INTRO TO SEMI-SUPERVISED LEARNING (SSL)
Semi-supervised learning

- Given:
  - A pool of labeled examples $L$
  - A (usually larger) pool of unlabeled examples $U$

- Option 1 for using $L$ and $U$:
  - Ignore $U$ and use supervised learning on $L$

- Option 2:
  - Ignore labels in $L+U$ and use k-means, etc find clusters; then label each cluster using $L$

- Question:
  - Can you use both $L$ and $U$ to do better?
SSL is Somewhere Between Clustering and Supervised Learning
SSL is Between Clustering and SL
What is a natural grouping among these objects?

Clustering is subjective

Simpson's Family  School Employees  Females  Males

slides: Bhavana Dalvi
SSL is Between Clustering and SL

Clustering is unconstrained and may not give you what you want.

Maybe this clustering is as good as the other.
SSL is Between Clustering and SL
SSL is Between Clustering and SL
SSL is Between Clustering and SL

supervised learning with few labels is also unconstrained and may not give you what you want
SSL is Between Clustering and SL
SSL is **Between Clustering and SL**

This clustering isn’t consistent with the **labels**
SSL is **Between Clustering and SL**

$|\text{Predicted Green}|/|U| \approx 50\%$
SSL in Action: The NELL System
Type of SSL

- Margin-based: transductive SVM
  - Logistic regression with entropic regularization
- Generative: seeded k-means
- Nearest-neighbor like: graph-based SSL
  - Label propagation
SSL via “Label Propagation”
Semi-Supervised Classification of Network Data Using Very Few Labels

Frank Lin  
Carnegie Mellon University, Pittsburgh, Pennsylvania  
Email: frank@cs.cmu.edu

William W. Cohen  
Carnegie Mellon University, Pittsburgh, Pennsylvania  
Email: wcohen@cs.cmu.edu

ASONAM-2010 (Advances in Social Networks Analysis and Mining)
Network Datasets with Known Classes

- UBMCBlog
- AGBlog
- MSPBlog
- Cora
- Citeseer
Some intuition
Given: A graph $G = (V, E)$, corresponding to nodes in $G$ are instances $X$, composed of unlabeled instances $X^U$ and labeled instances $X^L$ with corresponding labels $Y^L$, and a damping factor $d$.

Returns: Labels $Y^U$ for unlabeled nodes $X^U$.

For each class $c$
1. Set $u_i \leftarrow 1$, $\forall Y_i^L = c$
2. Normalize $u$ such that $\|u\|_1 = 1$
3. Set $R_c \leftarrow \text{RandomWalk}(G, u, d)$

For each instance $i$
- Set $X^U_i \leftarrow \arg \max_c (R_{ci})$

Fig. 1. The MultiRankWalk algorithm.

Seed selection
1. order by PageRank, degree, or randomly
2. go down list until you have at least $k$ examples/class

$u$ is uniform over the seeds for class $c$

$RWR$ - fixpoint of:
$$r = (1 - d)u + dW_r$$
Some intuition
Results – Blog data

We’ll discuss this soon….
Results – More blog data

Random

Degree

PageRank
Results – Citation data

Random

Degree

PageRank
Seeding – MultiRankWalk
Seeding – HF/wvRN

![Graphs showing F1 scores for different seeding methods over iterations for UMBCBlog, AGBlog, MSPBlog, and Cora datasets.](image-url)
Back to Experiments: Network Datasets with Known Classes

- UBMCBlog
- AGBlog
- MSPBlog
- Cora
- Citeseer
MultiRankWalk vs wvRN/HF/CoEM

Figure 2.6: Scatter plots of HF F1 score versus MRW F1 score. The left plot marks different seeding preferences and the right plot marks varying amount of training labels determined by $m$. 
How well does MWR work?

Fig. 5. Citation datasets results compared to supervised relational learning methods. The x-axis indicates number of labeled instances and y-axis indicates labeling accuracy.
Fig. 7. Results on three datasets varying the damping factor. The x-axis indicates number of labeled instances and y-axis indicates labeling macro-averaged F1 score.
Harmonic Fields aka coEM aka wvRN
CoEM/HF/wvRN

• One definition [MacKassey & Provost, JMLR 2007]:...

