Prediction Problems on Networked Data

Claudio Gentile
Universita’ dell’Insubria, Italy

ICML Workshop on Mining and Learning with Graphs
(MLG 2012)
July 1st, 2012

Joint with:
N. Cesa-Bianchi, F. Vitale, G. Zappella
Networked data/1: Intro

Networked data as an undirected, connected and (possibly) weighted graph

Either **nodes** or **edges** (or both) carry **information**

- Web network
  - nodes: websites; edges: content similarity, citations, etc.

- Biological networks
  - nodes: proteins; edges: protein interactions, etc.

- Social Networks
  - nodes: individuals; edges: relationships between individuals

- Recommender systems
  - nodes: items; edges: similarity between items (or viceversa)
Networked data/2: Tasks

Learning: observe some information & infer about the rest

Restrict to **binary classification** tasks in **transductive** settings

- predict + or −
- graph is known in advance

Node classification:

- observe subset of training **nodes**
- build prediction model for remaining ones

Link classification:

- observe subset of training **edges**
- build prediction model for remaining ones

In both cases:

- Graph structure (and edge weights) encodes prior knowledge
- Need to decide on **inductive bias** (underlying regularity) to bet on
- want both **accuracy** and **scalability**
Node classification: Task & inductive bias

Edge weights: encode (possibly data-dependent) strength of node similarity

Irregularity: thick edges connecting mismatching nodes (cut edges)

Inductive bias: Total weight over cut edges tends to be small
Node classification: Task & inductive bias

Edge weights: encode (possibly data-dependent) strength of node similarity

Irregularity: thick edges connecting mismatching nodes (cut edges)

Inductive bias: Total weight over cut edges tends to be small
**Link classification: Task & inductive bias/1**

Links can represent both **positive** and **negative** relationships.

- Social networks: approval/disapproval, positive/negative endorsement
- Biological networks: excitatory/inhibitory interactions

More concrete examples:

- Slashdot: friend/foe
- Epinions: positive/negative ratings of products AND users
- Wikipedia: votes of admin in favor/against another admin
Link classification: Task & inductive bias/2

Task: Learn edge classifier in network where links with positive (+) and negative (−) signs exist, based on training subset

Note: Different from link prediction!
**Link classification: Task & inductive bias/2**

**Task:** Learn edge classifier in network where links with positive (+) and negative (−) signs exist, based on training subset.

Note: Different from link prediction!
Link classification: Task & inductive bias/3

- **Inductive bias**: Minimum cost over node partitions tends to be small
- Sometimes called clustering with side info \([D+99,BDo+99,...]\)
Link classification: Task & inductive bias

2-partitions:
"The enemy of my enemy is my friend"

3-partitions:
"The enemy of my enemy is my enemy"

Cost of 2-partition = 2

Multiplicative rule: Even parity of every cycle

Social balance theory: [H46,H53,CH56, …]

In social networks minimum cost over 2-partitions likely to be small
Learning protocols

- **Standard statistical batch passive:**
  train $\implies$ build static prediction model for remaining items $\implies$ test
  Count total no. of mistakes on test

- **Online passive:**
  (sequential prediction of item signs according to adversarial ordering)
  Keeps dynamically adjusted prediction model for remaining items
  Count total no. of mistakes across time

- **Batch active:**
  alg. chooses training set $\implies$ build static prediction model for remaining items $\implies$ test
  Count total no. of mistakes on test

Can turn online performance to batch performance via reductions
Agenda of presentation

- **Node classification**: theory & algorithms & experiments (online, batch, passive, active, ...)
  
- **Link classification**: theory & algorithms & experiments (online, batch, passive, active, ...)
  
