Learning from single networks:
The impact of network structure on relational learning and collective inference

Jennifer Neville
Departments of Computer Science and Statistics
Purdue University

(joint work with Paul Bennett, John Moore, Joel Pfeiffer, Rongjing Xiang, and Giselle Zeno)
Statistical relational learning

<table>
<thead>
<tr>
<th>Data representation</th>
<th>Knowledge representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic ILP</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td></td>
<td>Weighted FOL</td>
</tr>
</tbody>
</table>
## Statistical relational learning

<table>
<thead>
<tr>
<th></th>
<th>Data representation</th>
<th>Knowledge representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic ILP</td>
<td>Weighted FOL</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td>Probabilistic entity-relation models</td>
<td>Attributed networks</td>
<td>Templated graphical models</td>
</tr>
</tbody>
</table>
### Statistical relational learning

<table>
<thead>
<tr>
<th></th>
<th>Data representation</th>
<th>Knowledge representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Probabilistic ILP</strong></td>
<td>Weighted FOL</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td><strong>Probabilistic entity-relation models</strong></td>
<td>Attributed networks</td>
<td>Templated graphical models</td>
</tr>
<tr>
<td><strong>Relational factorization methods</strong></td>
<td>Matrices/tensors</td>
<td>Latent factors/entity embedding</td>
</tr>
</tbody>
</table>
Statistical relational learning

<table>
<thead>
<tr>
<th>Probabilistic ILP</th>
<th>Data representation</th>
<th>Knowledge representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weighted FOL</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td>Probabilistic entity-relation models</td>
<td>Attributed networks</td>
<td>Templated graphical models</td>
</tr>
<tr>
<td>Relational factorization methods</td>
<td>Matrices/tensors</td>
<td>Latent factors/entity embedding</td>
</tr>
</tbody>
</table>
\[ G = (V, E) \]
\[ V := \text{users} \]
\[ E := \text{friendships} \]
Data network

\[ G = (V, E) \]
\[ V := \text{users} \]
\[ E := \text{friendships} \]
Data network

\[ G = (V, E) \]
\[ V := users \]
\[ E := friendships \]
Attributed network

$G = (V, E)$

$V := users$

$E := friendships$
\[
G = (V, E)
\]
\[
V := users
\]
\[
E := friendships
\]
## Learning over graphs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Full data</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured ML</strong></td>
<td>Multiple <em>(usually small)</em> networks</td>
<td>Set of fully labeled networks</td>
<td>Set of unlabeled networks</td>
</tr>
<tr>
<td><em>(inductive)</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Learning over graphs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Full data</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured ML</strong></td>
<td>Multiple <em>(usually small)</em> networks</td>
<td>Set of fully labeled networks</td>
<td>Set of unlabeled networks</td>
</tr>
<tr>
<td><strong>(inductive)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transductive learning</strong></td>
<td>One large, partially-labeled network</td>
<td>Induced fully labeled subnetwork(s)</td>
<td>Full network, predict over unlabeled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Learning over graphs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Full data</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured ML</strong></td>
<td>Multiple <em>(usually small)</em> networks</td>
<td>Set of fully labeled networks</td>
<td>Set of unlabeled networks</td>
</tr>
<tr>
<td>(inductive)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transductive</strong></td>
<td>One large, partially-labeled network</td>
<td>Induced fully labeled subnetwork(s)</td>
<td>Full network, predict over unlabeled</td>
</tr>
<tr>
<td>learning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Inductive</strong></td>
<td>One <em>(or few)</em> large, partially-labeled network(s)</td>
<td>Induced fully labeled subnetwork(s)</td>
<td>Within or across networks, predict over unlabeled</td>
</tr>
<tr>
<td>learning</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Learning over graphs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Full data</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured ML</strong></td>
<td>Multiple (usually small) networks</td>
<td>Set of fully labeled networks</td>
<td>Set of unlabeled networks</td>
</tr>
<tr>
<td>(inductive)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transductive learning</strong></td>
<td>One large, partially-labeled network</td>
<td>Induced fully labeled subnetwork(s)</td>
<td>Full network, predict over unlabeled</td>
</tr>
<tr>
<td><strong>Inductive learning</strong></td>
<td>One (or few) large, partially-labeled networks(s)</td>
<td>Induced fully labeled subnetwork(s)</td>
<td>Within or across networks, predict over unlabeled</td>
</tr>
</tbody>
</table>
Graph regularization methods: Assume linked nodes exhibit homophily and make predictions via optimization (i.e., label propagation)

