learning
... in Bayesian nets

- Easy! (From fully observed data)
- Just collect empirical counts

\[
P(y_1 = x \mid y_2 = \alpha, y_4 = \beta) = \frac{\text{count}(y_1 = x, y_2 = \alpha, y_4 = \beta)}{\sum_{x'} \text{count}(y_1 = x', y_2 = \alpha, y_4 = \beta)}
\]

- Use EM if there are missing values
... in Markov nets

- Cannot be computed in closed form
- Use gradient descent or any other optimization procedure
- If we use a log-linear model with one parameter per factor,
  - gradient wrt $\theta_i$ corresponding to factor $\phi_i$ is given by:

$$
\frac{\partial}{\partial \theta_i} = \phi_i(D) - \mathbb{E}_{\theta_i}(\phi_i)
$$

Value in the data

Expected value according to current estimate

© Getoor & Mihalkova 2010
Structure Learning

- Given: A data set $\mathcal{D}$
- Goal: Induce a factor graph $\mathcal{F}$ to maximize prob. of the data

Two flavors of approaches

- Independence-based: Use statistical tests to first determine conditional independencies among the variables
  - e.g., [Spirtes et al., Book01]
- Generate-and-test: generate candidate factors and score them based on how well they model the data
  - e.g. [Della Pietra et al., IEEE97; Heckerman, TR95]
Independence-Based

- Use a series of statistical tests to identify conditional independencies among the variables
  - In Bayesian nets corresponds to finding a set of parents for each of the nodes
  - In Markov nets corresponds to finding the Markov blanket of each node
    - i.e., the set of nodes that render it conditionally independent from the rest
Generate-And-Test

- Perform greedy search through hypothesis space
  - At each step generate new factors from current ones
    - e.g., in Markov nets new factors may be formed as conjunctions of existing factors and atomic features
    - in Bayesian nets edges may be added, removed, or reversed
  - Test generated candidate factors based on improvement in data likelihood score
  - Keep new factors that lead to an improvement

... next, let’s see how this works in the special case of Bayesian nets
Heuristic Search

We address the problem by using heuristic search

- Define a search space:
  - nodes are possible structures
  - edges denote adjacency of structures

- Traverse this space looking for high-scoring structures

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

Slide credit: Nir Friedman & Daphne Koller
Heuristic Search (cont.)

- Typical operations:

Slide credit: Nir Friedman & Daphne Koller
Exploiting Decomposability in Local Search

- **Caching**: To update the score of after a local change, we only need to re-score the families that were changed in the last move.

*Slide credit: Nir Friedman & Daphne Koller*
Greedy Hill-Climbing

- Simplest heuristic local search
  - Start with a given network
    - empty network
    - best tree
    - a random network
  - At each iteration
    - Evaluate all possible changes
    - Apply change that leads to best improvement in score
    - Reiterate
  - Stop when no modification improves score
- Each step requires evaluating approximately $n$ new changes

Slide credit: Nir Friedman & Daphne Koller
Part II Recap So Far

- Started with Flat Relational models
  - Pros: efficient, flexible, can incorporate variety of features
  - Cons: cannot make collective decisions, cannot impose global constraints
- To address above cons, we considered collective classification models
  - ICA: learning and inference
  - Factor graphs: learning and inference
- Are we done?
  - Let’s reconsider the transitivity example
Want to say: If A is friends with B and B is friends with C, then A is friends with C. For all permutations of the letters.

\[ y_1 = (A \leftrightarrow B) \]
\[ y_2 = (B \leftrightarrow C) \]
\[ y_3 = (A \leftrightarrow C) \]
\[ \phi_{1,2,3} \]

What if we have more than 3 individuals?

We can have a separate \( \phi \) for every subset of 3.