- Possible research directions
Node classification: Algorithms/1

- Weighted graph $G = (V, E, W)$
  $$W = [w_{i,j}]_{i,j=1}^{[V]}$$
- Laplacian matrix $L$
  $$\frac{1}{4} Y^\top L Y = \frac{1}{4} \sum_{i,j \in V} w_{i,j} (Y_i - Y_j)^2$$
  \[ \text{cutsize } \Phi_W(Y) \text{ induced by } Y \]

Energy minimization:

- training set $Y_m = (y_1, y_2, \ldots, y_m, 0, \ldots, 0) \in \{-1, +1\}^{[V]}$
- Prediction model:
  $$\left( y_1, y_2, \ldots, y_m, y_{m+1}, \ldots, y_{|V|} \right) = \arg\min_{Y \in \mathbb{R}^{[V]}} \left\{ \text{training error} + \text{regulariz.} \right\}$$
  $$= \arg\min_{Y \in \mathbb{R}^{[V]}} \left\{ \|Y - Y_m\|^2 + c \left( Y^\top L Y \right) \right\}$$
  then threshold to $\pm 1$
Node classification: Algorithms/1

- Weighted graph $G = (V, E, W)$
  $$W = [w_{i,j}]_{i,j=1}^{V}$$

- Laplacian matrix $L$
  $$\frac{1}{4} Y^\top L Y = \frac{1}{4} \sum_{i,j \in V} w_{i,j} (Y_i - Y_j)^2$$
  cutsize $\Phi_W(Y)$ induced by $Y$

Energy minimization:

- training set $Y_m = (y_1, y_2, \ldots, y_m, 0, \ldots, 0) \in \{-1, +1\}^{|V|}$

- Prediction model:
  $$(y_1, y_2, \ldots, y_m, y_{m+1}, \ldots, y_{|V|}) = \arg\min_{Y \in \mathbb{R}^{|V|}} \underbrace{\text{training error}}_{||Y - Y_m||^2} + \underbrace{\text{regulariz.}}_{c \ Y^\top L Y}$$
  then threshold to $\pm 1$
Node classification: Algorithms/2

Remarks on energy minimization:

- Many variants (e.g., spectrally sparse) [BDR06,...]

- Crisp and accurate in practice but slow: $|V||E|
  \text{matrix implementation also memory demanding: } |V|^2$

- Batch and online (passive learning) variants

Other approaches:
Standard vector space algs. apply once nodes embedded into $|V|$-dim space with pseudometric induced by $L^+$

- Any kernel alg could be applied: $K(i,j) = x_i^\top L^+ x_j$

- Not very accurate in practice and slow: $|V|^3$ in general, $|V|^2$ on trees also memory demanding: $|V|^2$

- Online & batch performance guarantees exist [HP07,HPG09, ...]
• Sparsify: Draw random spanning tree $T$ of $G$ before seeing labels

• Operate optimally on $T$:
  – Partition $T$ into component subtrees bordered by either forks or revealed nodes
  – Estimate label of forks by mincut (minimizing weighted cutsize in $T$ consistent with labels so far)
  – Apply resistance distance-based NN on node(s) to predict
Node classification: Algorithms/3: Shazoo/1 

- Sparsify: Draw random spanning tree $T$ of $G$ before seeing labels
- Operate optimally on $T$:
  - Partition $T$ into component subtrees bordered by either forks or revealed nodes
  - Estimate label of forks by mincut (minimizing weighted cutsize in $T$ consistent with labels so far)
  - Apply resistance distance-based NN on node(s) to predict
Node classification: Algorithms/3: Shazoo/1

- Sparsify: Draw random spanning tree $T$ of $G$ before seeing labels

- Operate optimally on $T$:
  - Partition $T$ into component subtrees bordered by either forks or revealed nodes
  - Estimate label of forks by mincut (minimizing weighted cutsize in $T$ consistent with labels so far)
  - Apply resistance distance-based NN on node(s) to predict

- Sparsify: Draw random spanning tree $T$ of $G$ before seeing labels
- Operate optimally on $T$:
  - Partition $T$ into component subtrees bordered by either forks or revealed nodes
  - Estimate label of forks by mincut (minimizing weighted cutsize in $T$ consistent with labels so far)
  - Apply resistance distance-based NN on node(s) to predict
Node classification: Algorithms/3: Shazoo/1