Make predictions for unlabeled nodes using **collective inference** (observed labels seed inference process)

Small world graph
Labeled nodes: 30%
Autocorrelation: 0.5
Make predictions for unlabeled nodes using **collective inference** (observed labels seed inference process)

Small world graph
Labeled nodes: 30%
Autocorrelation: 0.5
Probabilistic modeling: Learn templated graphical model and use joint inference to make predictions

Model template

Data network
Probabilistic modeling: Learn templated graphical model and use joint inference to make predictions.

Model graph
(rolled-out graphical model)
Probabilistic modeling: Learn templated graphical model and use joint inference to make predictions

Model graph
(rolled-out graphical model)
**Probabilistic modeling:** Learn templated graphical model and use joint inference to make predictions.

Model graph

(rolled-out graphical model)
**Probabilistic modeling**: Learn templated graphical model and use joint inference to make predictions

Learn model parameters from fully labeled network

\[
P(y_G|x_G) = \frac{1}{Z(\theta, x_G)} \prod_{T \in T} \prod_{C \in C(T(G))} \Phi_T(x_C, y_C; \theta_T)
\]
Probabilistic modeling: *Learn* templated graphical model and use joint inference to make predictions.

Learn model parameters from fully labeled network:

\[ P(y_G | x_G) = \frac{1}{Z(\theta, x_G)} \prod_{T \in T} \prod_{C \in C(T(G))} \Phi_{\text{NN}}(y_G, \theta) \]
How does network structure impact model performance?
Collective classification uses full joint, rolled out model for inference... on partially labeled network
Collective classification uses full joint, rolled out model for inference... on partially labeled network.
Collective classification uses full joint, rolled out model for inference... on partially labeled network.
Collective classification uses full joint, rolled out model for inference... on partially labeled network
Collective classification uses full joint, rolled out model for inference... on partially labeled network
Collective classification uses full joint, rolled out model for inference... on partially labeled network
Collective classification uses full joint, rolled out model for inference... on partially labeled network

The structure of collective classification models is determined by the structure of the underlying data network, including location + availability of labels...

...this can impact performance of learning and inference methods
Collective classification uses full joint, rolled out model for inference... on partially labeled network

The structure of collective classification models is determined by the structure of the underlying data network, including location + availability of labels

...this can impact performance of learning and inference methods

due to differences between learning and inference networks
Methods for collective classification

<table>
<thead>
<tr>
<th>Probabilistic modeling (inductive)</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-supervised learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMN (Taskar et al.'01); MLN (Richardson et al.'06); RDN (Neville et al.'06)</td>
<td>SocialDim (Tang et al.'09); RelationStr (Xiang et al.'10);</td>
<td></td>
<td>LBC (Lu et al.'03); PL-EM (Xiang et al.'08); CC-HLR (McDowell et al.'12); RDA (Pfeiffer et al.'14);</td>
</tr>
</tbody>
</table>
Methods for collective classification

<table>
<thead>
<tr>
<th>Probabilistic modeling (inductive)</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-supervised learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMN (Taskar et al.’01); MLN (Richardson et al.’06); RDN (Neville et al.’06)</td>
<td>SocialDim (Tang et al.’09); RelationStr (Xiang et al.’10)</td>
<td></td>
<td>LBC (Lu et al.’03); PL-EM (Xiang et al.’08); CC-HLR (McDowell et al.’12); RDA (Pfeiffer et al.’14)</td>
</tr>
</tbody>
</table>

Change objective/model to match data
### Methods for collective classification

<table>
<thead>
<tr>
<th>Graph regularization (transductive)</th>
<th>No learning (optimization only)</th>
<th>Add links</th>
<th>Learn link weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRF (Zhu et al.'03); wvRN (Macskaey et al.'07)</td>
<td>wvRN+ (Macskaey '07); GhostEdges (Gallagher et al.'08); SCRN (Wang et al.'13)</td>
<td>GNetMine (Ji et al.'10); LNP (Shi et al.'11); LGCW (Dhurandhar et al.'13)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probabilistic modeling (inductive)</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-super. learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMN (Taskar et al.'01); MLN (Richardson et al.'06); RDN (Neville et al.'06)</td>
<td>SocialDim (Tang et al.'09); RelationStr (Xiang et al.'10);</td>
<td>LBC (Lu et al.'03); PL-EM (Xiang et al.'08); CC-HLR (McDowell et al.'12); RDA (Pfeiffer et al.'14)</td>
<td></td>
</tr>
</tbody>
</table>