However, since we expect transitivity to always work the same way, we can have all these separate \( \phi \)-s share a set of parameters.
Parameter Tying

- Factors with tied parameters
  - Means that they share their parameter vectors
  - Can view them as a function that gets evaluated for different (sets of) nodes in the graph

- Advantages of tying:
  - Fewer parameters to estimate
    - Avoid overfitting
    - More robust estimation
  - Better generalization
    - E.g., we learn about transitivity in general, not about the transitivity between Ann, Bob, and Carl’s friendships
Learning with Tied Parameters

- In Bayesian Nets:

\[
P(y_1 = x | y_2 = \alpha, y_4 = \beta) = \frac{\text{count}(y_1 = x, y_2 = \alpha, y_4 = \beta)}{\sum_{x'} \text{count}(y_1 = x', y_2 = \alpha, y_4 = \beta)}
\]
Learning with Tied Parameters

- In Markov nets:

  \[ \frac{\partial}{\partial \theta_i} = \phi_i(D) - \mathbb{E}_{\theta_i}(\phi_i) \]

  \[ \frac{\partial}{\partial \theta_i} = \sum_{\phi \in \Phi_{\theta_i}} \phi(D) - \mathbb{E}_{\theta_i} \left( \sum_{\phi \in \Phi_{\theta_i}} \phi \right) \]

\[ \Phi_{\theta_i} : \text{set of factors that share } \theta_i \]

© Getoor & Mihalkova 2010
Representing Tied Factors

- Parameter tying sounds like a great idea
  - ... but how to specify which factors are tied?
  - ... and how to express their general form?
- We need a language that:
  - Allows us to easily specify tied factors
  - Can handle the various kinds of relations
- Good news: There are many options
- We’ll consider two flavors:
  - Dynamic models
  - Richer relational languages
Parameter Tying across Time

- Parameter tying naturally occurs in a temporal model, where we have:
  - variable dependencies within a time slice
  - variable dependencies from one time slice to the next
- For example, in an hidden Markov model (HMM) we need two CPTs: for transitions and for emissions.
- A more general case is the dynamic Bayesian net:

![Dynamic Bayesian Network Diagram]

© Getoor & Mihalkova 2010
Case Study

- Next we consider a case study that models relevance of Web search results as a DBN
- Task: Using query log data, learn a model to predict the probability that a URL is relevant to a query
- Interesting challenge: making the most of the possibly noisy click signal
  - Does a (lack of) click mean (ir)relevance?
  - Decoupling the effect of result position and result relevance
- Based on [Chapelle & Zhang, WWW09]
DBN for Predicting Relevance

- [Chapelle & Zhang, WWW09]
- Dynamic sequence over URL result list

User examined URL?
User clicked on URL? (observed)
User attracted by URL?
User satisfied by URL?
DBN for Predicting Relevance

- [Chapelle & Zhang, WWW09]

$$S_i = 1 \Rightarrow E_{i+1} = 0$$
$$E_i = 0 \Rightarrow E_{i+1} = 0$$
$$P(E_{i+1} = 1 | E_i = 1, S_i = 0) = \gamma$$

$$A_i = 1, E_i = 1 \Leftrightarrow C_i = 1$$

$$P(A_i = 1) = a_u$$

$$P(S_i = 1 | C_i = 1) = s_u$$
$$C_i = 0 \Rightarrow S_i = 0$$

Perceived Relevance

Learnable parameters

User satisfaction given click

© Getoor & Mihalkova 2010
DBN for Predicting Relevance

- Training is done using EM
- The relevance $r_u$ of a URL is defined as the probability that user is satisfied given that she has examined the document:

$$r_u = P(S_i | E_i = 1)$$

$$= P(S_i = 1 | C_i = 1) P(C_i = 1 | E_i = 1)$$

$$= \alpha_u s_u$$

[Chapelle & Zhang, WWW09]
Part II Road Map

- “Flat” Relational Models
- Collective Classification Models
- Richer SRL Models
  - Plan:
    - Start with general representation analogous to factor graphs
    - Discuss analogs of Bayesian nets
    - Discuss analogs of Markov nets
Par-factor Graphs

- Factor graphs with parameterized factors
  - Terminology introduced by [Poole, IJCAI03]
- A par-factor is defined as the triple
  - $\mathcal{A}$: set of parameterized random variables
  - $f$: function that operates on these variables and evaluates to > 0
  - $\mathcal{C}$: set of constraints
- A par-factor graph is a set of par-factors
Parameterized Random Vars

- Can be viewed as a blueprint for manufacturing random variables
- For example:
  - Let $A$ and $B$ be variables, then $A \leftrightarrow B$ is a parameterized random variable.
  - Given specific individuals, we can manufacture random variables from it:
    - $A \leftrightarrow B$
    - $Ann \leftrightarrow Bob$
    - $Ada \leftrightarrow Don$
    - $Xin \leftrightarrow Yan$
    - ...