- **Sparsify:** Draw random spanning tree \( T \) of \( G \) before seeing labels

- **Operate optimally on \( T \):**
  - Partition \( T \) into component subtrees bordered by either forks or revealed nodes
  - Estimate label of forks by mincut (minimizing weighted cutsize in \( T \) consistent with labels so far)
  - Apply resistance distance-based NN on node(s) to predict
Node classification: Algorithms/3: Shazoo/1

- Sparsify: Draw random spanning tree $T$ of $G$ before seeing labels
- Operate optimally on $T$:
  - Partition $T$ into component subtrees bordered by either forks or revealed nodes
  - Estimate label of forks by mincut (minimizing weighted cutsize in $T$ consistent with labels so far)
  - Apply resistance distance-based NN on node(s) to predict

Shazoo’s properties:

- **Thm:** In the online setting Shazoo’s no. of mistakes $M$ satisfies
  \[
  E[M] \leq \frac{E_T[\Phi_T(Y)]}{\text{lower bound}} \log(\ldots)
  \]
  mistakes, $\Phi_T(Y) =$ cutsize induced by labeling $Y$ on spanning tree $T$

- Sparsification with random spanning tree (RST) faces an adaptive adversary and retains relevant properties of original graph

- Running time on passive batch:
  \[
  \underbrace{\text{RST}}_{O(|V|)} + O(|V|) \text{ on “most” graphs}
  \]
  for predicting whole test set (constant amortized time per prediction)

- Running time on passive online: $O(|V|)$ per prediction (worst case)

- Memory space: Linear in $|E|$
Node classification: Some experiments

Batch passive experiments with varying train/test mix

Datasets:

- Web spam detection (~100K nodes)
- OCR, text categorization (~10K nodes)
- Bioinfo (~2K nodes)

<table>
<thead>
<tr>
<th>Webspam (F-measure)</th>
<th>Shazoo</th>
<th>Wta</th>
<th>Omv</th>
<th>Labprop</th>
<th>3*Shazoo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.954</td>
<td>0.947</td>
<td>0.706</td>
<td>0.931</td>
<td>0.964</td>
</tr>
</tbody>
</table>

Others (accuracy)

- Shazoo + MST is comparable/better than Labprop, but faster
- Committee of RST + Shazoo outperforms Labprop on small train size
Link classification: Correlation clustering (CC) index

- Undirected and unweighted $G = (V, E)$
- Binary labeling $Y_{i,j} \in \{-1, +1\}$ $(i,j) \in E$

CC index:

$\Delta(Y) = \text{Minimum cost over all partitions of } V$

$= \text{smallest no. of edges to drop to eliminate all bad cycles (obtaining cost = 0)}$

$\Delta_2(Y) = \text{minimum cost over all 2-partitions}$

$\Delta(Y) \leq \Delta_2(Y)$

- (CC) index $\Delta$ (or $\Delta_2$) characterizes accuracy of learning link classifier in adversarial settings (online passive, batch passive, active)
- NP-hard to compute (or to approximate within constants) on general graphs . . . [B+04,D+06,GG06]
- . . . but nice approximation results around [D+06]
- Hardly used for learning in real-world scenarios $\implies$ relaxations!
Link classification: Online passive

CC-index characterizes online passive classification:

Sequential prediction of edge signs according to adversarial ordering

Thm:

- There is an alg. that for any $K \geq 0$ and any $(G = (V,E), Y)$ makes
  
  $$O \left( |V| + \frac{K}{0 \ldots O(|V|^2)} \right) \log \frac{|E|}{K}$$

  prediction mistakes, while $\Delta(Y) \leq K$

- Not improvable on some graphs (cliques), possibly improvable by only log factor on others

- Cannot be implemented in poly-time unless $\text{RP} = \text{NP}$
Link classification: (Statistical) batch passive \cite{CGVZ12}

Combines off-the-shelf techniques:

- Random draw of training edges
- Computes partition approximating $\Delta$ (or $\Delta_2$) on training graph
- Use that partition to predict test edges \cite{EYP09}

E.g.:

- Train and test of comparable size
- Train by approximating $\Delta$ on training graph within log $|V|$ factor \cite{D+06}
- Yields $O \left( \Delta(Y) \log |V| + \sqrt{|E||V| \log |V|} \right)$ mistakes on test set (w.h. prob.)