**Definition.** Given \( v_i \in V^U \), the weighted-vote relational-neighbor classifier (wvRN) estimates \( P(x_i | \mathcal{N}_i) \) as the (weighted) mean of the class-membership probabilities of the entities in \( \mathcal{N}_i \):

\[
P(x_i = c | \mathcal{N}_i) = \frac{1}{Z} \sum_{v_j \in \mathcal{N}_i} w_{i,j} \cdot P(x_j = c | \mathcal{N}_j),
\]

Another definition: A harmonic field (HF) – the score of each node in the graph is the harmonic (linearly weighted) average of its neighbors’ scores --- also sometimes called LP-ZGL

[X. Zhu, Z. Ghahramani, and J. Lafferty, ICML 2003]
Co-EM Learner: equivalent to HF on a bipartite graph (Ghani & Nigam, 2000)
The HF Algorithm

\[ \{(x^1, y^1), \ldots, (x^m, y^m)\} \quad \text{= labeled examples} \]
\[ \{x^{m+1}, \ldots, x^{m+n}\} \quad \text{= unlabeled examples} \]
\[ W[i, j] \quad \text{= graph = similarity between } x_i \text{ and } x_j \]

Optimization problem: minimize

\[
\text{Loss} = \sum_{i>m, j>m} W[i, j](\hat{y}_i - \hat{y}_j)^2
\]

subject to constraint that all labeled examples are classified correctly
The HF Loss In Matrix Form

\[ W[i, j] = \text{graph} = \text{similarity between } x_i \text{ and } x_j \text{ is symmetric} \]
\[ \text{Loss} = \sum_{i,j} w_{i,j} (\hat{y}_i - \hat{y}_j)^2 \]
\[ = \sum_{i,j} w_{i,j} \hat{y}_i^2 + \sum_{i,j} w_{i,j} \hat{y}_j^2 - 2 \sum_{i,j} w_{i,j} \hat{y}_i \hat{y}_j \]
\[ = \sum_i (\sum_j w_{i,j}) \hat{y}_i^2 + \sum_j (\sum_i w_{i,j}) \hat{y}_j^2 - 2 \sum_{i,j} w_{i,j} \hat{y}_i \hat{y}_j \]
\[ = 2 \sum_i d_i \hat{y}_i^2 - 2 \sum_{i,j} w_{i,j} \hat{y}_i \hat{y}_j \]
\[ = 2(\hat{y}^T D \hat{y} - \hat{y}^T W \hat{y}) \]
\[ = 2\hat{y}^T (D - W) \hat{y} \]
The HF Algorithm

\{ (x^1, y^1), \ldots, (x^m, y^m) \} = \text{labeled examples}
\{ x^{m+1}, \ldots, x^{m+n} \} = \text{unlabeled examples}

W[i, j] = \text{graph} = \text{similarity between } x_i \text{ and } x_j

S[i, i] = 1 \text{ for all seed nodes } i \leq m+1

Optimization problem: minimize \( \hat{y}^T (D - W) \hat{y} \)
subject to \( S\hat{y} = Sy \)
The HF Algorithm

Optimization problem: minimize

$$\hat{y}^T (D - W) \hat{y}$$

subject to

$$S \hat{y} = Sy$$

1. Let $\hat{y}^0$ be any label assignment consistent with the seed labels.

2. For $t = 0, \ldots, T$:

   (a) For every unlabeled node $i > m$, let $\hat{y}_{i}^{t+1} = \frac{1}{d_i} \sum_j w_{i,j} \hat{y}_{j}^{t}$

   (b) For every labeled node $i \leq m$, let $\hat{y}_{i}^{t+1} = y_i$ (where $y_i$ is the seed label for example $i$).

This converges quickly: on Frank’s data usually 5-10 iterations was best (and more tends to overfit)
What is HF aka coEM aka wvRN?

Algorithmically:

- HF propagates weights and then resets the seeds to their initial value
- MRW propagates weights and does not reset seeds

$$P(x_i = c | \mathcal{N}_i) = \frac{1}{Z} \sum_{v_j \in \mathcal{N}_i} w_{i,j} \cdot P(x_j = c | \mathcal{N}_j),$$
MultiRank Walk vs HF/wvRN/CoEM

Seeds are marked S

HF

MRW
MultiRank Walk vs HF/wvRN/CoEM
SSL as optimization and Modified Adsorption slides from Partha Talukdar
\( \hat{Y}_v,l \) : score of estimated label \( l \) on node \( v \)

\( Y_v,l \) : score of seed label \( l \) on node \( v \)

