Semi-supervised learning
SSL (on graphs)
Announcement

• No office hour for William after class today!
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
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 and SL
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
Harmonic Fields
aka coEM aka wvRN…
• Idea: construct a graph connecting the most similar examples (k-NN graph)
• Intuition: nearby points should have similar labels – labels should “propagate” through the graph
• Formalization: try and minimize “energy” defined as:

energy: \( E(y) = \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2 \)
Result 1: at the minimal energy state, each node’s value is a **weighted average** of its neighbor’s weights:

\[
\Delta f = 0 \quad \text{or} \quad f_i = \frac{\sum_{j \sim i} w_{ij} f_j}{\sum_{j \sim i} w_{ij}}, \quad i \in U
\]

**energy:** \( E(y) = \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2 \)

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

happy, low energy

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

unhappy, high energy
“Harmonic field” LP algorithm

• Result 2: you can reach the minimal energy state with a simple iterative algorithm:
  – Step 1: For each seed example \((x_i, y_i)\):
    • Let \(V^0(i,c) = [\mid y_i = c \mid]\)
  – Step 2: for \(t=1,\ldots,T\)    ---  \(T\) is about 5
    • Let \(V^{t+1}(i,c) =\) weighted average of \(V^{t+1}(j,c)\) for all \(j\) that are linked to \(i\), and renormalize
      \[
      V^{t+1}(i,c) = \frac{1}{Z} \sum_j w_{i,j} V^t(j,c)
      \]
    • For seeds, reset \(V^{t+1}(i,c) = [\mid y_i = c \mid]\)
This family of techniques is called “Label propagation”
This experiment points out some of the issues with LP:
1. What distance metric do you use?
2. What energy function do you minimize?
3. What is the right value for K in your K-NN graph? Is a K-NN graph right?
4. If you have lots of data, how expensive is it to build the graph?
NELL: Uses Co-EM $\sim=$ HF

Extract cities:
- Paris
- Pittsburgh
- Seattle
- Cupertino

Examples
- San Francisco
- Austin
- denial

Features
- mayor of \texttt{arg1}
- live in \texttt{arg1}

\texttt{arg1} is home of traits such as \texttt{arg1}

anxiety
selfishness
Berlin
Semi-Supervised Bootstrapped Learning via Label Propagation
Semi-Supervised Bootstrapped Learning via Label Propagation

Paris

San Francisco

Austin

Pittsburgh

Seattle

mayor of arg1

arg1 is home of

Information from other categories tells you “how far” (when to stop propagating)

traits such as arg1

traits such as arg1

denial

Nodes “near” seeds

Nodes “far from” seeds

24
Difference: graph construction is not instance-to-instance but instance-to-feature.

Important reformulation: the k-NN graph is expensive to build, the instance-feature graph may not be.
Some other general issues with SSL

• How much unlabeled data do you want?
  – Suppose you’re optimizing $J = J_L(L) + J_U(U)$
  – If $|U| >> |L|$ does $J_U$ dominate $J$?
    • If so you’re basically just clustering
  – Often we need to balance $J_L$ and $J_U$

• Besides L, what other information about the task is useful (or necessary)?
  – Common choice: relative frequency of classes
  – Various ways of incorporating this into the optimization problem
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
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 $u_c \leftarrow 1$, $\forall Y_i^L = c$
2) Normalize $u$ such that $||u||_1 = 1$
3) Set $R_c \leftarrow \text{RandomWalk}(G, u, d)$

For each instance $i$
- Set $X^U_i \leftarrow \text{argmax}_c(R_{ci})$

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
Results – Blog data

HF method
Results – More blog data

Random

Degree

PageRank

MSPBlog Random Seeding F1

MSPBlog LinkCount Seeding F1

MSPBlog PageRank Seeding F1
Results – Citation data

Random

Degree

PageRank
Seeding – MultiRankWalk
Seeding – HF/wvRN
MultiRank Walk vs HF/wvRN/CoEM

Seeds are marked S

HF

MRW
Back to Experiments: Network Datasets with Known Classes

- UBMCBlog
- AGBlog
- MSPBlog
- Cora
- Citeseer
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 $m$. 
Harmonic Fields
aka coEM aka wvRN

**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 $\mathcal{N}_i$:

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

“Simple relational classifier” is same as the harmonic field – the score of each node in the graph is the harmonic (linearly weighted) average of its neighbors’ scores.
• Another justification of the same algorithm goes back to 2003

• ... start with co-training with a naïve Bayes learner

• **Inputs:** An initial collection of labeled documents and one of unlabeled documents.