**Drawbacks:** i) Algs not easy to implement, ii) do not scale to large graphs, and iii) bound vacuous when $|E| = O(|V|)$
Link classif. : Spectral (but heuristic) approaches on $\Delta_2$

Signed graph Laplacian:

$$L_s = D - Y, \quad D = \text{diag}(d_1, \ldots, d_{|V|}), \quad Y = [Y_{i,j}]$$

$$\frac{4}{n} \Delta_2(Y) = \min_{x \in \{-1, +1\}^n} \frac{x^\top L_s x}{\|x\|^2} \approx \min_{x \in \mathbb{R}^n} \frac{x^\top L_s x}{\|x\|^2} \implies x \text{ is minimal eigenvector}$$

Minimal eigenvector heuristic:

- $\hat{L}_s$ = signed Laplacian of training subgraph
- Compute $x = \text{minimal eigenvector}$ (assuming $\det(\hat{L}_s) > 0$)
- Predict $Y_{i,j}$ for remaining edges $(i, j)$ as $\text{sign}(x_i x_j)$

Remarks:

- Similar to empirical risk minimizer, but lacks theoretical motivation
- Many variants (e.g., $k$ smallest, matrix exponentials, ...)
- Scaling issues ($|V|^2$ to $|V|^3$)
**Link classification: Active (on $\Delta_2$)/1: Lower bound**

Batch active learning setting:

- **Selection**: select (non-adaptively) a subset of edges, observe their labels
- **Prediction**: build prediction model for the rest

**Thm:**
For any $K \geq 0$, $G = (V, E)$, and any active learning alg selecting $\alpha$-fraction of edges there is labeling $Y$ s.t. $\Delta_2(Y) \leq K$ and

$$M = \text{No. of mistakes in prediction phase} \geq (1 - \alpha) \frac{K}{2}$$
Link classification: Active (on $\Delta_2$)/2: Relaxations

P-random model:
Labeling $Y$ in $(G = (V, E), Y)$ generated by perturbing initial $Y^0 : \Delta_2(Y^0) = 0$

\[ Y_{i,j} = \begin{cases} Y^0_{i,j} & \text{with prob. } p \\ -Y^0_{i,j} & \text{with prob. } 1 - p \end{cases} \]

- Notice that $E[\Delta_2(Y)] \leq p|E| \implies$ replace $\Delta_2(Y)$ by $p|E|$

- Previous lower bound still holds:
  \[ E[M] \geq (1 - \alpha)p|E| \]

- Greatly simplifies design of algorithms

- Main concern becomes predicting via short paths
**Link classification: Active (on $\Delta_2$)/3: Algorithms/1**

**Rule of thumb:**
Sparsify by breadth-first (or shortest-path) spanning tree $T$

![Graphs](image)

- **Selection:** query labels of the $|V| - 1$ edges of $T$
- **Prediction:** predict edge labels by parity of (unique) path in $T$

If $G = (V, E)$ has small diameter $d_G$

$$E[M] \leq 2d_G p|E|$$

**Drawbacks:** i) only $|V| - 1$ edges, ii) What if $d_G$ not small
Link classification: Active (on $\Delta_2$)/3: Algorithms/2

Low-stretch spanning trees (metric sparsifiers)

![Graphs G and T](image)

Stretch of $T = \frac{\text{average distance } i-j \text{ over } T}{\text{average distance } i-j \text{ over } G}$

$|E| \log |V|$-time alg $\exists$ that builds $T$ with stretch $\log^2 |V| \log \log |V|$ [EEST10]