\( R_v,l \) : regularization target for label \( l \) on node \( v \)

\( S \) : seed node indicator (diagonal matrix)

\( W_{uv} \) : weight of edge \( (u, v) \) in the graph
LP-ZGL (Zhu et al., ICML 2003) yet another name for HF/wvRN/coEM

\[
\begin{align*}
\arg\min_{\hat{Y}} \quad & \sum_{l=1}^{m} W_{uv} (\hat{Y}_{ul} - \hat{Y}_{vl})^2 \\
\text{such that} \quad & Y_{ul} = \hat{Y}_{ul}, \quad \forall S_{uu} = 1
\end{align*}
\]

Smooth

Match Seeds (hard)

\[
\sum_{l=1}^{m} \hat{Y}_l^T L \hat{Y}_l
\]

Graph Laplacian \( L = D - W \) (PSD)

- Smoothness
  - two nodes connected by an edge with high weight should be assigned similar labels

- Solution satisfies harmonic property
Modified Adsorption (MAD)  
[Talukdar and Crammer, ECML 2009]

\[ \arg \min_{\hat{Y}} \sum_{l=1}^{m+1} \left[ \| S \hat{Y}_l - SY_l \|^2 + \mu_1 \sum_{u,v} M_{uv}(\hat{Y}_{ul} - \hat{Y}_{vl})^2 + \mu_2 \| \hat{Y}_l - R_l \|^2 \right] \]

- \( m \) labels, +1 dummy label
- \( M = W^T + W' \) is the symmetrized weight matrix
- \( \hat{Y}_{vl} \): weight of label \( l \) on node \( v \)
- \( Y_{vl} \): seed weight for label \( l \) on node \( v \)
- \( S \): diagonal matrix, nonzero for seed nodes
- \( R_{vl} \): regularization target for label \( l \) on node \( v \)
• $M = \mathbf{W}^\dagger + \mathbf{W}'$ is the symmetrized weight matrix

**Adsorption SSL algorithm**

- Continue walk with prob. $p_v^{cont}$
- Assign V’s seed label to U with prob. $p_v^{inj}$
- Abandon random walk with prob. $p_v^{abnd}$
  - assign U a *dummy label*
$M = W^\dagger + W'$ is the symmetrized weight matrix

**Random Walk View**

- Continue walk with prob. $p_v^{\text{cont}}$
- Assign V's seed label to U with prob. $p_v^{\text{inj}}$
- Abandon random walk with prob. $p_v^{\text{abnd}}$
  - assign U a dummy label

New Edge Weight

$W'_{uv} = p_u^{\text{cont}} \times W_{uv}$

$S_{uu} = \sqrt{p_u^{\text{inj}}}$

$R_{uU} = p_u^{\text{abnd}}$, and 0 for non-dummy labels

Dummy Label
Modified Adsorption (MAD)  
[Talukdar and Crammer, ECML 2009]

$$\arg \min_{\hat{Y}} \sum_{l=1}^{m+1} \left[ \| S\hat{Y}_l - SY_l \|^2 + \mu_1 \sum_{u,v} M_{uv} (\hat{Y}_{ul} - \hat{Y}_{vl})^2 + \mu_2 \| \hat{Y}_l - R_l \|^2 \right]$$

- $m$ labels, +1 dummy label
- $M = W^T + W'$ is the symmetrized weight matrix
- $\hat{Y}_{vl}$: weight of label $l$ on node $v$
- $Y_{vl}$: seed weight for label $l$ on node $v$
- $S$: diagonal matrix, nonzero for seed nodes
- $R_{vl}$: regularization target for label $l$ on node $v$
How to do this minimization?
First, differentiate to find min is at

\[
(\mu_1 S + \mu_2 L + \mu_3 I) \hat{Y}_l = (\mu_1 SY_l + \mu_3 R_l).
\]

The minimize with *Jacobi method* (which works for linear matrix equations like this one)
MapReduce Implementation of MAD

• Map
  – Each node send its current label assignments to its neighbors

• Reduce
  – Each node updates its own label assignment using messages received from neighbors, and its own information (e.g., seed labels, reg. penalties etc.)

