Exploiting Statistical and Relational Information on the Web and in Social Media

Lise Getoor & Lily Mihalkova

Tutorial at SDM-2011

Slides available at: http://lb.vg/0jt9Z
Statistical Relational Learning and the Web

Challenges Addressed by SR Learning and Inference

- Multi-relational data
  - Entities can be of different types
  - Entities can participate in a variety of relationships
- Probabilistic reasoning under noise and/or uncertainty

Challenges Arising in Web Applications

- Entities of different types
  - E.g., users, URLs, queries
- Entities participate in variety of relations
  - E.g., click-on, search-for, link-to, is-refinement-of
- Noisy, sparse observations

© Getoor & Mihalkova 2010-2011
Tutorial Goals

- Understand the interactions between SRL and Web/social media applications:
  - What are some sources of relational and statistical information on the Web/social media?
  - What are the basic SRL methods and techniques?
  - To what extent are existing SRL techniques a good fit for the challenges arising on the Web?
  - What future developments would make these areas more closely integrated?
Tutorial Road Map

- **Introduction**: Brief survey of statistical and relational info on the Web and in social media

- **Main**: Survey of SRL Models & Techniques
  - Relational Classifiers
  - Collective Classification
  - Advanced SRL models

- **Conclusion**: Looking Ahead
Disclaimer

- Not an attempt to provide a complete survey of the Web, social media, or SRL literatures
  - 3 hours is not enough for this!

- We provide a biased view, motivated by our goal of identifying the interesting intersection points of SRL and Web/social media applications
Relational Info on the Web

- Search engine log applications
  - Sessionization, clustering/refining queries, query personalization/disambiguation, click models, predicting commercial intent, query advertisement matching, many others

- Social networks/social media applications
  - Finding important nodes/influentials, understanding social roles/collaborative dynamics, viral marketing & information flow, link recommendation, community discovery
Sessionization

- Two kinds of sessions:
  - Search session
    - Determined using time-outs
  - Logical session
    - The same search session may contain queries for more than one information-seeking intent or search mission
    - Logical sessions may:
      - straddle search sessions
      - be intertwined

- Goal: Use query logs to determine whether two queries are part of the same logical session

- Following example is based on [Boldi et al., CIKM08] and [Jones & Klinkner, CIKM08]
Sessionization

Features Derived From:
- Clicked-For
- Shares-Words
- Same-Session
- Precedes-In-Session
- Precedes-Temporally

Used to Learn to Predict:
- Precedes-In-Logical-Session
- Same-Logical-Session

Weight indicates frequency with which one query follows another.

© Getoor & Mihalkova 2010-2011
Sessionization: Features

- Relations are typically not used directly; rather features are defined over them.

- Clicked-For
- Shares-Words
- Same-Session
- Precedes-In-Session
- Precedes-Temporally

Word/character similarity, such as:
- Number of common words/characters
- Cosine, Jaccard similarity
- Character edit distance

© Getoor & Mihalkova 2010-2011
Sessionization: Features

- Relations are typically not used directly; rather features are defined over them.

For example:
- Number of sessions in which co-occur
- Variety of stats over co-occurrence sessions, e.g. average length, average position of queries
- Statistical test indicating significance of co-occurrence

© Getoor & Mihalkova 2010-2011
Sessionization: Features

- Relations are not used directly; rather, features are defined over them.

Examples:
- Average time between queries
- Time between queries > threshold
Click Models/Ranking

- The quintessential search engine problem
- Predict the probability that a URL is clicked, given that it is shown for a given query
- URLs with higher probability of being clicked/relevant ranked higher
- Lots of different approaches developed
  - Here we focus on some that use query log data to infer relevance
Click Models/Ranking

[Joachims et al., SIGIR05]
[Radlinski & Joachims, KDD05]
[Agichtein et al., SIGIR06]
[Bilenko & White, WWW08]
[Chapelle & Zhang, WWW09]
[Guo et al., WWW09]

© Getoor & Mihalkova 2010-2011
Click Models/Ranking

- Much research has focused on defining informative features on these relations
  - e.g., a small sample from [Agichtein et al., SIGIR06]

Share-Terms
- Features to relate a query to a URL:
  - Term overlap between query and page title
  - Term overlap between query and URL
  - Term overlap between query and page summary

Clicked-For
- Features describing click-for relation:
  - Position of clicked URL on the page of results
  - Relative frequency of a click
  - Is previous/next result clicked?
Click Models/Ranking