So far we are not assuming a particular language for expressing par-RVs.
Parameterized Random Vars

- Can be viewed as a blueprint for manufacturing random variables

- For example:
  - Let A and B be variables, then $A \rightarrow B$ is a parameterized random variable.
  - Given specific individuals, we can manufacture random variables:
    - Ann$\rightarrow$Bob
    - Ada$\rightarrow$Don
    - Xin$\rightarrow$Yan
    - ...

So far we are not assuming a particular language for expressing par-RVs.
Constraints

- The constraints in set $C$ govern how par-RVs can be instantiated.
- For example, one constraint for our par-RV could be that $B \neq \text{Don}$.

- With this constraint, the possible instantiations are:
  - $\text{Ann} \leftrightarrow \text{Bob}$
  - $\text{Ada} \leftrightarrow \text{Don}$ (not allowed)
  - $\text{Xin} \leftrightarrow \text{Yan}$
Transitivity Par-factor

\( A = \{ A \leftrightarrow B, B \leftrightarrow C, A \leftrightarrow C \} \)

\( f \) can be defined as before

\( C = \{ A \neq B \neq C \} \)

However, whereas before these referred to the potential friendships of specific individuals, now they refer to variables, i.e. to people in general.
Transitivity Par-factor

\[ A = \{ A \leftrightarrow B, B \leftrightarrow C, A \leftrightarrow C \} \]

- \( f \) can be defined as before

\[ C = \{ A \neq B \neq C \} \]

This means that now we can train on one set of individuals and apply our models to an entirely different set.

However, whereas before these referred to the potential friendships of specific individuals, now they refer to variables, i.e. to people in general.

© Getoor & Mihalkova 2010
To instantiate a par-factor, we need a set of individuals: Ann, Bob, Don.

Then we consider all possible instantiations of the par-RVs with these individuals:

- Ann<->Bob
- Ann<->Don
- Bob<->Don
- Bob<->Ann
- Don<->Bob
- Don<->Ann

... etc.
To instantiate a par-factor, we need a set of individuals:

- Ann, Bob, Don

Then we consider all possible instantiations of the par RVs with these individuals:

- Ann<->Bob
- Ann<->Don
- Bob<->Don
- Bob<->Ann
- Don<->Bob
- Don<->Ann

... etc.

Moral of the story:
So much power can be dangerous!

- Starting with just 3 individuals, we've ended up with a huge and densely connected graph (for n individuals, we would get $O(n^3)$ factors)
- Inference becomes very problematic

© Getoor & Mihalkova 2010
Managing our Power

Constraints

- One way of keeping the factor graph size manageable is by imposing appropriate constraints on permitted instantiations.

Par-factor size

- More par-RVs per par-factor translate into more RVs per factor.

When defining a par-factor, it is important to think:

- How many instantiations will this par-factor have?
- How many RVs per instantiation?

This is easier said than done

- Will discuss more
Part II Recap So Far

- Extended factor graphs to allow for convenient parameter tying
  - Parameter learning: an extension of parameter learning in Bayesian/Markov nets
  - Inference: instantiate the par-factors and perform inference as before

- Are we done?
  - We still do not have a convenient language for specifying the function part of a par-factor
  - A wide range of languages have been introduced and studied in the field of statistical relational learning (SRL). Here we review just a few
Road Map

Factor Graphs

Bayesian Nets

Markov Nets

Par-factor Graphs

Directed Models
- BLPs [Kersting & De Raedt, ILP01]
- PRMs [Koller & Pfeffer, AAAI98]
etc.

Undirected Models
- RMNs [Taskar et al., UAI02]
- MLNs [Richardson & Domingos, MLJ06]
etc.

