CMPS 290C: Advanced Analytics for Heterogeneous Information Networks
Introduction

Lise Getoor
University of California, Santa Cruz
April 1, 2014

https://courses.soe.ucsc.edu/courses/cmps290c/Spring14/02/

Outline

• Motivation

• Relational Classifiers

• Collective Classification

• Lifted Graphical Models

• Course Logistics
Motivation

Most of the data that is available in the newly emerging era of big data does not look like this.

Or even like this.

It looks more like this.

Web & Query Logs

The Internet
Social Media

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
Social Media Relationships

Predict:
- Sentiment
- Friendship/Fan Affiliations

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

Biological & Biomedical Domains
Predict:
Protein-Protein Relations
Drug-Disease Relations
Protein-Disease Relation

Biological & Biomedical Relationships

Protein-Protein
- Carry
- Inactivate/destroy
- Combine
- Co-express in a condition

Protein-Gene
- Up/Down-regulate

Protein-Disease
- Up/Down-regulated in

Drug-Drug
- Chemical Structure Similarity
- Interaction/Conflict

Drug-Disease
- Treat
- Aggravate/Cause

Disease-Disease
- Family
- Co-Morbidity

Disease-Symptom
- Cause
Massively Open Online Courses (MOOCs)

MOOC Relationships

- **Student-Class**
  - Takes
- **Instructor-Class**
  - Teaches
- **Student-Student**
  - Friends
  - Collaborators
  - Mentors
- **Student-Quiz**
  - Takes
- **Student-Meetup**
  - Joins
- **Student-Question**
  - Posts
  - Answers
Course Overview

- Survey key ideas in representation, inference and learning in structured statistical models
- Study a few ‘canonical’ problems in HINS/MMND/Graphs:
  - Collective classification
  - Link Prediction
  - Entity Resolution
- Learn a few of the languages available
- Highlight opportunities for applying structured statistical models to important big data and data science challenges
Statistical Relational Learning

• Challenges Addressed
  – Multi-relational data
    • Entities can be of different types
    • Entities can participate in a variety of relationships
  – Probabilistic reasoning under noise and/or uncertainty
    • Structured probabilistic models
    • Methods for scaling large structured models

• Closely related areas
  – Structured prediction, hierarchical models, latent-variable relational models, matrix factorization and multi-relational tensors, representation learning, …

Why Statistical Relational Learning (SRL)?

• Traditional **statistical** machine learning approaches assume:
  – A random sample of homogeneous objects from single relation
  – Independent, identically distributed (IID)

• Traditional **relational** machine learning approaches assume:
  – Logical language for describing structure in sample
  – No noise and no uncertainty

• Real world data sets:
  – Multi-relational and heterogeneous
  – Noisy and uncertain
SRL in One Slide

- Collection of techniques which combine rich relational knowledge AI/DB representations with statistical models
  - First-order logic, SQL, graphs
  - Graphical models, directed, undirected, mixed; relational decision trees, etc.
- Examples:
  - Markov Logic Networks (MLNs)
  - Relational Dependency Networks
  - Bayesian Logic Programs
  - Probabilistic Relational Models
  - Probabilistic Soft Logic (PSL)
  - Many others.....
- Key ideas
  - Relational Feature Construction
  - ‘Lifted’ Graphical Models
  - Efficient lifted inference & learning

Road Map

- Motivation
- Relational Classifiers
  - Definition
  - Case Studies
  - Key Idea: Relational Feature Construction
- Collective Classification
- Lifted Graphical Models
- Course Logistics
**IID Classification**

Given:

Task: Predict $Y$ given $X_1$, $X_2$, $X_3$

Classifiers: Use your favorite, logistic regression/SVM, neural net, naïve Bayes, decision trees, etc.

---

**Relational Classification**

Given:
Relational Classification I: Attribute Prediction

Given:

Task: Predict $Y$ given

Relational Classification II: Link Prediction

Given:

Alternate task: Predict existence of relationship between entities

Task: Predict $Y_i - Y_j$ given
Relational Classifiers

- Relational features are pre-computed by aggregating over observed links and attributes of related entities
- 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

Sample Applications

- Case Study 1 (attribute prediction) : Predicting click-through rate of search result ads
- Case Study 2 (link prediction): 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

- Based on approach by [Richardson et al., WWW07]

Relational Features Used
Case Study 2: Predicting Friendships

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

• Description is based on approach by [Zheleva et al., SNAKDD08]
Key Idea: Feature Construction

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

- Relational feature construction
  - Node-specific measures
    - Aggregates: summarize attributes of relational neighbors
    - Structural properties: capture characteristics of the relational structure
  - Node-pair 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 user based on demographic information

**Attribute Aggregates: Level 2**

Based on [Perlich & Provost, KDD03]

- Aggregation over attributes of related entities
  - Values at related entities considered
- Example:

  ![Diagram](image)

  What is this user's political affiliation?