- The pattern of clicking or skipping a search result has been used to infer relevance of URLs
- e.g., some examples from [Joachims et al. SIGIR05; Radlinski & Joachims, KDD05]

© Getoor & Mihalkova 2010-2011
Personalized Search

- Can also include information about users, their searches and their information needs
Summary of Query Logs Apps

- Clicked-For
- Shares-Terms
- Shares-Words
- Same-Topic
- Hyperlink
- Same-Session
- Precedes-In-Session
- Subset-URLs
- Identical-URLs
- Partial-Overlap-URLs
- Precedes-Temporally
- Prec-In-Logical-Sess.
- Same-Logical-Session
- Is-Represented-By
- Fulfills-Info-Need
- Targets-Info-Need
- Have-Info-Need
- Search-For
- Search-For-&-Click
- Similar Users
- More-Relevant-Than
Relational Info in Social Media

© Getoor & Mihalkova 2010-2011
Online Social Networks

Friends

Collaborators

Family

Fan/Follower

Comments, Replies, Edits, Co-Edits, Co-Mentions, etc.
[Agrawal et al., WWW03]: Use the fact that this is typically an antagonistic relationship to infer separation into opposing camps
Nuanced Collaborations

[Brandes et al., WWW09]: Studied editor interactions in Wikipedia
Polarity, Trust/Distrust

Variants of transitivity:

- If you are my friend, my enemies, are your enemies
- If we are enemies, your enemies are my friends

[Kunegis et al., WWW09]: Which are the unpopular users? What is the sign of a relationship between users?

[Guha et al., WWW04]: How does trust and distrust propagate?

[Leskovec et al., WWW10]: What are user attitudes (sign of relationship) toward one another?
Affiliation Networks

G_1 \rightarrow U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9

G_2 \rightarrow U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9

G_3 \rightarrow U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9

G_4 \rightarrow U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9

Belongs-To-Group
[Singla & Richardson WWW08]: Similarities between querying behavior and talking to each other or having friend in common.
Social Tagging, View 1

- Ternary relationships between tags, users, documents
Social Tagging, View 2

- Tri-partite graph
  - Aggregate over documents/tags

Document recommendations are based on not just preferences of similar users but also preferences for tags.

[Shepitsen et al., RS08] [Guan et al., WWW10]

© Getoor & Mihalkova 2010-2011
Summary of Social Media Relationships

User-User
- Friends
- Collaborators
- Family
- Fan/Follower
- Replies
- Co-Edits
- Co-Mentions, etc.

User-Doc
- Comments
- Edits, etc.

User-Group

User-Query-Click

User-Tag-Doc
SURVEY OF SRL MODELS & TECHNIQUES
Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
Road Map

- Relational Classifiers
  - Definition
  - Case Studies
  - Key Idea: Relational Feature Construction

- Collective Classification

- Advanced SRL Models
Relational Classifiers

Given:

Task: Predict attribute of some of the entities

Alternate task: Predict existence of relationship between entities

© Getoor & Mihalkova 2010-2011
Relational Classifiers

- 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 relational classifiers

- 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

BT2

BT3

Ad1

Ad2

Ad3

Ad4

Ad5

Ad6

contains-bid-term

contains-bid-term (according to search engine)

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

queried-bid-term

Count

Count

Average CTR

Average CTR

© Getoor & Mihalkova 2010-2011
Case Study 2: Predicting Friendships

- Task: Predict new friendships among users, based on their descriptive attributes, their existing friendships, and their family ties.

- Our description is based on approach by
  - [Zheleva et al., SNAKDD08]
Relational Features Used

- “Petworks” - social networks of pets
Key Idea: Feature Construction

- Feature informativeness is key to the success of a relational classifier

- Next we provide a systematic review of relational feature construction
  - Global measures
  - Node-specific measures
  - Node pair measures

- These will be useful also for collective classifiers and other SRL models
Global Measures

- Summarize properties of entire graph (or subgraph)

- Next we discuss:
  - Graph Cohesion
  - Clustering coefficient
  - Bipolarity

- Many others possible…
Graph Cohesion

- Density (% of possible edges)
- Average Degree
- Average Tie Strength
- Max flow
- Size of largest clique
- Average geodesic distance
- Diameter (max distance)
- F Measure - proportion of pairs of nodes that are unreachable from each other

- Many others….