Hybrid Models
- RDNs [Neville & Jensen, JMLR07]
Directed Models

- Bayesian logic programs (BLPs)
  - Based on first-order logic
  - [Kersting & De Raedt, ILP01]

- Probabilistic relational models (PRMs)
  - Using an object-oriented, frame-based representation
  - [Koller & Pfeffer, AAAI98]
Relational Schema

- **Author**
  - Good Writer
  - Smart

- **Review**
  - Mood
  - Length

- **Paper**
  - Quality
  - Accepted

- Author of

- Has Review

- Describes the types of objects and relations in the database
Probabilistic Relational Model

Author
- Smart
- Good Writer

Review
- Mood
- Length

Paper
- Quality
- Accepted

\[
\]

© Getoor & Mihalkova 2010
Probabilistic Relational Model

Author

- Smart
- Good Writer

Paper

- Quality
- Accepted

Review

- Mood
- Length

| Q, M | P(A | Q, M) |
|------|-----------|
| f, f | 0.1 0.9   |
| f, t | 0.2 0.8   |
| t, f | 0.6 0.4   |
| t, t | 0.7 0.3   |

© Getoor & Mihalkova 2010
Fixed relational skeleton $\sigma$:  
- set of objects in each class  
- relations between them
PRM defines distribution over instantiations of attributes

© Getoor & Mihalkova 2010
### A Portion of the BN

| Q, M | P(A | Q, M) |
|------|-----------|
| f, f | 0.1       |
|       | 0.9       |
| f, t | 0.2       |
|       | 0.8       |
| t, f | 0.6       |
|       | 0.4       |
| t, t | 0.7       |
|       | 0.3       |

© Getoor & Mihalkova 2010
A Portion of the BN

P2.Accepted

Q   M   |   P(A | Q, M) |
-----|----------|
| f, f | 0.1 0.9  |
| f, t | 0.2 0.8  |
| t, f | 0.6 0.4  |
| t, t | 0.7 0.3  |

P3.Accepted

Pissy

Low

High

© Getoor & Mihalkova 2010
PRM: Aggregate Dependencies

Paper

- Quality
- Accepted

Review

- Mood
- Length

Review R1
- Mood

Review R2
- Mood

Review R3
- Mood
- Length

Paper P1
- Quality
- Accepted

© Getoor & Mihalkova 2010
### PRM: Aggregate Dependencies

#### Paper
- Quality
- Accepted

#### Review
- Mood
- Length

#### Table: $P(A | Q, M)$

| $Q, M$ | $P(A | Q, M)$ |
|--------|---------------|
| $f, f$ | 0.1 0.9       |
| $f, t$ | 0.2 0.8       |
| $t, f$ | 0.6 0.4       |
| $t, t$ | 0.7 0.3       |

#### Diagram:

- **Paper P1**
  - Quality
  - Accepted

- **Review R1**
  - Mood
  - Length

- **Review R2**
  - Mood
  - Length

- **Review R3**
  - Mood
  - Length

#### Functions:
- sum, min, max, avg, mode, count

---

© Getoor & Mihalkova 2010
PRM Semantics

PRM + relational skeleton $\sigma$ =

probability distribution over completions $l$:

$$P(l | \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A | \text{parents}_{S,\sigma}(x.A))$$

Objects Attributes

© Getoor & Mihalkova 2010
Learning PRMs

- Parameter estimation
- Structure selection

© Getoor & Mihalkova 2010
θ* = \frac{N_{P, \overline{Q}, R, \overline{M}, P, A}}{N_{P, \overline{Q}, R, \overline{M}}}

where \( N_{P, \overline{Q}, R, \overline{M}, P, A} \) is the number of accepted, low quality papers whose reviewer was in a poor mood.
ML Parameter Estimation

Review
Mood
Length

Paper
Quality
Accepted

\[ \theta^* = \frac{N_{P, \overline{Q}, R, \overline{M}, P, A}}{N_{P, \overline{Q}, R, M}} \]

Query for counts:

Count \( \left\{ \pi \right\} \) P.Quality
P.Accepted

© Getoor & Mihalkova 2010
Structure Selection

Idea:
- define scoring function
- do local search over legal structures

Key Components:
- legal models
- scoring models
- searching model space
Structure Selection

Idea:
- define scoring function
- do local search over legal structures

Key Components:
- legal models
- scoring models
- searching model space
PRM defines a coherent probability model over a skeleton $\sigma$ if the dependencies between object attributes is acyclic.