  Number of friends who oppose a tax raise

  Number of friends with observed political affiliation
Attribute Aggregates: Level 3
Based on [Perlich & Provost, KDD03]

- Joint aggregation over attributes of related entities
  - values at related entities considered together as function
- Example:

![Diagram]

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

Attribute Aggregates: Level 4
Based on [Perlich & Provost, KDD03]

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

![Diagram]

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
Based on [Perlich & Provost, KDD03, Popescul & Ungar, MRDM03]

• 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
Based on [Perlich & Provost, KDD03]

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

• Features based on vector distances
  – Difference in distribution over values
Key Idea: Feature Construction

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

• Relational feature construction
  – Node-specific measures
    • Aggregates: summarize attributes of relational neighbors
    • Structural properties: capture characteristics of the relational structure
      – Centrality, Cohesion
      – For more, see [Wasserman & Faust, 94]
  – Node-pair measures

Centrality

• **Degree centrality**: # of neighbors
  – Sometimes normalized by total number of nodes in graph

• **Betweenness centrality**: number of shortest paths from all vertices to all others that pass through that node

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

• **Closeness, eigenvalue centrality**, others
Cohesion

- **Clustering Coefficient**: measures cliquishness; computed as the proportion of all incident edge pairs that are completed by a third one to form a triangle

\[
CC(n) = \frac{\text{# of neighbor links}}{\text{# of possible neighbor links}}
\]

- **Stability**: valence of triads, e.g.,
  - +++, --- are stable
  - ++- instable

Key Idea: Feature Construction

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

- Relational feature construction
  - Node-specific measures
    - Aggregates: summarize attributes of relational neighbors
    - Structural properties: capture characteristics of the relational structure
  - Node-pair measures: summarize properties of (potential) edges
    - Attribute-based measures
    - Edge-based measures
    - Neighborhood similarity measures
Attribute-based Similarity Measures

- Measures defined on pairs of nodes

- Compare nodes based on their attributes’
  - Boolean match/not match
  - String similarity
  - Cosine
  - etc.

- Component similarities are features for relational classifier

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., emailing 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., frequency of exchanged emails
Structural Similarity Measures

• Set similarity measures to compare nodes based on 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

\[
Jaccard(P_1.Friends, P_2.Friends) = \frac{|P_1.Friends \cap P_2.Friends|}{|P_1.Friends \cup P_2.Friends|}
\]
Preferential Attachment Score
[Newman, PRL01, Liben-Nowell & Kleinberg, JASIST07]

- 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
[Adamic & Adar, SN03]

- 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 \[\text{[Jeh & Widom, KDD02]}\]

- Two objects similar if related to similar objects
- Defined as the unique solution to:

\[
s(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} s(I_i(a), I_j(b))
\]

- 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 short paths

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

- Since expensive to compute, often use approximate Katz, assuming some max path length of \(k\)
Key Idea: Feature Construction

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

• Relational feature construction
  – Node-specific measures
    • Aggregates: summarize attributes of relational neighbors
    • Structural properties: capture characteristics of the relational structure
  – Node-pair measures: summarize properties of (potential) edges
    • Attribute-based measures
    • Edge-based measures
    • Neighborhood similarity measures

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 are based on observed values/evidence, cannot be based on attributes or relations that are being predicted

• Makes incorrect independence assumptions
  – Need for better theoretical characterization of how aggregates and structural features affect learnability

• 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

• Motivation

• Relational Classifiers

• Collective Classification

• Lifted Graphical Models

• Course Logistics
Motivation: Joint Inference Problems

• Collective Classification: labeling nodes in a multi-relational graph
  – One large graph – transductive setting
  – What’s different? Not a chain, not a grid, irregular structure
  – Many approaches have been proposed – we will be studying them in the next few weeks!
• Link prediction: predicting edges in multi-relational graph
  – Dependencies among edges – we will be studying them in the following few weeks!
• Group detection: identifying latent structure in multi-relational graph

Road Map

• Motivation
• Relational Classifiers
• Collective Classification
• Lifted Graphical Models
• Course Logistics
Introductions

• Name/nickname
• Where you’re from (interpret as you like, 😊)
• Department, Research interest, advisor, etc.
• Background: newbie, graphical models/ML expert, data guru, domain guru
• Ice breaker question: favorite thing about UCSC?

• Me: Lisa, {San Diego, UCSB, UCB, Stanford, UMD}, CS, ML, Daphne Koller, new @ UCSC

HOMEWORK!!! Post these on the Forum, include an identifiable nickname that we will use in forums and collaborative endeavors.
Introductions

• Name/nickname
• Where you’re from (interpret as you like, 😊)
• Department, Research interest, advisor, etc.
• Background: newbie, graphical models/ML expert, data guru, domain guru
• Ice breaker question: favorite thing about UCSC?

• Me: Lisa, {San Diego, UCSB, UCB, Stanford, UMD}, CS, ML, Daphne Koller, new @ UCSC

• HOMEWORK!!! Post these on the Forum, include an identifiable nickname that we will use in forums and collaborative endeavors. Please do this by next class!