[Everett & Borgatti, 1999]
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

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

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

[Watts & Strogatz, Nature98]

© Getoor & Mihalkova 2010-2011
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

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

Max Cut

Value between -1 and +1

[Brandes et al, WWW09]

© Getoor & Mihalkova 2010-2011
Node-specific Measures

- Summarize properties of node

- Next we discuss:
  - Attribute aggregates
  - Structural measures
Attribute Aggregates: Level 1
Based on [Perlich & Provost, KDD03]

- No aggregation necessary
  - Use an attribute of the entity about which a prediction is made
  - Relationships to other entities are not used
- Example: Predicting the political affiliation of a social network user can be based on whether user opposes a tax raise
Attribute Aggregates: Level 2

- Aggregation over independent attributes of related entities
  - Values at related entities are considered independently of one another

- Example:

  - What is this user's political affiliation?
  - Number of friends who oppose a tax raise
Attribute Aggregates: Level 3

- Aggregation over dependent attributes of related entities
  - Values at related entities need to be considered together as a set

- Example:

What is this user's political affiliation?

Trend of friendships to people who oppose a tax raise made over time

© Getoor & Mihalkova 2010-2011
Attribute Aggregates: Level 4
Based on [Perlich & Provost, KDD03]

- Level 4: Aggregation over dependent attributes across multiple relations
  - Aggregate computed over multiple “hops” across relational graph
  - Values need to be considered together

- Example:
  - What is this user’s political affiliation?
  - Trend of friendships made over time to liberal users that are members of the same groups as $U_1$
Representing Attribute Aggregates with First-Order Logic

Defining Boolean-valued features using FOL

- A feature that checks if $U_1$ has a liberal friend who shares group membership:

$$\exists u: \text{friends}(U_1, u) \land \text{inGroup}(U_1, g) \land \text{inGroup}(u, g) \land \text{liberal}(u)$$

Augmenting FOL with arbitrary aggregation functions

- A feature that counts the number of such friends

$$\text{Count}(u): \text{friends}(U_1, u) \land \text{inGroup}(U_1, g) \land \text{inGroup}(u, g) \land \text{liberal}(u)$$

Advantage: Can represent arbitrary chains of relations
Disadvantage: Numerical values are cumbersome
Numeric Aggregations

- Features based on frequently occurring values
  - Most common value
  - Most common value in positive/negative training examples

- Value whose frequency of occurring differs the most in positive vs negative examples

- Features based on vector distances
  - Difference in distribution over values

Based on [Perlich & Provost, KDD03]
Structural Measures

- Cohesion
  - $CC(v)$ – clustering coefficient at a node
  - Stability - valence of triads: +++, --- are stable; +-+ instable

- Centrality
  - Degree centrality
  - Betweenness centrality
  - Eigenvalue centrality (a.k.a. PageRank)

- For more, see [Wasserman & Faust, 94]
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.

$$BC(a) = \sum_{j<k} \frac{|SP^{\rightarrow a}(j, k)|}{|SP(j, k)|}$$

- Number of shortest paths between $j$ and $k$ that go through $a$
- Total number of shortest paths between $j$ and $k$
Node-Pair Measures

- Summarize properties of (potential) edges

Next we discuss:
- Attribute-based measures
- Edge-based measures
- Neighborhood similarity measures
Attribute Similarity Measures

- Measures defined on pairs of nodes

- Attribute similarity measures to compare nodes based on their attributes’
  - String similarity
  - Hamming distance
  - Cosine
  - etc.

- Component similarities are features for relational classifier*

*or overall attribute similarity based on some weighted combination of components and simple threshold is applied
Edge-Based Measures

- Edges can be of different types, corresponding to different kinds of relationships
  - Edges of one type can be predictive of edges of another type, e.g., working together is predictive of friendship
- Edges can be weighted or have other associated attributes to indicate the strength, or other qualities, of a relationship
  - E.g., the thickness of an edge between two users indicates frequency of exchanged emails
Structural Similarity Measures

- Set similarity measures to compare nodes based on set of related nodes, e.g., compare neighborhoods

- Examples:
  - Average similarity between set members
  - Jaccard coefficient
  - Preferential attachment score
  - Adamic/Adar measure
  - SimRank
  - Katz score

- For more details, see [Liben-Nowell & Kleinberg, JASIST07]
Jaccard Coefficient

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

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

[Adamic & Adar, SN03]
SimRank

“Two objects are similar if they are related to similar objects”

Defined as the unique solution to:

Decay factor between 0 and 1

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

Computed by iterating to convergence

Initialization to \( s(a, b) = 1 \) if \( a=b \) and 0 otherwise

[Jeh & Widom, KDD02]
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

- Since expensive to compute, often use approximate Katz, assuming some max path length of k
Relational Classifiers: 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
Relational Classifiers: Cons

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

CTR?