How do we guarantee that a PRM is acyclic for every skeleton?
Attribute Stratification

PRM dependency structure $S$

$\text{Paper.Accepted} \quad \text{if } \text{Researcher.Reputation depends directly on } \text{Paper.Accepted}$

Attribute stratification:

dependency graph acyclic $\Rightarrow$ acyclic for any $\sigma$

Algorithm more flexible; allows certain cycles along guaranteed acyclic relations

© Getoor & Mihalkova 2010
Structure Selection

- Idea:
  - define scoring function
  - do local search over legal structures

- Key Components:
  - legal models
  - scoring models – same as BN
  - searching model space
Structure Selection

Idea:
- define scoring function
- do local search over legal structures

Key Components:
- legal models
- scoring models
  » searching model space
Phase 0: consider only dependencies within a class

Potential-Parents(R.A) = \bigcup \{R.B\mid R.B \in \text{descriptive-attributes}(R)\}

- Add A.S \rightarrow A.W
- Delete R.M \rightarrow R.L
Phase 1: consider dependencies from “neighboring classes, via schema relations

Potential-Parents(R.A) = \bigcup S.C

S.C \in descriptive-attributes(R\triangleright S)
Phase 2: consider dependencies from “further” classes, via relation chains

Potential-Parents\((R.A) = \bigcup T.D\)

\(T.D\in\text{descriptive-attributes}(R \rightarrow S \rightarrow T)\)

© Getoor & Mihalkova 2010
Road Map

Factor Graphs

Bayesian Nets

Markov Nets

Par-factor Graphs

Directed Models

BLPs [Kersting & De Raedt, ILP01]
PRMs [Koller & Pfeffer, AAAI98]
etc.

Undirected Models

RMNs [Taskar et al., UAI02]
MLNs [Richardson & Domingos, MLJ06]
etc.

Hybrid Models

RDNs [Neville & Jensen, JMLR07]
Undirected Models

- Relational Markov networks
  - Using database query language (SQL)
  - [Taskar et al., UAI02]

- Markov logic networks
  - Use first-order logic
  - [Richardson & Domingos, MLJ06]

- Both define a Markov network over relational data
Relational Markov Networks

- Par-factors are defined using SQL statements
  - Essentially selecting the relational tuples that should be connected in a clique

A par-factor consists of:

Set of parameterized RVs

Constraints on instantiations

Function operating on the RVs

select doc1.Category, doc2.Category
from Doc doc1, Doc doc2, Link link
where link.from = doc1.Key and link.To = doc2.Key

[Taskar et al., UAI02]
Relational Markov Networks

- Par-factors are defined using SQL statements
  - Essentially selecting the relational tuples that should be connected in a clique

A par-factor consists of:

Set of parameterized RVs

Constraints on instantiations

Function operating on the RVs

The function $\phi$ is defined as a potential function over the selected tuples and has the form:

$$\phi(A) = \exp(w \cdot f(A))$$
Unrolling the RMN

All use the same $\phi$ and the same parameterization
Par-factors are defined using first-order logic statements

\[
\text{hyperlink}(D_1, D_2) \Rightarrow \text{category}(D_1, C) \land \text{category}(D_2, C)
\]

The predicates whose values are known during inference can be seen as constraining the cliques that are constructed over the unknown ones.

\[ A = \{ \text{category}(D_1, C), \text{category}(D_2, C) \} \]
Par-factors are defined using first-order logic statements.

\[ \phi = \exp(w \cdot TruthValue(f(A))) \]
MLNs Unrolling & Joint Distribution

\[ P(X = x) = \frac{\exp \left( \sum_{f_i \in \mathcal{F}} w_i n_i(x) \right)}{Z} \]

For each formula in the MLN

Possible world

Number of satisfied instantiations

© Getoor & Mihalkova 2010
Next we consider an application of Markov logic to web query disambiguation.

