Message passing and node classification

CS224W: Analysis of Networks
Jure Leskovec with Srijan Kumar, Stanford University
http://cs224w.stanford.edu
Main question today: Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?

Example: In a network, some nodes are fraudsters and some nodes are fully trusted. How do you find the other fraudsters and trustworthy nodes?
Main question today: Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?

Collective classification: Idea of assigning labels to all nodes in a network together

Intuition: Correlations exist in networks. Leverage them!

We will look at three techniques today:

- Relational classification
- Iterative classification
- Belief propagation
Individual behaviors are correlated in a network environment.

Three types of dependencies that lead to correlation:

- **Homophily**
  - Individual Characteristics
  - Social Connections

- **Influence**
  - Social Connections
  - Individual Characteristics

- **Confounding**
  - Environment
  - Individual Characteristics
  - Social Connections
Example:

- Real social network
  - Nodes = people
  - Edges = friendship
  - Node color = race

- People are segregated by race due to homophily

(Easley and Kleinberg, 2010)
Classification with network data

- How to leverage this correlation observed in networks to help predict user attributes or interests?

How to predict the labels for the nodes in yellow?
Similar entities are typically close together or directly connected:

- “Guilt-by-association”: If I am connected to a node with label X, then I am likely to have label X as well.

Example: Malicious/benign web page: Malicious web pages link to one another to increase visibility, look credible, and rank higher in search engines.
Motivation

- **Classification label** of an object $O$ in network may depend on:
  - **Features** of $O$
  - **Labels** of the objects in $O$’s neighborhood
  - **Features** of objects in $O$’s neighborhood
Given:
• graph and
• few labeled nodes

Find: class (red/green) for rest nodes

Assuming: networks have homophily
Let $W$ be a $n \times n$ (weighted) adjacency matrix over $n$ nodes.

Let $Y = \{-1, 0, 1\}^n$ be a vector of labels:

1: **positive** node, known to be involved in a gene function/biological process

-1: **negative** node

0: **unlabeled** node

**Goal:** Predict which **unlabeled** nodes are likely **positive**
Collective classification

- **Intuition**: simultaneous classification of interlinked objects using correlations
- **Several applications**
  - Document classification
  - Part of speech tagging
  - Link prediction
  - Optical character recognition
  - Image/3D data segmentation
  - Entity resolution in sensor networks
  - Spam and fraud detection
Collective classification overview

- **Markov Assumption**: the label $Y_i$ of one node $i$ depends on the label of its neighbors $N_i$

\[
P(Y_i|i) = P(Y_i|N_i)
\]

- Collective classification involves 3 steps:
  - **Local Classifier**: Assign initial label
  - **Relational Classifier**: Capture correlations between nodes
  - **Collective Inference**: Propagate correlations through network
Collective classification overview

Local Classifier: used for initial label assignment
- Predicts label based on node attributes/features
- Classical classification learning
- Does not employ network information

Relational Classifier: capture correlations based on the network
- Learn a classifier from the labels or/and attributes of its neighbors to label one node
- Network information is used

Collective Inference: propagate the correlation
- Apply relational classifier to each node iteratively
- Iterate until the inconsistency between neighboring labels is minimized
- Network structure substantially affects the final prediction
Collective classification models

- Exact inference is practical only when the network satisfies certain conditions
- We will look at techniques for approximate inference:
  - Relational classification
  - Iterative classification
  - Belief propagation
- All are *iterative algorithms*
Problem setting

- How to predict the labels $Y_i$ for the nodes $i$ in yellow?
- Each node $i$ has a feature vector $f_i$
- Labels for some nodes are given (+ for green, - for blue)
- Task: find $P(Y_i)$ given all features and the network
Basic idea: Class probability of $Y_i$ is a weighted average of class probabilities of its neighbors.

For labeled nodes, initialize with ground-truth $Y$ labels.

For unlabeled nodes, Initialize $Y$ uniformly.