---

Change objective/model to match data
## Methods for collective classification

<table>
<thead>
<tr>
<th>Graph regularization (transductive)</th>
<th>No learning (optimization only)</th>
<th>Add links</th>
<th>Learn link weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRF (Zhu et al.'03); wvRN (Macsassy et al.'07)</td>
<td>wvRN+ (Macsassy '07); GhostEdges (Gallagher et al.'08); SCRN (Wang et al.'13)</td>
<td></td>
<td>GNetMine (Ji et al.'10); LNP (Shi et al.'11); LGCW (Dhurandhar et al.'13)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probabilistic modeling (inductive)</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-supervised learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMN (Taskar et al.'01); MLN (Richardson et al.'06); RDN (Neville et al.'06)</td>
<td>SocialDim (Tang et al.'09); RelationStr (Xiang et al.'10);</td>
<td></td>
<td>LBC (Lu et al.'03); PL-EM (Xiang et al.'08); CC-HLR (McDowell et al.'12); RDA (Pfeiffer et al.'14)</td>
</tr>
</tbody>
</table>

*Change data to match objective*

*Change objective/model to match data*
How do regularization methods compare to probabilistic modeling?
Regularization vs. probabilistic modeling

(Zeno and N., MLG’16)
Regularization vs. probabilistic modeling
(Zeno and N., MLG’16)

Probabilistic modeling can exploit dependencies across wider range of scenarios, link density impacts performance
### Methods for collective classification

<table>
<thead>
<tr>
<th>Graph regularization</th>
<th>No learning (optimization only)</th>
<th>Add links</th>
<th>Learn link weights</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRF</strong> (Zhu et al.'03); <strong>wvRN</strong> (Macskassy et al.'07)</td>
<td><strong>wvRN+</strong> (Macskassy '07); <strong>GhostEdges</strong> (Gallagher et al.'08); <strong>SCRN</strong> (Wang et al.'13)</td>
<td><strong>GNetMine</strong> (Ji et al.'10); <strong>LNP</strong> (Shi et al.'11); <strong>LGCW</strong> (Dhurandhar et al.'13)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probabilistic modeling</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-supervised learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RMN</strong> (Taskar et al.'01); <strong>MLN</strong> (Richardson et al.'06); <strong>RDN</strong> (Neville et al.'06)</td>
<td><strong>SocialDim</strong> (Tang et al.'09); <strong>RelationStr</strong> (Xiang et al.'10);</td>
<td><strong>LBC</strong> (Lu et al.'03); <strong>PL-EM</strong> (Xiang et al.'08); <strong>CC-HLR</strong> (McDowell et al.'12); <strong>RDA</strong> (Pfeiffer et al.'14)</td>
<td></td>
</tr>
</tbody>
</table>
# Methods for collective classification

<table>
<thead>
<tr>
<th>Graph regularization</th>
<th>No learning (optimization only)</th>
<th>Add links</th>
<th>Learn link weights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>GRF</strong> (Zhu et al.’03); <strong>wvRN</strong> (Macskassy et al.’07)</td>
<td><strong>wvRN+</strong> (Macskassy ’07); <strong>GhostEdges</strong> (Gallagher et al.’08); <strong>SCRN</strong> (Wang et al.’13)</td>
<td><strong>GNetMine</strong> (Ji et al.’10); <strong>LNP</strong> (Shi et al.’11); <strong>LGCW</strong> (Dhurandhar et al.’13)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probabilistic modeling</th>
<th>Learn from labeled only</th>
<th>Add graph features</th>
<th>Semi-supervised learning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>RMN</strong> (Taskar et al.’01); <strong>MLN</strong> (Richardson et al.’06); <strong>RDN</strong> (Neville et al.’06)</td>
<td><strong>SocialDim</strong> (Tang et al.’09); <strong>RelationStr</strong> (Xiang et al.’10);</td>
<td><strong>LBC</strong> (Lu et al.’03); <strong>PL-EM</strong> (Xiang et al.’08); <strong>CC-HLR</strong> (McDowell et al.’12); <strong>RDA</strong> (Pfeiffer et al.’14)</td>
</tr>
</tbody>
</table>