Based on [Mihalkova & Mooney, ECML09]
Problem: Given an ambiguous query, determine which URLs more likely reflect user interest

[Mihalkova & Mooney, ECML09] considered a constrained setting in which very little was known about previous user browsing history
  - About 3 previous searches on average
Relationships

Active Session:

Historical Sessions:

[Mihalkova & Mooney, ECML09]

© Getoor & Mihalkova 2010
Clauses

- Collaborative: User will click on result chosen in sessions related by:
  - Shared click
  - Shared keyword click-to-click, click-to-search, search-to-click, or search-to-search
  - e.g.,

\[
\text{SharesClick}(S,D) \land \text{ChoseResult}(S,R,Q) \Rightarrow \text{ClickOn}(R,Q)
\]

- Popularity: User will choose result chosen by any previous session, regardless of whether it is related
Clauses Continued [Mihalkova & Mooney, ECML09]

- Local: User will choose result that shares keyword with previous search or click in current session
  - Didn’t find it to be effective because of brevity of sessions
- If the user chooses one of the results, she will not choose another
  - Sets up a competition among possible results
  - Allows the same set of weights to work well for different-size problems

\[
\text{ClickOn}(R_1, Q) \land R_1 \neq R_2 \Rightarrow \neg \text{ClickOn}(R_2, Q)
\]
Let’s see how these rules define a factor graph
Will do this for
- a single query Q and
- a set of possible results $R_1$, $R_2$, and $R_3$ for it

1. Set up decision nodes.
   We have one for each grounding of the unknown predicate ClickOn
Let’s see how these rules define a factor graph.

Will do this for:
- a single query Q and
- a set of possible results $R_1$, $R_2$, and $R_3$ for it.

2. Ground out each clause and construct factors corresponding to groundings.

- $\text{clickOn}(R_1, Q)$
- $\text{clickOn}(R_2, Q)$
- $\text{clickOn}(R_3, Q)$
Let’s see how these rules define a factor graph.

Will do this for:
- a single query Q and
- a set of possible results $R_1$, $R_2$, and $R_3$ for it.

\[ \text{SharesClick}(S, D) \land \text{ChoseResult}(S, R, Q) \Rightarrow \text{ClickOn}(R, Q) \]
Let’s see how these rules define a factor graph.
Will do this for:
- a single query \( Q \) and
- a set of possible results \( R_1, R_2, \) and \( R_3 \) for it

\[
\text{clickOn}(R_1, Q) \quad \text{clickOn}(R_2, Q) \quad \text{clickOn}(R_3, Q)
\]

Note: So far, what we have is a flat relational model: no connection between the decision nodes!

\[
\text{SharesClick}(S, D) \land \text{ChooseResult}(S, R, Q) \Rightarrow \text{ClickOn}(R, Q)
\]

Total number equals number of sessions that share a click with current one.
Let’s see how these rules define a factor graph

Will do this for:
- a single query Q and
- a set of possible results $R_1$, $R_2$, and $R_3$ for it

\[
\text{ClickOn}(R_1, Q) \land R_1 \neq R_2 \Rightarrow \neg \text{ClickOn}(R_2, Q)
\]
Let's see how these rules define a factor graph.

Will do this for:

- a single query \( Q \)
- a set of possible results \( R_1, R_2, \) and \( R_3 \)

This demonstrates an advantage of using a richer statistical relational representation:

Making our model collective was as easy as adding a rule!

\[
\text{ClickOn}(R1, Q) \land R1 \neq R2 \Rightarrow \neg \text{ClickOn}(R2, Q)
\]
MLN Structure Learning

- Will discuss according to several dimensions:
  - Transfer/Scratch
    - Transfer: Algorithm requires previously learned model
    - Scratch: Algorithm does not make use transferred model
  - Discriminative?
    - Algorithm requires knowledge about test attributes/relations
  - Top-Down/Bottom-Up
    - Top-Down: Algorithm largely follows a generate-and-test strategy
    - Bottom-Up: Algorithm uses a data-driven procedure to generate candidate structures
MLN Structure Learning