• Repeat until convergence

Code in Junto Label Propagation Toolkit (includes Hadoop-based implementation)
http://code.google.com/p/junto/
Text Classification

PRBEP (macro-averaged) on WebKB Dataset, 3148 test instances

- k-NN graph

precision-recall break even point
k-NN graph

Sentiment Classification

Precision on 3568 Sentiment test instances
Class-Instance Acquisition

Freebase-2 Graph, 192 WordNet Classes

Mean Reciprocal Rank (MRR)

- LP-ZGL
- Adsorption
- MAD

Amount of Supervision

192 x 2
192 x 10

Graph with 303k nodes, 2.3m edges.
Assigning class labels to WebTable instances from HTML tables on the web that are used for data, not formatting.

<table>
<thead>
<tr>
<th>Year</th>
<th>Artist</th>
<th>Albums</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Johnny Cash</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bob Dylan</td>
<td></td>
</tr>
</tbody>
</table>

from mining patterns like “musicians such as Bob Dylan”

Score (musician, Johnny Cash) = 0.87
## New (Class, Instance) Pairs Found

<table>
<thead>
<tr>
<th>Class</th>
<th>A few non-seed Instances found by Adsorption</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFL Players</td>
<td>Tony Gonzales, Thabiti Davis, Taylor Stubblefield, Ron Dixon, Rodney Hannan, …</td>
</tr>
</tbody>
</table>

Total classes: 9081
Scaling up Graph SSL
Followup work (AIStats 2014)

- Propagating labels requires usually small number of optimization passes
  - Basically like label propagation passes
- Each is linear in
  - the number of edges
  - and the number of labels being propagated
- Can you do better?
  - basic idea: store labels in a countmin sketch
  - which is basically an compact approximation of an object→double mapping
Count-min sketches

split a \textit{real} vector into \(k\) ranges, one for each hash function

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“fred flintstone”, 3)}:

\[
\begin{array}{cccccc}
0 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{add the value to each hash location}

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“barney rubble”, 5)}:

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

add the value to each hash location

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

split a \textit{real} vector into \(k\) ranges, one for each hash function

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“fred flintstone”, 3)}:

\[
\begin{array}{cccccc}
0 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{add the value to each hash location}

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“barney rubble”, 5)}:

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

add the value to each hash location

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

split a \textit{real} vector into \(k\) ranges, one for each hash function

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“fred flintstone”, 3)}:

\[
\begin{array}{cccccc}
0 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{add the value to each hash location}

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\texttt{cm.inc(“barney rubble”, 5)}:

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]

add the value to each hash location

\[
\begin{array}{cccccc}
5 & 3 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Count-min sketches

Split a real vector into k ranges, one for each hash function

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

cm.get(“fred flintstone”): 3

\[
\begin{array}{cccccc}
5 & 3 & 0 & 8 & 0 & 0 \\
\end{array}
\]

h1 \rightarrow 5, h2 \rightarrow 8, h3 \rightarrow 5

take min when retrieving a value

cm.get(“barney rubble”): 5

\[
\begin{array}{cccccc}
5 & 3 & 0 & 8 & 0 & 0 \\
\end{array}
\]

h1 \rightarrow 5, h2 \rightarrow 8, h3 \rightarrow 5
Followup work (AIStats 2014)

• Propagating labels requires usually small number of optimization passes
  – Basically like label propagation passes
• Each is linear in
  – the number of edges
  – and the number of labels being propagated
  – the sketch size
– sketches can be combined linearly without “unpacking” them: sketch($av + bw$) = $a \cdot \text{sketch}(v) + b \cdot \text{sketch}(w)$
– sketches are good at storing skewed distributions
Followup work (AIStats 2014)

• Label distributions are often very skewed
  – sparse initial labels
  – community structure: labels from other subcommunities have small weight
Followup work (AIStats 2014)

“self-injection”: similarity computation

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes (n)</th>
<th>Edges</th>
<th>Labels (m)</th>
<th>Seed Nodes</th>
<th>k–Sparsity</th>
<th>$\left\lceil \frac{ek}{c} \right\rceil$</th>
<th>$\left\lfloor \ln \frac{m}{\delta} \right\rfloor$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>301,638</td>
<td>1,155,001</td>
<td>192</td>
<td>1917</td>
<td>2</td>
<td>109</td>
<td>8</td>
</tr>
<tr>
<td>Flickr-10k</td>
<td>41,036</td>
<td>73,191</td>
<td>10,000</td>
<td>10,000</td>
<td>1</td>
<td>55</td>
<td>12</td>
</tr>
<tr>
<td>Flickr-1m</td>
<td>1,281,887</td>
<td>7,545,451</td>
<td>1,000,000</td>
<td>1,000,000</td>
<td>1</td>
<td>55</td>
<td>17</td>
</tr>
</tbody>
</table>