Update all nodes in a random order till convergence or till maximum number of iterations is reached.
Probabilistic relational classifier

- **Repeat** for each node $i$ and label $c$

\[ P(Y_i = c) = \frac{1}{|N_i|} \sum_{(i,j) \in E} W(i,j)P(Y_j = c) \]

- $W(i,j)$ is the edge strength from $i$ to $j$
- $|N_i|$ is the number of neighbors of $i$

- **Challenges:**
  - Convergence is not guaranteed
  - Model cannot use node feature information
Initialization: All labeled nodes to their labels and all unlabeled nodes uniformly
Update for the 1\textsuperscript{st} Iteration:

- For node 3, $N_3 = \{1, 2, 4\}$

\[
\begin{align*}
\Pr(Y=1|N_3) &= \frac{1}{3} (0 + 0 + 0.5) = 0.17 \\
\Pr(Y=1) &= 1 \\
\Pr(Y=1) &= 0.5 \\
\Pr(Y=1) &= 0.5 \\
\Pr(Y=1) &= 0.5 \\
\Pr(Y=1) &= 0.5 \\
\Pr(Y=1) &= 0 \\
\Pr(Y=1) &= 0 \\
\Pr(Y=1) &= 1 \\
\Pr(Y=1) &= 0
\end{align*}
\]
Update for the 1\textsuperscript{st} Iteration:

- For node 4, $N_4 = \{1, 3, 5, 6\}$

\[
P(Y = 1) = 1
\]

\[
P(Y = 1) = 0.5
\]

\[
P(Y = 1) = 0.5
\]

\[
P(Y = 1) = 1
\]

\[
P(Y = 1) = 0.17
\]

\[
P(Y = 1) = \frac{1}{4}(0 + 0.17 + 0.5 + 1) = 0.42
\]

\[
P(Y = 1) = 0
\]
Probabilistic relational classifier example

- Update for the 1st Iteration:
  - For node 5, \( N_5 = \{4, 6, 7, 8\} \)

\[
\begin{align*}
P(Y = 1) &= 1 \\
P(Y = 1 | N_5) &= \frac{1}{4} (0.42 + 1 + 1 + 0.5) = 0.73
\end{align*}
\]

\[
\begin{align*}
P(Y = 1) &= 0.5 \\
P(Y = 1 | N_4) &= 0.42
\end{align*}
\]

\[
\begin{align*}
P(Y = 1) &= 0.17 \\
P(Y = 1 | N_5) &= 0.73
\end{align*}
\]

\[
\begin{align*}
P(Y = 1) &= 0 \\
P(Y = 1 | N_4) &= 0.42
\end{align*}
\]
After Iteration 1

P(Y = 1) = 1.00

P(Y = 1) = 0.73

P(Y = 1) = 0.91

P(Y = 1) = 0.42

P(Y = 1) = 0.17

P(Y = 1) = 0

P(Y = 1) = 0
After Iteration 2

P(Y = 1) = 1.00

P(Y = 1) = 0.95

P(Y = 1) = 0.85

P(Y = 1) = 0.47

P(Y = 1) = 0.14

P(Y = 1) = 0
After Iteration 3

P(Y = 1) = 1.00

P(Y = 1) = 0.95

P(Y = 1) = 0.86

P(Y = 1) = 0.50

P(Y = 1) = 0.16

P(Y = 1) = 0.00

P(Y = 1) = 0.00

P(Y = 1) = 0.00

P(Y = 1) = 0.00

P(Y = 1) = 0.00
After Iteration 4

\[ P(Y = 1) = 1.00 \]
\[ P(Y = 1) = 0.95 \]
\[ P(Y = 1) = 0.86 \]
\[ P(Y = 1) = 0.51 \]
\[ P(Y = 1) = 0.16 \]
\[ P(Y = 1) = 0 \]
\[ P(Y = 1) = 0 \]
All scores stabilize after 5 iterations:

- Nodes 5, 8, 9 are + \((P(Y_i = 1) > 0.5)\)
- Node 3 is – \((P(Y_i = 1) < 0.5)\)
- Node 4 is in between \((P(Y_i = 1) =0.5)\)
Collective classification models

- Relational classifiers
- **Iterative classification**
- Loopy belief propagation
Iterative classification

- Relational classifiers **do not use node attributes**. How can one leverage them?

- **Main idea of iterative classification:** classify node $i$ based on its attributes as well as labels of neighbor set $N_i$. 
Iterative classification

- Relational classifiers do not use node attributes. How can one leverage them?