- Blank field means that algorithm is agnostic to a dimension or can be adapted either way.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Scratch/Transfer</th>
<th>Discriminative?</th>
<th>Top-Down/Bottom-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Kok &amp; Domingos, ICML05]</td>
<td></td>
<td></td>
<td>Top-Down</td>
</tr>
<tr>
<td>[Mihalkova &amp; Mooney, ICML07]</td>
<td>Scratch</td>
<td></td>
<td>Bottom-Up</td>
</tr>
<tr>
<td>[Mihalkova &amp; Mooney, AAAI07]</td>
<td>Transfer</td>
<td></td>
<td>Bottom-Up</td>
</tr>
<tr>
<td>[Huynh &amp; Mooney, ICML08]</td>
<td>Scratch</td>
<td>Yes</td>
<td>Bottom-Up</td>
</tr>
<tr>
<td>[Biba et al., ILP08]</td>
<td></td>
<td>Yes</td>
<td>Top-Down</td>
</tr>
<tr>
<td>[Davis &amp; Domingos, ICML09]</td>
<td>Transfer</td>
<td></td>
<td>Top-Down</td>
</tr>
<tr>
<td>[Kok &amp; Domingos, ICML09]</td>
<td>Scratch</td>
<td></td>
<td>Bottom-Up</td>
</tr>
<tr>
<td>[Kok &amp; Domingos, ICML10]</td>
<td>Scratch</td>
<td></td>
<td>Bottom-Up</td>
</tr>
<tr>
<td>[Khorsavi et al., AAAI10]</td>
<td>Scratch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bottom-Up Idea Nuggets

- **BUSL** [Mihalkova & Mooney, ICML07]
  - Observation: An MLN defines a Markov net, so why not:
    - Start from learning a Markov network template
    - Constrain structure search to this template

- **Algorithm of** [Huynh & Mooney, ICML08]
  - Use a bottom-up ILP learner (ALEPH) to induce a set of clauses

- **LHL** [Kok & Domingos, ICML09]
  - Observation: Relational pathfinding is very effective in finding long-range clauses, but can blow up:
    - Cluster the entities in the domain
    - Perform relational pathfinding in the clustered graph

- **LSM** [Kok & Domingos, ICML10]
  - Want to learn even longer-range dependencies
    - Perform a random walk on the relational graph to find well-treaded patterns (structural motifs)
    - Constrain structure search to within motifs
## Directed vs Undirected Models

<table>
<thead>
<tr>
<th></th>
<th>Directed</th>
<th>Undirected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation</strong></td>
<td>• Need to define combining rules</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter Learning</strong></td>
<td>• Amounts to counting</td>
<td>• Cannot compute in closed form</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requires running inference</td>
</tr>
<tr>
<td><strong>Structure Learning</strong></td>
<td>• Parameters updated only where structure changed</td>
<td>• Parameters updated globally</td>
</tr>
<tr>
<td></td>
<td>• Need to maintain acyclicity</td>
<td></td>
</tr>
<tr>
<td><strong>Inference</strong></td>
<td></td>
<td>• Need to compute normalizing function</td>
</tr>
</tbody>
</table>

© Getoor & Mihalkova 2010
Hybrid Models

- Hybrid models aim at combining the advantages of directed and undirected ones, while avoiding the disadvantages

- Next we briefly introduce relational dependency networks (RDNs) [Neville & Jensen, JMLR07]
Relational Dependency Networks

- An extension of dependency networks [Heckerman et al., JMLR00] to relational domains

- In a dependency network:
  - As in Markov nets, one’s neighbors render it independent of all other variables
    - No need to worry about maintaining acyclicity
  - As in Bayesian nets, potential functions are represented as conditional probability tables (CPTs)
    - No normalization necessary

- RDN “lift” DNs to relational domains:
  - Dependencies are described for parameterized RVs
  - Upon instantiation, RDNs define a DN in which CPTs are shared
Part II Summary

- Started with Flat Relational models
- Moved to collective classification models
  - ICA: learning and inference
  - Factor graphs: learning and inference
- Extended factor graphs to allow for parameter tying
- Considered several flavors of languages for specifying par-factor graphs
- So we are done?
  - Not nearly