Ad

contains-bid-term

BT\_1 \quad BT\_2 \quad BT\_3

Average CTR

Ad\_1 \quad Ad\_2

CTRs of these ads have to be observed

Average CTR

Ad\_5 \quad Ad\_6

BT\_4 \quad BT\_5 \quad BT\_6
If $P_1$ and $P_2$ become friends, $P_7$ and $P_{11}$ are likely to also become friends.
Relational Classifiers: Cons

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

... 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
Relational Classifiers: Cons

- Relational features cannot be based on attributes or relations that are being predicted.

- 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.
Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
Road Map

- Relational Classifiers

- Collective Classification
  - Definition
  - Case Studies
  - Key Idea: Iteration / Propagation

- Advanced SRL Models
Collective Classification

- Disadvantages of relational classifiers can be addressed by making collective predictions
  - Can help correct errors
  - Can coordinate assignments to satisfy constraints
- Variety of algorithms
  - Iterated conditional modes [Besag 1986; …]
  - Relaxation labeling [Rosenfeld et al. 1976; …]
- Make coherent joint assignments by iterating over individual decision points, changing them based on current assignment to related decision points
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: 

Learn models (local and relational) from fully labeled training set
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 collective classification

- Variations:
  - Propagate probabilities, rather than mode (see also Gibbs Sampling later)
  - Batch vs. Incremental updates
  - Ordering strategies

- Related Work:
  - Cautious Inference [McDowell et al., JMLR09]
  - Weighted neighbor [Macskassy, AAAI07]
  - Active Learning [Bilgic et al., TKDD09, ICML10]
Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models
  - Key Ideas: Par-factor graphs
  - Languages
Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models
  - Key Ideas: Par-factor graphs
  - Languages
Factor Graphs

- Bipartite graph containing two kinds of nodes
  - variables $y_1$
  - factors $\square, \blacksquare$: strictly positive functions of the variables to which they are connected in the graph
The probability of a joint assignment of values $y$ to the set of variables $Y$ is computed as:

$$
P(Y = y) = \frac{\prod_{f \in \text{Factors}} f(y_{\{f\}})}{Z}
$$

- Normalizing constant
- Subset of variables that participate in the computation of $f$
Factor Graphs: Example

- Each orange square represents \( \exp(\theta_L \cdot f_i(y_i)) \)
- Each black 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.
Markov nets (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.
Markov Nets Continued

- 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 networks

The probability of a joint assignment of values $y$ to the set of variables $Y$ is computed as:

$$P(Y = y) = \frac{\prod_{C \in \text{Cliques}} \phi(y_C)}{Z}$$

Normalized constant

Variables in clique $C$
Markov networks

... assuming the log-linear representation for the clique potentials:

$$\phi(y_C) = \exp(\theta_C f(y_C))$$

The joint probability becomes:

$$P(Y = y) = \frac{\exp(\sum_{C \in \text{Cliques}} \theta_C f(y_C))}{Z}$$
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

We often draw 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
Quick Aside: Two Kinds of Graphs

In Part I we were drawing social networks like this:

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

\[ y_3 = (A \leftrightarrow C) \]

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

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

\[ y_1 = (A \leftrightarrow B), y_2 = (B \leftrightarrow C), y_3 = (A \leftrightarrow C) \]

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.

© Getoor & Mihalkova 2010-2011
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

<table>
<thead>
<tr>
<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>$\sim e^0$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\sim e^0$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\times e^{-\theta}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\times e^{-\theta}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\sim e^0$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\times e^{-\theta}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\times e^{-\theta}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\checkmark e^\theta$</td>
</tr>
</tbody>
</table>

... one possibility

© Getoor & Mihalkova 2010-2011
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>$\times e^{-\theta}$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\checkmark e^\theta$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\checkmark e^\theta$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\checkmark e^0$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\checkmark e^\theta$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\checkmark e^0$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\checkmark e^0$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\checkmark e^0$</td>
</tr>
</tbody>
</table>