Graph structure is more consistent across training and test scenarios
How to learn from one partially labeled network?
Semi-supervised learning in graphs

- Use entire network to jointly learn parameters and make inferences about class labels of unlabeled nodes

\[
P(Y|X, E, \Theta_Y)
\]
Semi-supervised learning in graphs

- Use entire network to jointly learn parameters and make inferences about class labels of unlabeled nodes
- Lu and Getoor (ICML’03) use relational features and ICA
- McDowell and Aha (ICML’12) combine two classifiers with label regularization

\[ P(Y|X, E, \Theta_Y) \]
Relational EM (Xiang and N. ’08)
Relational EM *(Xiang and N. ’08)*

**Expectation (E) Step**

\[
P_{\tilde{\mathbf{y}}}(\tilde{y}_i | \mathbf{Y}_{MB}(v_i), \mathbf{x}_i, \Theta_{\tilde{\mathbf{y}}})
\]

Predict unknown labels with collective classification
Relational EM (Xiang and N. ’08)

Predict unknown labels with collective classification

\[ P_{\mathcal{Y}}(y_i | \mathbf{Y}_{MB}(v_i), \mathbf{x}_i, \Theta_{\mathcal{Y}}) \]

Expectation (E) Step

Maximization (M) Step

\[ \hat{\Theta}_{\mathcal{Y}} = \arg \max_{\Theta_{\mathcal{Y}}} \sum_{\mathbf{Y}_U \in \mathcal{Y}_U} P_{\mathcal{Y}}(\mathbf{Y}_U) \{ \text{summation over local log conditionals} \} \]

Use predicted probabilities during optimization (in local conditionals)
Relational EM \textit{(Xiang and N. ’08)}

**Expectation (E) Step**

\[ P_\mathcal{Y}(y_i | \mathcal{Y}_{MB}(v_i), x_i, \Theta_\mathcal{Y}) \]

**Maximization (M) Step**

\[ \hat{\Theta}_\mathcal{Y} = \arg \max_{\Theta_\mathcal{Y}} \sum_{Y_U \in Y_U} P_\mathcal{Y}(Y_U) \text{ [summation over local log conditionals]} \]

Predict unknown labels with collective classification

Use predicted probabilities during optimization (in local conditionals)
How does relational EM perform?
How does relational EM perform?

- Works well when network has a moderate amount of labels
How does relational EM perform?

• Works well when network has a moderate amount of labels
• If network is sparsely labeled, it is often better to use a model that is not learned
How does relational EM perform?

- Works well when network has a moderate amount of labels
- If network is sparsely labeled, it is often better to use a model that is **not learned**
  
- **Why?** In sparsely labeled networks, errors from the collective classification compound during propagation
Utilizing network structure during model estimation

Partially labeled network \((G)\)
Utilizing network structure during model estimation

Partially labeled network \((G)\)

Pseudolikelihood \((G_L)\)
Utilizing network structure during model estimation

Partially labeled network \((G)\)

Composite Likelihood \((G)\)

Pseudolikelihood \((G_L)\)
Utilizing network structure during model estimation

Partially labeled network ($G$)

Composite Likelihood ($G$)

Pseudolikelihood ($G_L$)

Pseudolikelihood ($G$)
Utilizing network structure during model estimation

Partially labeled network \((G)\)

Composite Likelihood \((G)\)

Fully incorporating unlabeled structure in EM exacerbates propagation error, more so when labels are sparse
Finding:
Network structure can bias inference in partially-labeled networks; *maximum entropy constraints correct for bias*
Effect of relational biases on relational EM
Effect of relational biases on relational EM

- We compared CL-EM and PL-EM and examined the distribution of predicted probabilities on a real world dataset
  - Amazon Co-occurrence (SNAP)
  - Varied class priors, 10% Labeled
Effect of relational biases on relational EM

- We compared CL-EM and PL-EM and examined the distribution of predicted probabilities on a real world dataset
  - Amazon Co-occurrence (SNAP)
  - Varied class priors, 10% Labeled
Effect of relational biases on relational EM

- We compared CL-EM and PL-EM and examined the distribution of predicted probabilities on a real world dataset
  - Amazon Co-occurrence (SNAP)
  - Varied class priors, 10% Labeled
- **Overpropagation error** during inference causes PL-EM to collapse to single prediction
- **Worse** on sparsely labeled datasets
Effect of relational biases on relational EM