• Loop while there exist documents without class labels:
  - Build classifier A using the A portion of each document.
  - Build classifier B using the B portion of each document.
  - For each class C, pick the unlabeled document about which classifier A is most confident that its class label is C and add it to the collection of labeled documents.
  - For each class C, pick the unlabeled document about which classifier B is most confident that its class label is C and add it to the collection of labeled documents.

• **Output:** Two classifiers, A and B, that predict class labels for new documents. These predictions can be combined by multiplying together and then renormalizing their class probability scores.

Table 1: The co-training algorithm described in Section 3.3.
CoEM/wvRN/HF

- One algorithm with several justifications....

- One is to start with co-training with a naïve Bayes learner

- And compare to an EM version of naïve Bayes
  - E: soft-classify unlabeled examples with NB classifier
  - M: re-train classifier with soft-labeled examples

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Labeled</th>
<th># Unlabeled</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>788</td>
<td>–0–</td>
<td>3.3%</td>
</tr>
<tr>
<td>Co-training</td>
<td>12</td>
<td>776</td>
<td>5.4%</td>
</tr>
<tr>
<td>EM</td>
<td>12</td>
<td>776</td>
<td>4.3%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>12</td>
<td>–0–</td>
<td>13.0%</td>
</tr>
</tbody>
</table>
CoEM/wvRN/HF

- A second experiment
  - each + example: concatenate features from two documents, one of class A+, one of class B+
  - each - example: concatenate features from two documents, one of class A-, one of class B-
  - features are prefixed with “A”, “B” ➔ disjoint

Table 3: The setup of the News 2x2 dataset. This data has class-conditional independence and redundancy between its two feature sets.

<table>
<thead>
<tr>
<th>Class</th>
<th>Feature Set A</th>
<th>Feature Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>comp.os.ms-windows.misc</td>
<td>talk.politics.misc</td>
</tr>
<tr>
<td>Neg</td>
<td>comp.sys.ibm.pc.hardware</td>
<td>talk.politics.guns</td>
</tr>
</tbody>
</table>
CoEM/wvRN/HF

- A second experiment
  - each + example: concatenate features from two documents, one of class A+, one of class B+
  - each - example: concatenate features from two documents, one of class A-, one of class B-
  - features are prefixed with “A”, “B” \(\Rightarrow\) disjoint

- NOW co-training outperforms EM

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Labeled</th>
<th># Unlabeled</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>1006</td>
<td>-0-</td>
<td>3.9%</td>
</tr>
<tr>
<td>Co-training</td>
<td>6</td>
<td>1000</td>
<td>3.7%</td>
</tr>
<tr>
<td>EM</td>
<td>6</td>
<td>1000</td>
<td>8.9%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>6</td>
<td>-0-</td>
<td>34.0%</td>
</tr>
</tbody>
</table>
CoEM/wvRN/HF

- Co-training with a naïve Bayes learner

vs an EM version of naïve Bayes
  - E: soft-classify unlabeled examples with NB classifier
  - M: re-train classifier with soft-labeled examples

<table>
<thead>
<tr>
<th>Method</th>
<th>Uses Feature Split?</th>
<th>Uses Random Feature Split?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Incremental</td>
<td>co-training</td>
<td>self-training</td>
</tr>
<tr>
<td>Iterative</td>
<td>co-EM</td>
<td>EM</td>
</tr>
</tbody>
</table>

incremental hard assignments
iterative soft assignments
Co-EM for a Rote Learner: equivalent to HF on a bipartite graph
SSL AS OPTIMIZATION
SSL as optimization and Modified Adsorption slides from Partha Talukdar
Notations

\( \hat{Y}_{v,l} \) : 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

\[
\arg \min_{\hat{Y}} \sum_{l=1}^{m} W_{uv}(\hat{Y}_{ul} - \hat{Y}_{vl})^2 = \sum_{l=1}^{m} \hat{Y}_l^T L \hat{Y}_l
\]

such that

\[
Y_{ul} = \hat{Y}_{ul}, \ \forall S_{uu} = 1
\]

Smooth

Match Seeds (hard)

- 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
- \( M = W^\top + W' \) 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 \( v \)
- \( M = W^{t+} + W' \) is the symmetrized weight matrix

**Adsorption SSL algorithm**

- Continue walk with prob. \( p^\text{cont}_v \)
- Assign \( V \)'s seed label to \( U \) with prob. \( p^\text{inj}_v \)
- Abandon random walk with prob. \( p^\text{abnd}_v \)
  - assign \( U \) a **dummy label**
- \( M = W^T + W' \) 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