© Getoor & Mihalkova 2010-2011
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.
Bayesian Nets

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!
Bayesian Nets

- The probability of a joint assignment of values $y$ to the set of variables $Y$ is computed as:

$$P(Y = y) = \prod_{y_i \in \text{Nodes}} P(y_i | Pa(y_i))$$
Inference in Graphical Models

- Two flavors:
  - Computing marginal probabilities of the variables
  - Finding the most likely joint assignment to the variables

- Variety of algorithms exist:
  - Variable elimination
  - Message passing, e.g., belief propagation
  - Sampling, e.g., Gibbs sampling
  - (Integer) linear programming

- Next we show a quick overview of Gibbs sampling
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
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 \(x\)

© Getoor & Mihalkova 2010-2011
Gibbs Sampling

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
Gibbs Sampling vs ICA

- 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>End Goal</td>
<td>Find most likely joint assignment to vars</td>
<td>Compute posterior marginal probabilities for vars</td>
</tr>
<tr>
<td>In each iteration…</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>Graph over which it is performed</td>
<td>Relational graph</td>
<td>Factor graph</td>
</tr>
<tr>
<td>Initialization</td>
<td>Bootstrap from local features</td>
<td>Random/MAP state + burn-in</td>
</tr>
</tbody>
</table>

© Getoor & Mihalkova 2010-2011
Learning in Graphical Models

- Learning the parameters
  - In a Markov network with a log-linear representation of the potential functions:
    \[ \exp(\theta f(y_f)) \]
  - In a Bayesian network, the graph is given, want to learn conditional probability distributions for each child given parents

- Learning the structure
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

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

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

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models
  - Key Ideas: Par-factor graphs
  - Languages
Par-factor Graphs

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

Explanation coming up
Parameterized Random Vars

- Can be viewed as a blueprint for manufacturing random variables

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

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 $\lambda$ is a parameterized random variable.
  - Given specific individuals, we can manufacture random variables:
    - $\text{Ann} \rightarrow \text{Bob}$
    - $\text{Ada} \rightarrow \text{Don}$
    - $\text{Xin} \rightarrow \text{Yan}$
    - ...

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:
  
  - $\text{Ann} \leftrightarrow \text{Bob}$
  - $\text{Ada} \leftrightarrow \text{Don}$ (crossed out)
  - $\text{Xin} \leftrightarrow \text{Yan}$
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
- \( C = \{ A \neq B \neq C \} \)

However, whereas before these referred to the potential friendships of specific individuals, now they refer to variables, i.e. to people in general. This means that now we can train on one set of individuals and apply our models to an entirely different set.
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:
To instantiate a par-factor, we need a set of individuals. 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... etc.
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
Transitivity Par-Factor Instantiated

- 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.
```
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

- Parameter learning can be easily extended to learn with tied parameters
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
SRL 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]
Describes the types of objects and relations in the database
Probabilistic Relational Model

- Author
  - Smart
  - Good Writer

- Review
  - Mood
  - Length

- Paper
  - Quality
  - Accepted
Probabilistic Relational Model

\[
P \left( \text{Paper.Accepted} \mid \text{Paper.Quality, Paper.Review.Mood} \right)
\]
Probabilistic Relational Model

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

© Getoor & Mihalkova 2010-2011
Fixed relational skeleton $\sigma$:
- set of objects in each class
- relations between them
PRM defines distribution over instantiations of attributes
A Portion of the BN

\[ P(A | Q, M) \]

| Q, M | P(A | Q, M) |
|------|------------|
| f, f | 0.1        |
| f, t | 0.2        |
| t, f | 0.6        |
| t, t | 0.7        |
A Portion of the BN

\[
P(A | Q, M)
\]

| 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    |
PRM: Aggregate Dependencies

Paper

- Quality
- Accepted

Review

- Mood
- Length

Review R1

- Mood

Review R2

- Mood
- Length

Review R3

- Mood
- Length

Paper P1

- Quality
- Accepted

© Getoor & Mihalkova 2010-2011
PRM: Aggregate Dependencies

| 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   |

sum, min, max, avg, mode, count
PRM Semantics

PRM + relational skeleton $\sigma$ =

probability distribution over completions $I$:

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

Objects Attributes

© Getoor & Mihalkova 2010-2011
Learning PRMs

- Parameter estimation
- Structure selection
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.

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

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

Query for counts:

Count \[ \pi \]

\[ \begin{align*}
&\text{P.Quality} \\
&\text{R.Mood} \\
&\text{P.Accepted}
\end{align*} \]

© Getoor & Mihalkova 2010-2011
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
Legal Models

- 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 $S$

dependency graph

$Paper.Accepted$

if $Researcher.Reputation$ depends directly on $Paper.Accepted$

$Researcher.Reputation$

Attribute stratification:
dependency graph acyclic $\Rightarrow$ acyclic for any $\sigma$

Algorithm more flexible; allows certain cycles along guaranteed acyclic relations
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

R.B ∈ descriptive-attributes(R)
Phased Structure Search

Phase 1: consider dependencies from “neighboring classes, via schema relations

Potential-Parents(R.A) = \bigcup_{S.C \in \text{descriptive-attributes}(R \bowtie S)} S.C
Phased Structure Search

Phase 2: consider dependencies from “further” classes, via relation chains

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

\( T.D \subseteq \text{descriptive-attributes}(R \rightarrow S \rightarrow T) \)
Road Map

Factor Graphs
  
Bayesian Nets
  
Markov Nets

Par-factor Graphs

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

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

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
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

\[
\begin{align*}
A & \quad \text{Function operating on the RVs} \\
C & \quad \text{Constraints on instantiations}
\end{align*}
\]