- We compared CL-EM and PL-EM and examined the distribution of predicted probabilities on a real world dataset
  - Amazon Co-occurrence (SNAP)
  - Varied class priors, 10% Labeled

- **Overpropagation error** during inference causes PL-EM to collapse to single prediction
- **Worse** on sparsely labeled datasets

![Error Graph]

Need method to correct bias for any method based on local (relational) conditional
Maximum entropy inference for PL-EM (Pfeiffer et al. WWW’15)
Maximum entropy inference for PL-EM *(Pfeiffer et al. WWW’15)*

- Correction to *inference* (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
Maximum entropy inference for PL-EM *(Pfeiffer et al. WWW’15)*

- Correction to *inference* (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: *Shift* the probabilities up/down
Maximum entropy inference for PL-EM (Pfeiffer et al. WWW’15)

- Correction to *inference* (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down
Maximum entropy inference for PL-EM \textit{(Pfeiffer et al. WWW’15)}

- Correction to \textit{inference} (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- \textbf{Idea}: The proportion of negatively \textit{predicted} items should equal the proportion of negatively \textit{labeled} items
  - \textbf{Fix}: Shift the probabilities up/down
Maximum entropy inference for PL-EM (Pfeiffer et al. WWW’15)

- Correction to inference (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- Idea: The proportion of negatively predicted items should equal the proportion of negatively labeled items
  - Fix: Shift the probabilities up/down

Transform probabilities to logit space:
\[ h_i = \sigma^{-1}(P(y_i = 1)) \]

\((P(y) = 0.5)\)
Maximum entropy inference for PL-EM (*Pfeiffer et al. WWW’15*)

- Correction to *inference* (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)

- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down

\[
h_i = \sigma^{-1}(P(y_i = 1))
\]
Maximum entropy inference for PL-EM *(Pfeiffer et al. WWW’15)*

- **Correction to inference** (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down

- **Transform probabilities to logit space**:
  \[ h_i = \sigma^{-1}(P(y_i = 1)) \]
- **Compute offset location**:
  \[ \phi = P(0) \cdot |V_U| \]

(P(y) = 0.5)

Pivot = 5/7
Maximum entropy inference for PL-EM *(Pfeiffer et al. WWW’15)*

- **Correction to inference** (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down

Transform probabilities to logit space:

\[ h_i = \sigma^{-1}(P(y_i = 1)) \]

Compute offset location:

\[ \phi = P(0) \cdot |V_U| \]

Adjust logits:

\[ h_i = h_i - h(\phi) \]

\[ (P(y) = 0.5) \]

Pivot = 5/7
Maximum entropy inference for PL-EM (*Pfeiffer et al. WWW’15*)

- **Correction to inference** (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down

- **Transform probabilities to logit space**:
  \[ h_i = \sigma^{-1}(P(y_i = 1)) \]

- **Compute offset location**:
  \[ \phi = P(0) \cdot |V_U| \]

- **Adjust logits**:
  \[ h_i = h_i - h(\phi) \]

- **Transform back to probabilities**:
  \[ P(y_i) = \sigma(h_i) \]

**Diagram**

- **Pivot = 5/7**
- **(P(y) = 0.5)**
Maximum entropy inference for PL-EM (Pfeiffer et al. WWW’15)

- Correction to *inference* (E-Step)
  - Enables estimation with the pseudolikelihood (M-Step)
- **Idea**: The proportion of negatively *predicted* items should equal the proportion of negatively *labeled* items
  - **Fix**: Shift the probabilities up/down
- Repeat for each *inference itr*

\[
\begin{align*}
\text{transform probabilities to logit space:} & & h_i = \sigma^{-1}(P(y_i = 1)) \\
\text{compute offset location:} & & \phi = P(0) \cdot |V_U| \\
\text{adjust logits:} & & h_i = h_i - h(\phi) \\
\text{transform back to probabilities:} & & P(y_i) = \sigma(h_i)
\end{align*}
\]
Corrected probabilities are used to retrain during PL-EM (M-Step)

\[ P(y) = 0.5 \]

Pivot = 5/7
Experimental results - Correction effects
Experimental results - Correction effects

Amazon (small class prior)

Amazon (large class prior)
Experimental results - Correction effects

Amazon (small class prior)