Freebase

Flickr-10k
### Followup work (AIStats 2014)

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes ($n$)</th>
<th>Edges</th>
<th>Labels ($m$)</th>
<th>Seed Nodes</th>
<th>$k$-Sparsity</th>
<th>$\frac{ek}{\epsilon}$</th>
<th>$\ln \frac{m}{\delta}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>301,638</td>
<td>1,155,001</td>
<td>192</td>
<td>1917</td>
<td>2</td>
<td>109</td>
<td>8</td>
</tr>
<tr>
<td>Flickr-10k</td>
<td>41,036</td>
<td>73,191</td>
<td>10,000</td>
<td>10,000</td>
<td>1</td>
<td>55</td>
<td>12</td>
</tr>
<tr>
<td>Flickr-1m</td>
<td>1,281,887</td>
<td>7,545,451</td>
<td>1,000,000</td>
<td>1,000,000</td>
<td>1</td>
<td>55</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Average Memory Usage (GB)</th>
<th>Total Runtime (s) [Speedup w.r.t. MAD-EXACT]</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAD-EXACT</td>
<td></td>
<td>516.63 [1.0]</td>
<td>0.28</td>
</tr>
<tr>
<td>MAD-SKETCH ($w = 109, d = 8$)</td>
<td>3.54</td>
<td>110.42 [4.7]</td>
<td>0.28</td>
</tr>
<tr>
<td>MAD-SKETCH ($w = 109, d = 3$)</td>
<td>1.37</td>
<td>54.45 [9.5]</td>
<td>0.29</td>
</tr>
<tr>
<td>MAD-SKETCH ($w = 20, d = 8$)</td>
<td>1.06</td>
<td>47.72 [10.8]</td>
<td>0.28</td>
</tr>
<tr>
<td>MAD-SKETCH ($w = 20, d = 3$)</td>
<td>1.12</td>
<td>48.03 [10.8]</td>
<td>0.23</td>
</tr>
</tbody>
</table>
**Followup work (AIStats 2014)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes ($n$)</th>
<th>Edges</th>
<th>Labels (m)</th>
<th>Seed Nodes</th>
<th>$k$—Sparsity</th>
<th>$\frac{ek}{\epsilon}$</th>
<th>$\ln \frac{m}{\delta}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>301,638</td>
<td>1,155,001</td>
<td>192</td>
<td>1917</td>
<td>2</td>
<td>109</td>
<td>8</td>
</tr>
<tr>
<td>Flickr-10k</td>
<td>41,036</td>
<td>73,191</td>
<td>10,000</td>
<td>10,000</td>
<td>1</td>
<td>55</td>
<td>12</td>
</tr>
<tr>
<td>Flickr-1m</td>
<td>1,281,887</td>
<td>7,545,451</td>
<td>1,000,000</td>
<td>1,000,000</td>
<td>1</td>
<td>55</td>
<td>17</td>
</tr>
</tbody>
</table>

100 Gb available
Even more recent work

Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation

Sujith Ravi  
Google Inc., Mountain View, CA, USA  
sravi@google.com

Qiming Diao
Carnegie Mellon University, Pittsburgh, PA, USA  
Singapore Mgt. University, Singapore  
qiming.ustc@gmail.com

AIStats 2016
Differences: objective function

\[ C(\hat{Y}) = \mu_1 \sum_{v \in V_l} s_{vv} ||\hat{Y}_v - Y_v||_2^2 \]
\[ + \mu_2 \sum_{v \in V, u \in \mathcal{N}(v)} w_{vu} ||\hat{Y}_v - \hat{Y}_u||^2 \]
\[ + \mu_3 \sum_{v \in V} ||\hat{Y}_v - U||_2^2 \]
\[ s.t. \sum_{l=1}^{L} \hat{Y}_{vl} = 1, \forall v \]

seeds
smoothness
close to uniform label distribution
normalized predictions
Differences: scaling up

• Updates done in parallel with Pregel
• Replace count-min sketch with “streaming approach”
  – updates from neighbors are a “stream”
    • break stream into “sections”
      – maintain a list of \((y, \text{Prob}(y), \Delta)\)
      – filter out labels at end of “section” if \(\text{Prob}(y) + \Delta\) is small
Results with EXPANDER