```sql
select doc1.Category, doc2.Category
from Doc doc1, Doc doc2, Link link
where link.from = doc1.Key and link.To = doc2.Key
```
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
- Function operating on the RVs
- Constraints on instantiations

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.

\[ A = \{ \text{category}(D_1, C), \text{category}(D_2, C) \} \]
Markov Logic Networks

- Par-factors are defined using first-order logic statements

\[ \phi = \exp(w \cdot \text{TruthValue}(f(A))) \]
For each formula in the MLN

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

- Next we consider an application of Markov logic to web query disambiguation

- Based on [Mihalkova & Mooney, ECML09]
Web Query Disambiguation

- 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:

- huntsvillehospital
- huntsvillehospital.org
- ebay
- ebay.com
- scrubs
- ???

Historical Sessions:

- huntsville school
- ... scrubs
- scrubs.com
- ... hospitallink.com
- ... scrubs
- scrubs-tv.com
- ebay.com

© Getoor & Mihalkova 2010-2011
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
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`.

```
clickOn(R_1, Q)
clickOn(R_2, Q)
clickOn(R_3, Q)
```
Instantiated Factor Graph

- 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
Instantiated Factor Graph

- 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

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

Total number equals number of sessions that share a click with current one.
Instantiated Factor Graph

- Let's see how these rules define a factor graph.
- Will do this for a single query Q and a set of possible results R1, R2, and R3 for it.

Note: So far, what we have is a relational classifier: 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 \( Q \) and
- a set of possible results \( R \), \( R_1 \), \( R_2 \), and \( R_3 \) for it.

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

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

\[
\text{ClickOn}(R_1, Q) \land R_1 \neq R_2 \Rightarrow \neg \text{ClickOn}(R_2, Q)
\]
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

<table>
<thead>
<tr>
<th>Reference</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>

- Blank means that algorithm can be adapted either way
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>• Capture causal relationships</td>
<td>• Capture symmetric relationships</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>
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

© Getoor & Mihalkova 2010-2011
Summary So Far

- Started with relational classifiers
  - Focus: relational feature construction
- Moved to collective classification models
  - Focus: propagating label assignments
- Considered advanced SRL languages
  - Focus: representing shared structure while allowing for principled learning and inference
LOOKING AHEAD
Statistical Relational Learning and the Web

**Challenges Addressed by SR Learning and Inference**

- Multi-relational data
  - Entities can be of different types
  - Entities can participate in a variety of relationships
- Probabilistic reasoning under noise and/or uncertainty

**Challenges Arising in Web Applications**

- Entities of different types
  - E.g., users, URLs, queries
- Entities participate in a variety of relations
  - E.g., click-on, search-for, link-to, is-refinement-of
- Noisy, sparse observations

What are some challenges that we swept under the rug?
Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted Inference
- Learning from data streams
Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
  - Probabilistic Soft Logic (PSL)
    - [Bröcheler et al., UAI10]
- Lifted Inference
- Learning from data streams
Probabilistic Soft Logic

- First-order-logic-like language for expressing relational dependencies