Amazon (large class prior)
Experimental results - Correction effects

Amazon (small class prior)

Amazon (large class prior)
Max entropy correction removes bias due to over propagation in collective inference.
Experimental results - Large patent dataset
Experimental results - Large patent dataset

Computers

Organic
Experimental results - Large patent dataset

Computers

Organic
Experimental results - Large patent dataset
Experimental results - Large patent dataset
Experimental results - Large patent dataset

Correction allows relational EM to improve over competing methods in sparsely labeled domains

Note: McDowell & Aha (ICML’12) may correct same effect, but during estimation rather than inference
Can neural networks improve collective inference by further reducing bias?
Deep collective inference *(Moore and N. AAAI’17)*

- **Approach**: Use a neural network with neighbors’ attributes and class label predictions as inputs

- **Key ideas**:
  - Represent set of neighbors as a *sequence*, in *random order*
  - To deal with heterogenous inputs (i.e., varying number of neighbors), use a *recurrent* model (LSTM)
  - To learn with partially labeled network, use *semi-supervised* collective classification
Example network

- red = target node
- blue = neighbors
- grey = labeled
- white = unlabeled
Example network

red = target node
blue = neighbors
grey = labeled
white = unlabeled

b e i f a d
Example network

red = target node
blue = neighbors
grey = labeled
white = unlabeled

\[
\begin{align*}
\hat{y}_b^{(t_c-1)} & \quad [f_b, \hat{y}_b^{(t_c-1)}] \\
\hat{y}_d^{(t_c)} & \quad [f_d, \hat{y}_d^{(t_c-1)}]
\end{align*}
\]
Deep collective inference (DCI) model

• For node $v_i$, and current iteration $t_c$, the input is node features concatenated with previous prediction $[f_i, \hat{y}_i^{(t_c-1)}]$ and neighbor features concatenated with predictions/labels $\{[f_j, (y_j \text{ or } \hat{y}_j^{(t_c-1)})] | v_j \in \mathcal{N}_i\}$.

• For node $v_i$, specified input is:

$$x_i = \begin{bmatrix} x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(|\mathcal{N}_i|)} \end{bmatrix}$$

$$= \begin{bmatrix} f_{j_1}, y_{j_1}, f_{j_2}, y_{j_2}, \ldots, f_i, \hat{y}_i^{(t_c-1)} \end{bmatrix}$$

$$x_d = \begin{bmatrix} <f_b, y_b>, <f_e, y_e>, <f_i, y_i>, <f_p, \hat{y}_i^{(t_c-1)}>, <f_a, y_a>, <f_d, \hat{y}_d> \end{bmatrix}$$

$$= \begin{bmatrix} x_d^{(0)}, x_d^{(1)}, x_d^{(2)}, x_d^{(3)}, x_d^{(4)}, x_d^{(5)} \end{bmatrix}$$
LSTM Structure

Structure of LSTM with sequential inputs
LSTM Structure

Structure of LSTM with sequential inputs

LSTM input at end of sequence with \( w \) hidden units and \( p \) features
Learning: key aspects

- **Initialize label predictions** with non-collective version of model—Deep Relational Inference (DRI)

- **Semi-supervised learning**: Estimate model parameters until convergence, then perform collective inference to make predictions for all unlabeled nodes

- **Randomize neighbor order** on every iteration

- **Correct for imbalanced classes**, either by balancing the objective function or by balancing the data with augmentation

- Use backpropagation through time with early stopping and cross-entropy loss
Evaluation on small-medium sized networks

Lower BAE is better
Evaluation on large network (900K nodes)

Patents (17/83)

Lower BAE is better
Evaluation on large network (900K nodes)

**Patents (17/83)**

Overall:
- 12% gain over PLEM+N2V
- 20% gain over RNCC (competing NN)