- Main idea of iterative classification: classify node $i$ based on its attributes as well as labels of neighbor set $N_i$.
- Create a flat vector $a_i$ for each node $i$.
- Train a classifier to classify using $a_i$.
- Node may have various number of neighbors, so we can aggregate using: count, mode, proportion, mean, exists, etc.
Basic architecture of iterative classifiers

- **Bootstrap phase**
  - Convert each node $i$ to a flat vector $a_i$
  - Use local classifier $f(a_i)$ (e.g., SVM, kNN, ...) to compute best value for $Y_i$

- **Iteration phase:** Iterate till convergence
  - Repeat for each node $i$
    - Update node vector $a_i$
    - Update label $Y_i$ to $f(a_i)$. This is a hard assignment
  - Iterate until class labels stabilize or max number of iterations is reached

- **Note:** Convergence is not guaranteed
  - Run for max number of iterations
**Example: Web page classification**

- $w_1, w_2, w_3, \ldots$ represent presence of words
- **Baseline**: train a classifier (e.g., k-NN) to classify pages based on words

Wrong. Can we improve?

Same words, but different link structure. Word-based classifier gives same label A to both. Can we use link to improve prediction?
Web page classification example

- Each node maintains a vector of neighborhood labels: \((I_A, I_B, O_A, O_B)\). \(I = \text{In}, O = \text{Out}\)
- \(I_A = 1\) if at least one of the incoming pages is labelled A. Similar definitions for \(I_B, O_A,\) and \(O_B\)

Ground truth: B

Include network features
**Training set**

On a different **training set**, train two classifiers:

1. Word vector only (green circles)
2. Word and link vectors (red circles)

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

---

On a different training set, train two classifiers:
1. Word vector only (green circles)
2. Word and link vectors (red circles)
On test set

Use trained word-vector classifier to bootstrap on test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Ground truth: B

Ground truth: B

Ground truth: B

Ground truth: A
On test set

Use trained word-vector classifier to bootstrap on test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Wrong using word only

Ground truth: A

Ground truth: B

Ground truth: B

Ground truth: B

11/15/18
On test set

1. Train
2. Bootstrap
3. Iterate

Update neighborhood vector for all nodes

a. Update relational features
b. Classify

Ground truth: B

Ground truth: B

Ground truth: A

11/15/18
On test set

1. Train
2. Bootstrap
3. Iterate

Reclassify all nodes

a. Update relational features
b. Classify
On test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Reclassify all nodes

Ground truth: B
10101111

Ground truth: B
10101111

Ground truth: A
10110000

On test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Ground truth: B

Continue till convergence
1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Correct now!
Application of iterative classification framework: fake reviewer/review detection

REV2: Fraudulent User Predictions in Rating Platforms
Kumar et al. ACM Web Search and Data Mining, 2018
Fake review spam

- Review sites are an attractive target for spam: a +1 star increase in rating increases 5-9% revenue!
- Often hype/defame spam
- Paid spammers
Fake review spam detection

- Behavioral analysis
  - individual features, geographic locations, login times, session history, etc.
- Language analysis
  - use of superlatives, many self-referencing, rate of misspell, many agreement words, ...
- Easy to fake!

- Hard to fake: graph structure
  - Graphs capture relationships between reviewers, reviews, stores
**Problem setup**

- **Input**: bipartite rating graph as a weighted signed network:
  - Nodes: users, products
  - Edges: rating scores between -1 and +1

- **Output**: set of users that give fake ratings

Red edges = -1 rating
Green edges = +1 rating
Basic idea: Users, products, and ratings have intrinsic quality scores:
- Users have fairness scores
- Products have goodness scores
- Ratings have reliability scores
- All values are unknown

Each user has a ‘fairness’ score $F(u) \in [0,1]$
Each rating has a ‘reliability’ score $R(u, p) \in [0,1]$

Each product has a ‘goodness’ score $G(p) \in [-1,1]$
**Basic idea:** Users, products, and ratings have intrinsic quality scores:

- Users have fairness scores
- Products have goodness scores
- Ratings have reliability scores

All values are unknown

How can one calculate the values for all nodes and edges simultaneously?