  \[ \text{Category}(A, C) \iff \text{Category}(B, C) \land \text{Unknown}(A) \land \text{link}(A, B) \land A \neq B \]

- Arbitrary similarity functions on entity attributes:
  \[ A \approx B \iff A.name \approx B.name \]

- Relation-defined sets:
  \[ A \approx B \iff \{A.friends\} \approx \{B.friends\} \]
Combining Soft Values in PSL

\[ H_1 \oplus H_2 \oplus \cdots \oplus H_m \leftarrow B_1 \otimes B_2 \otimes \cdots \otimes B_n \]

- Soft values in a rule are combined using T-norms:
  - Lukasiewicz T-norm (can be customized)
    - \( \oplus(h_1, h_2) = \min(1, h_1+h_2) \)
    - \( \otimes(h_1, h_2) = \max(0, 1- h_1+h_2) \)

Slide credit: Adapted from slides by Matthias Bröcheler
Efficient Inference in PSL

- Attribute and set similarity functions computed externally as “black boxes”
- PSL rules are instantiated “lazily,” on an as-needed basis
- Inference is cast as a constrained continuous numerical optimization problem, solved in polynomial time
PSL in Wikipedia

Graphic credit: Matthias Bröcheler
Wikipedia Rules

\[
\text{hasCat}(A,C) \iff \text{hasCat}(B,C) \land A! = B \land \\
\text{unknown}(A) \land \text{document}(A,T) \land \\
\text{document}(B,U) \land \text{similarText}(T,U)
\]

\[
\text{hasCat}(A,C) \iff \text{hasCat}(B,C) \land \text{unknown}(A) \\
\land \text{link}(A,B) \land A! = B
\]

\[
\text{hasCat}(D,C) \iff \text{talk}(D,A) \land \text{talk}(E,A) \land \\
\text{hasCat}(E,C) \land \text{unknown}(D) \land A! = B
\]

Slide credit: Matthias Bröcheler
Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted inference
  - What is lifted inference?
- Learning from data streams
Lifted Inference Intuitions

- Instantiating an SRL model fully can:
  - Result in intractably large inference problem
  - Be wasteful because computations are repeated due to tying of factors

- Lifted inference approaches recognize redundancies due to symmetries and organize computations to avoid them
  - e.g., summing over entire sets of variables, recognizing identical messages being sent and consolidating them

- Active area of research and a promising direction for successfully scaling to large domains
Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted Inference
- Learning from data streams
  - Work on accurate parameter learning from data streams, e.g. [Huynh & Mooney; SDM11]
Some Other Things We Skipped

- Probabilistic databases
  - [Dalvi & Suciu, VLDB04; Das Sarma et al., ICDE06; Antova et al., VLDBJ09; Sen et al. VLDBJ09]

- Other lifted inference techniques, e.g.,
  - Lifted variable elimination: [Poole, IJCAI03; de Salvo Braz et al IJCAI05, AAAI06; Milch et al., AAAI08]
  - Lifted belief propagation [Jaimovich et al., UAI07; Singla & Domingos, AAAI08; Kersting et al., UAI09; de Salvo Braz et al, SRL-09]

- SRL Models based on probabilistic programming languages
  - E.g., IBAL [Pfeffer, IJCAI01], BLOG [Milch et al., IJCAI05], Church [Goodman et al., UAI08], Factorie [McCallum et al., NIPS09]
Conclusion

- Web & Social Media inherently noisy and relational
- Described a set of well-suited tools for dealing with noisy, relational data
- However, as of yet, not many success stories
- Enablers:
  - Scaling
  - Online Feature construction
  - Dealing with dynamic data
- Time is right: technology & data
  - New platforms, parallel processing
  - More data
  - Growing need for both personalization and privacy
Acknowledgements

- Thank you to the Search Group at Microsoft, Misha Bilenko, Matthias Bröcheler, Amol Deshpande, Nir Friedman, Ariel Fuxman, Aris Gionis, Anitha Kannan, Alex Ntoulas, Hossam Sharara, Marc Smith, Elena Zheleva and others for their slides, comments, and/or helpful discussions.

- Lily is supported by a CI Fellowship under an NSF CRA grant.

- The linqs group at UMD [http://www.cs.umd.edu/linqs](http://www.cs.umd.edu/linqs) is supported by ARO, KDD, NSF, MIPS, Microsoft Research, Google, Yahoo!

© Getoor & Mihalkova 2010-2011