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

\[
S_{uu} = \sqrt{p_u^{\text{inj}}}
\]

\[
R_{u^\top} = p_u^{\text{abnd}}, \text{ and 0 for non-dummy labels}
\]

Dummy Label
Modified Adsorption (MAD)

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

- \( m \) labels, +1 dummy label
- \( M = W^\top + W' \) 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 \( v \)

Seed Scores
Label Priors
Estimated Scores
How to do this minimization?
First, differentiate to find min is at

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

Jacobi method:
- To solve \( Ax=b \) for \( x \)
- Iterate:
  \[
x^{(k+1)} = D^{-1} (b - Rx^{(k)})
  \]
  \[
x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.
  \]
Inputs $Y, R: |V| \times (|L| + 1)$, $W: |V| \times |V|$, $S: |V| \times |V|$ diagonal

$\hat{Y} \leftarrow Y$

$M = W' + W^\dagger$

$Z_v \leftarrow S_{vv} + \mu_1 \sum_{u \neq v} M_{vu} + \mu_2 \quad \forall v \in V$

repeat

for all $v \in V$ do

$\hat{Y}_v \leftarrow \frac{1}{Z_v} \left( (SY)_v + \mu_1 M_v.\hat{Y} + \mu_2 R_v \right)$

end for

until convergence

- Extends Adsorption with well-defined optimization
- Importance of a node can be discounted
- Easily Parallelizable: Scalable
MapReduce Implementation of MAD

• Map
  – Each node sends 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
Sentiment Classification

Precision on 3568 Sentiment test instances
Class-Instance Acquisition

Freebase-2 Graph, 192 WordNet Classes

Mean Reciprocal Rank (MRR)

LP-ZGL /HF/… 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

from mining patterns like “musicians such as Bob Dylan”

WebTable

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

A8

<table>
<thead>
<tr>
<th>musician</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bob Dylan</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Johnny Cash

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, Thabití Davis, Taylor Stubblefield, Ron Dixon, Rodney Hannan, …</td>
</tr>
</tbody>
</table>

Total classes: 9081
MAD SKETCHES
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 real vector into k ranges, one for each hash function

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0
\end{array}
\]

cm.inc("fred flintstone", 3):

\[
\begin{array}{cccccccccc}
0 & 3 & 0 & | & 3 & 0 & 0 & | & 0 & 3 & 0
\end{array}
\]

add the value to each hash location

cm.inc("barney rubble", 5):

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

take min when retrieving a value

$$\begin{array}{cccccccc}
5 & 3 & 0 & 8 & 0 & 0 & 5 & 3 & 0
\end{array}$$

cm.get("barney rubble"): 5

$$\begin{array}{cccccccc}
5 & 3 & 0 & 8 & 0 & 0 & 5 & 3 & 0
\end{array}$$
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
  – the number of labels being propagated
  – the sketch size
  – sketches can be combined linearly without “unpacking” them:
    \[ \text{sketch}(av + bw) = a \cdot \text{sketch}(v) + b \cdot \text{sketch}(w) \]
  – 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>$\frac{ek}{\epsilon}$</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>
### 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>$\frac{ek}{c}$</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>3.54</td>
<td>516.63 [1.0]</td>
<td>0.28</td>
</tr>
<tr>
<td>MAD-SKETCH ($w = 109$, $d = 8$)</td>
<td>2.68</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>

Freebase
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>

Per Iteration Memory usage over Flickr Graph (1m Labels)

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\textsuperscript{1}  
Carnegie Mellon University, Pittsburgh, PA, USA  
Singapore Mgt. University, Singapore  
qiming.ustc@gmail.com

AIStats 2016
Differences: objective function

\[ C(\hat{Y}) = \mu_1 \sum_{v \in V_I} s_{vu} \| \hat{Y}_v - Y_v \|^2_2 \]

\[ + \mu_2 \sum_{v \in V, u \in N(v)} w_{vu} \| \hat{Y}_v - \hat{Y}_u \|^2 \]

\[ + \mu_3 \sum_{v \in V} \| \hat{Y}_v - U \|^2_2 \]

\[ s.t. \sum_{l=1}^{L} \hat{Y}_{vl} = 1, \forall v \]
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 and end of section if \(\text{Prob}(y) + \Delta\) is small
Results with EXPANDER

Graph 1: Memory (Gigabyte) vs. $\log_{10}$ (number of labels)

Graph 2: Computation time(s) vs. $\log_{10}$ (number of nodes)