**Solution:** Iterative classification

Each user has a ‘fairness’ score  
\[ F(u) \in [0,1] \]

Each rating has a ‘reliability’ score  
\[ R(u, p) \in [0,1] \]

Each product has a ‘goodness’ score  
\[ G(p) \in [-1,1] \]
Fixing goodness and reliability, fairness is updated as:

\[ F(u) = \frac{\sum_{(u,p) \in \text{Out}(u)} R(u,p)}{|\text{Out}(u)|} \]
Fixing fairness and reliability, goodness is updated as:

\[ G(p) = \frac{\sum_{(u,p) \in \text{In}(p)} R(u,p) \cdot \text{score}(u,p)}{|\text{In}(p)|} \]
Reliability of ratings

- Fixing fairness and goodness, reliability is updated as:

\[ R(u,p) = \frac{1}{y_1 + y_2} (y_1 \cdot F(u) + y_2 \cdot (1 - \frac{|score(u,p) - G(p)|}{2})) \]

How fair is the user who gives the rating

How close is the rating from the goodness of product
Initialization to best scores

\[
G(p) = 1
\]

\[
F(u) = 1
\]

\[
R(u,p) = 1
\]
Updating goodness, iteration 1

\[ G(p) = \frac{\sum_{(u,p) \in \text{In}(p)} R(u,p) \cdot \text{score}(u,p)}{\left| \text{In}(p) \right|} \]
Updating reliability, iteration 1

\[ R(u, p) = \frac{1}{\gamma_1 + \gamma_2} \left( \gamma_1 \cdot F(u) + \gamma_2 \cdot (1 - \frac{|\text{score}(u, p) - G(p)|}{2}) \right) \]

Both gamma values are set to 1
Update fairness, iteration 1

\[ F(u) = \frac{\sum_{(u, p) \in \text{Out}(u)} R(u, p)}{|\text{Out}(u)|} \]
After convergence
Properties of REV2 solution

- Guaranteed to converge
- Number of iterations till convergence is upper-bounded
- Time–complexity: linear
Low fairness users = Fraudsters
127 of 150 lowest fairness users in Flipkart were real fraudsters
Linear scalability

- Multiple iterations, but linear scalability
Collective classification: belief propagation
Collective classification models

- Relational classifiers
- Iterative classification
- Loopy belief propagation
Loopy belief propagation

- Used to estimate marginals (beliefs) or the most likely states of all variables (nodes)
- Iterative process in which neighbor variables “talk” to each other, passing messages

“I (variable $x_1$) believe you (variable $x_2$) belong in these states with various likelihoods...”

- When consensus is reached, calculate final belief
Message passing basics

Task: Count the number of nodes in a graph*

Condition: Each node can only interact (pass message) with its neighbors

Example: straight line graph

* Graph can not have loops. Explanation later.

adapted from MacKay (2003) textbook

11/15/18
**Task**: Count the number of nodes in a graph

**Condition**: Each node can only interact (pass message) with its neighbors

**Solution**: Each node listens to the message from its neighbor, updates it, and passes it forward
Each node only sees incoming messages

Belief: Must be $2 + 1 + 3 = 6$ of us

There's 1 of me

2 before you

Only see my incoming messages

3 behind you
Message passing basics

Each node only sees incoming messages

Belief: Must be 1 + 1 + 4 = 6 of us
Belief: Must be 1 + 1 + 3 = 6 of us

4 behind you

1 before you

only see my incoming messages

there's 1 of me

Message passing in a tree

Each node receives reports from all branches of the tree.
Message passing in a tree

Each node receives reports from all branches of tree

7 here
(= 3+3+1)

3 here

3 here
Message passing in a tree

Each node receives reports from all branches of tree

11 here

11 here

(= 7 + 3 + 1)

7 here

3 here
Message passing in a tree

Each node receives reports from all branches of tree

Belief: Must be 14 of us
Message passing in a tree

Each node receives reports from all branches of tree.

Belief:
Must be 14 of us wouldn't work correctly with a 'loopy' (cyclic) graph.
What message will I send to j?
- It depends on what i hears from its neighbors k
- Each neighbor k passes a message to i its beliefs of the state to i

“i (variable x1) believe you (variable x2) belong in these states with various likelihoods...”
Notations

- **Label-label potential matrix** \( \psi \): Dependency between a node and its neighbor. \( \psi(Y_i, Y_j) \) equals the probability of a node \( i \) being in state \( Y_i \) given that it has a \( j \) neighbor in state \( Y_j \).
- **Prior belief** \( \phi \): Probability \( \phi_i(Y_i) \) of node \( i \) being in state \( Y_i \).
- \( m_{i \rightarrow j}(Y_j) \) is \( i \)'s estimate of \( j \) being in state \( Y_j \).
- \( \mathcal{L} \) is the set of all states.
1. Initialize all messages to 1
2. Repeat for each node

\[ m_{i \rightarrow j}(Y_j) = \alpha \sum_{Y_i \in \mathcal{L}} \psi(Y_i, Y_j) \phi_i(Y_i) \prod_{k \in \mathcal{N}_i \setminus j} m_{k \rightarrow i}(Y_i) \]

Sum over all states
After convergence:
\[ b_i(Y_i) = \text{'i's belief of being in state } Y_i \]

\[ b_i(Y_i) = \alpha \phi_i(Y_i) \prod_{j \in \mathcal{N}_i} m_{j \rightarrow i}(Y_i), \forall Y_i \in \mathcal{L} \]
Loopy belief propagation

What if our graph has cycles?