Road Map

• Motivation

• Relational Classifiers

• Collective Classification

• Lifted Graphical Models
  – Representation
    • Background: Graphical Models
    • Key Idea: Par-factor graphs
    • Languages

• Course Logistics
Factor Graphs

• Bipartite graph containing two kinds of nodes
  – variables \( y_1 \)
  – factors \( f \): strictly positive functions of the variables they connect in the graph

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

- Normalizing constant
- Subset of variables that participate in the computation of \( f \)
Factor Graphs: Log-Linear Rep

- Each □ represents \( \exp(\theta_L \cdot f_i(y_i)) \)
- Each ■ represents \( \exp(\theta_G \cdot f_{i,j}(y_i, y_j)) \)

Markov Nets

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

Equivalent expressivity. However, factor graphs are more explicit.
Markov Nets Continued

- Factors are potential functions
- 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)
Important Distinction: Two Kinds of Graphs

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

Relational Graph:
- Nodes represent entities
- Edges represent relationships

While we often draw social networks like this:

**Markov Net:**
- $y_1 = (A \leftrightarrow B)$
- $y_2 = (B \leftrightarrow C)$
- $y_3 = (A \leftrightarrow C)$

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

While we often draw:

**Markov Net:**
- 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)

\[
\begin{align*}
y_1 &= (A \leftrightarrow B) \\
y_2 &= (B \leftrightarrow C) \\
y_3 &= (A \leftrightarrow C)
\end{align*}
\]

\[
\phi_{1,2,3}
\]

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

... one possibility
Variants

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

Bayesian Nets

To cast a Bayesian net as a factor graph, include a factor for each conditional probability distribution (CPD) as a function of each node and its parents.

Going the other way requires ensuring acyclicity.

no normalization required!
Road Map

• Motivation
• Relational Classifiers
• Collective Classification
• Lifted Graphical Models
  – Representation
    • Background: Graphical Models
    • Key Idea: Par-factor graphs
    • Languages
• Course Logistics

Par-factor Graphs

• Factor graphs with parameterized factors
  – Terminology introduced by [Poole, IJCAI03]
• Allow random variables to be parameterized
  – height(X)
  – each parameter is typed with a population, X : person
  – Each population has a size, |person| = 100K
    • height(X_1), ..., height(X_{100000})

• 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
Instantiating Parameterized Random Vars

• Method for manufacturing random variables from description of population (DB, KB, etc.)
• Let A and B be variables, and $A\rightarrow B$ is parameterized random variable
  – Given person = (Ann, Bob, Don, ..., Xin, Yan), we can manufacture random variables from it
  – by replacing A and B with individuals in all possible ways
• Can also be represented by plate model

Constraints

• The constraints in set $C$ govern how par-RVs can be instantiated
• For example, one constraint for our par-RV could be that $B \neq Don$
• With this constraint, the possible instantiations are
Transitivity Par-factor

- $A = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

- $f$ 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 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:

... etc.
Transitivity Par-Factor Instantiated

- To instantiate a par-factor, we need a set of individuals:
  - So much power can be dangerous!
  - Starting with just 3 individuals, end up with a large, densely connected graph
  - Inference becomes very problematic

Another Example

\[ A = \{ A \leftrightarrow F_1, B \leftrightarrow F_2 \} \]
\[ C = \{ \text{Friends}(A, B), A \text{ and } B \text{ are people, } F_1 \text{ and } F_2 \text{ are flavors} \} \]
\[ f = \begin{cases} \exp(\theta) & \text{if } F_1 = F_2 \\ 1 & \text{otherwise} \end{cases} \]
Key Idea: 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

Road Map

- Motivation
- Relational Classifiers
- Collective Classification
- Lifted Graphical Models
  - Representation
    - Background: Graphical Models
    - Key Idea: Par-factor graphs
    - Languages
- Course Logistics
Recap So Far

- Extended factor graphs to allow for convenient parameter tying
- 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 Alphabet Soup

- LBN
- PRISM
- pRN
- RMN
- BLOG
- DAPER
- SRM
- SLP
- RDN
- RDBN
- RBN
- RPM
- PSL
- CLP(BN)
- BLOG
- Factorie
- HMRF
- PRM
- IBAL
- RNM
- PROBLOG
Course Logistics

• Mix of:
  – Presentations, Reading papers, Free for Alls

• Papers:
  – Each person will lead the discussion of at least one paper
  – Everyone will submit QCRs: question, comment, research idea
  – First QCRs will be next Tuesday (3/8), on the knowledge graph identification paper

• Class Project:
  – Project Brainstorming day next Thursday 4/10
  – Can work independently or in groups
  – Project proposal due Thu 4/24
  – Project progress report due Thu 3/15
  – Project presentations Tue 6/3 and Thu 6/5
  – GOAL: Publication!

• Grading: 40% class participation (paper presentation, QCRs, etc.), 60% project

Next classes....

• Thursday 4/3:
  – PSL Tutorial by Steve Bach
  – Please bring your laptop, or share with a friend, with Java 6 and Maven 3 installed
  – Office hours Thu afternoon, Friday afternoon and Monday, in E2 485, for PSL help
  – Skim PSL intro paper

• Tuesday 4/8:
  – Knowledge Graph Identification by Jay Pujara
  – QCRs on the knowledge graph identification paper

• Thursday 4/10:
  – Project Brainstorming day
  – Think about datasets, problems, ideas!!