Lower BAE is better
<table>
<thead>
<tr>
<th>Data representation</th>
<th>Knowledge representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic ILP</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td>Probabilistic entity-relation models</td>
<td>Attributed networks</td>
</tr>
<tr>
<td>Relational factorization methods</td>
<td>Matrices/tensors</td>
</tr>
<tr>
<td></td>
<td>Data representation</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Probabilistic ILP</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td>Probabilistic entity-relation models</td>
<td>Attributed networks</td>
</tr>
<tr>
<td>Relational factorization methods</td>
<td>Matrices/tensors</td>
</tr>
<tr>
<td>Relational deep learning</td>
<td>Propositionalized relations</td>
</tr>
<tr>
<td></td>
<td>Data representation</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Probabilistic ILP</td>
<td>Weighted FOL</td>
</tr>
<tr>
<td>Probabilistic entity-relation models</td>
<td>Attributed networks</td>
</tr>
<tr>
<td>Relational factorization methods</td>
<td>Matrices/tensors</td>
</tr>
<tr>
<td>Relational deep learning</td>
<td>Propositionalized relations</td>
</tr>
</tbody>
</table>

Still much work needed to define data input and NN architectures that will work for relational data.
Other findings where network structure impacts SRL performance
Impact of network on collective classification
Impact of network on collective classification

• **Non-stationarity** in network structure reduces accuracy *(Angin & N ’08)*
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy *(Angin & N ’08)*
- **Propagation error** during inference is mediated by local network structure *(Xiang & N ’11)*
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy *(Angin & N ’08)*

- **Propagation error** during inference is mediated by local network structure *(Xiang & N ’11)*

- Difference in network structure and label availability results in **variance among node marginals**, which impairs performance *(Xiang & N ’13)*
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy (Angin & N ’08)
- **Propagation error** during inference is mediated by local network structure (Xiang & N ’11)
- Difference in network structure and label availability results in variance among node marginals, which impairs performance (Xiang & N ’13)
- Structure of rolled-out model can lead to inference instability in collective classification (Pfeiffer et al. ’14a)
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy (Angin & N ’08)

- **Propagation error** during inference is mediated by local network structure (Xiang & N ’11)

- Difference in network structure and label availability results in **variance among node marginals**, which impairs performance (Xiang & N ’13)

- Structure of rolled-out model can lead to **inference instability** in collective classification (Pfeiffer et al. ’14a)

- Active sampling of networks during learning can lead to **biased parameters estimates** unless unobserved edges are taken into account (Pfeiffer et al. ’14b)
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy *(Angin & N ‘08)*
- **Propagation error** during inference is mediated by local network structure *(Xiang & N ‘11)*
- Difference in network structure and label availability results in variance among node marginals, which impairs performance *(Xiang & N ‘13)*
- Structure of rolled-out model can lead to inference instability in collective classification *(Pfeiffer et al. ‘14a)*
- Active sampling of networks during learning can lead to biased parameter estimates unless unobserved edges are taken into account *(Pfeiffer et al. ‘14b)*

The common thread among these effects is a difference in graph distribution between the “learning” and “inference” settings...

which increases error due to **bias**—in learning and/or inference
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy (Angin & N’08)

- **Propagation error** during inference is mediated by local network structure (Xiang & N’11)

- Difference in network structure and label availability results in variance among node marginals, which impairs performance (Xiang & N’13)

- Structure of rolled-out model can lead to inference instability in collective classification (Pfeiffer et al. ’14a)

- Active sampling of networks during learning can lead to biased parameter estimates unless unobserved edges are taken into account (Pfeiffer et al. ’14b)

The common thread among these effects is a difference in graph distribution between the “learning” and “inference” settings…

which increases error due to bias—in learning and/or inference

(recall) Task is to estimate joint distribution: $P(Y|\{X\}_n, G)$
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy (Angin & N '08)
- **Propagation error** during inference is mediated by local network structure (Xiang & N '11)

The common thread among these effects is a difference in graph distribution between the “learning” and “inference” settings… which increases error due to bias—in learning and/or inference

(recall) Task is to estimate joint distribution: \( P(Y|\{X\}_n, G) \)

Need to accurately model the **target graph distribution** during learning:
Impact of network on collective classification

- **Non-stationarity** in network structure reduces accuracy \((\text{Angin} \& \text{N } '08)\)
- **Propagation error** during inference is mediated by local network structure \((\text{Xiang} \& \text{N } '11)\)

The common thread among these effects is a difference in graph distribution between the “learning” and “inference” settings…

which increases error due to bias—
in learning and/or inference

(recall) Task is to estimate joint distribution: \(P(Y|\{X\}_n, G)\)

Need to accurately model the **target graph distribution** during learning:

Goal: learn a model from \(<G_s, Y_s>\) and apply it to \(<G_t, Y_t>\).
Thanks
neville@cs.purdue.edu
www.cs.purdue.edu/~neville