- Messages from different subgraphs are no longer independent!

- But we can still run BP -- it's a local algorithm so it doesn't "see the cycles."
What can go wrong with belief propagation?

• Messages loop around and around: 2, 4, 8, 16, 32, ... More and more convinced that these variables are T!

• BP incorrectly treats this message as separate evidence that the variable is T.

• Multiplies these two messages as if they were independent.
  • But they don’t actually come from independent parts of the graph.
  • One influenced the other (via a cycle).

This is an extreme example. Often in practice, the cyclic influences are weak. (As cycles are long or include at least one weak correlation.)
Advantages of belief propagation

- **Advantages:**
  - Easy to program & parallelize
  - General: can apply to any graphical model w/ any form of potentials (higher order than pairwise)

- **Challenges:**
  - Convergence is not guaranteed (when to stop), especially if many closed loops
  - **Potential functions** (parameters)
    - require training to estimate
    - learning by gradient-based optimization: convergence issues during training
Application of belief propagation: Online auction fraud

Netprobe: A Fast and Scalable System for Fraud Detection in Online Auction Networks
Pandit et al., World Wide Web conference 2007
Online auction fraud

- Auction sites: attractive target for fraud
- 63% complaints to Federal Internet Crime Complaint Center in U.S. in 2006
- Average loss per incident: = $385
- Often non-delivery fraud:

![Diagram showing seller and buyer with an arrow indicating transaction and a cross indicating non-delivery fraud.](image)
Online auction fraud detection

- **Insufficient solution** to look at individual features: user attributes, geographic locations, login times, session history, etc.
- **Hard to fake**: graph structure
- Capture relationships between users

- **Main question**: how do fraudsters interact with other users and among each other?
  - In addition to buy/sell relations, are there more complex relations?
Each user has a reputation score
Users rate each other via feedback

- Question: How do fraudsters game the feedback system?
Auction “roles” of users

- Do they boost each other’s reputation?
  - No, because if one is caught, all will be caught

- They form near-bipartite cores (2 roles)
  - **Accomplice**: trades with honest, looks legit
  - **Fraudster**: trades with accomplice, fraud with honest
Detecting auction fraud

- How to find near-bipartite cores? How to find roles (honest, accomplice, fraudster)?
  - Use belief propagation!
- How to set BP parameters (potentials)?
  - prior beliefs: prior knowledge, unbiased if none
  - compatibility potentials: by insight

<table>
<thead>
<tr>
<th></th>
<th>Fraud</th>
<th>Accomplice</th>
<th>Honest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraud</td>
<td>$\varepsilon_p$</td>
<td>$1 - 2\varepsilon_p$</td>
<td>$\varepsilon_p$</td>
</tr>
<tr>
<td>Accomplice</td>
<td>0.5</td>
<td>$2\varepsilon_p$</td>
<td>$0.5 - 2\varepsilon_p$</td>
</tr>
<tr>
<td>Honest</td>
<td>$\varepsilon_p$</td>
<td>$(1 - 2\varepsilon_p)/2$</td>
<td>$(1 - 2\varepsilon_p)/2$</td>
</tr>
</tbody>
</table>
Belief propagation in action

Initialize all nodes as unbiased
Initialize all nodes as unbiased

At each iteration, for each node, compute messages to its neighbors
Belief propagation

1. Initialize all nodes as unbiased

2. At each iteration, for each node, compute messages to its neighbors

3. Continue till convergence
Final belief scores = final roles

P(fraudster)
P(associate)
P(honest)
Three collective classification algorithms:

- Simple relational models:
  - Weighted average of neighborhood properties
  - Can not take node attributes while labeling

- Iterative classification
  - Update each node’s label using own and neighbor’s labels
  - Can consider node attributes while labeling

- Belief propagation
  - Message passing to update each node’s belief of itself based on neighbors’ beliefs