- **Selection:** query labels of the $|V| - 1$ edges of $T$
- **Prediction:** predict edge labels by parity of (unique) path in $T$

$$\implies E[M] \leq (\log^2 |V| \log \log |V|) p |E|$$

**Drawbacks:** i) only $|V| - 1$ edges, ii) not easy to implement
Link classification: Active (on $\Delta_2$)/3: Algorithms/3

TreeletStar [CGVZ12]
• Selection:
  - Select node with highest degree
  - Query edges of its star subgraph
  - Erase edges and continue
  - $\forall$ pairs of stars, query one connecting edge

• Prediction:
  - Predict within star edges by parity of path within star
  - Predict between star edges by parity of path connecting the two stars

More refined (recursive) version with tunable parameter $k$:

Querying $\left(\frac{|V|}{k}\right)^{3/2}$ edges $\implies E[M] \leq kp|E|$.

• Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
• Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality
**Link classification: Active (on $\Delta_2$)/3: Algorithms/3**

**TreeletStar** [CGVZ12]

- **Selection:**
  - Select node with highest degree
  - Query edges of its star subgraph
  - Erase edges and continue
  - $\forall$ pairs of stars, query one connecting edge

- **Prediction:**
  - Predict within star edges by parity of path within star
  - Predict between star edges by parity of path connecting the two stars

More refined (recursive) version with tunable parameter $k$:

Querying $(|V|/k)^{3/2}$ edges $\implies E[M] \leq kp|E|$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality
**Link classification: Active (on $\Delta_2)/3$: Algorithms/3**

TreeletStar \[ {\text{[CGVZ12]}} \]

- **Selection:**
  - Select node with highest degree
  - Query edges of its star subgraph
  - Erase edges and continue
  - $\forall$ pairs of stars, query one connecting edge

- **Prediction:**
  - Predict within star edges by parity of path within star
  - Predict between star edges by parity of path connecting the two stars

More refined (recursive) version with tunable parameter $k$:

Querying $(|V|/k)^{3/2}$ edges $\implies E[M] \leq k p |E|$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality
Link classification: Active (on $\Delta_2)/3$: Algorithms/3

TreeletStar [CGVZ12]

- **Selection:**
  - Select node with highest degree
  - Query edges of its star subgraph
  - Erase edges and continue
  - $\forall$ pairs of stars, query one connecting edge

- **Prediction:**
  - Predict within star edges by parity of path within star
  - Predict between star edges by parity of path connecting the two stars

More refined (recursive) version with tunable parameter $k$:

Querying $\left(\frac{|V|}{k}\right)^{3/2}$ edges $\implies E[M] \leq kp|E|$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality
**Link classification: Active (on $\Delta_2$)/3: Algorithms/3**

**TreeletStar** [CGVZ12]

- **Selection:**
  - Select node with highest degree
  - Query edges of its star subgraph
  - Erase edges and continue
  - $\forall$ pairs of stars, query one connecting edge

- **Prediction:**
  - Predict within star edges by parity of path within star
  - Predict between star edges by parity of path connecting the two stars

More refined (recursive) version with tunable parameter $k$:

Querying $O(|V|/k)^{3/2}$ edges $\Rightarrow E[M] \leq kp|E|$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality
Link classification: Active (on $\Delta_2$)/3: Algorithms/4

Treeletstar’s computational aspects:

- Running time for predicting whole test set: $|E| + (|V|/k) \log(|V|/k)$
- Memory space: Linear in $|E|$
Link classification: Some experiments/1

Batch passive vs. active with varying train/test mix

Datasets:

- Delta\(X\), out of USPS dataset (OCR), \(X = 0, 100, 250\)
- Movielens (edges between users by cosine similarity)
- Snapshot of Slashdot (largest strongly connected component)
- Snapshot of Epinions (as above)

| Dataset     | \(|V|\) | \(|E|\)  | Neg.    | Avgdeg |
|-------------|--------|---------|---------|--------|
| DELTAX      | 1000   | 9138    | \(\approx 22.0\%\) | 18.2   |
| SLASHDOT    | 26996  | 290509  | 24.7\%  | 21.6   |
| EPINIONS    | 41441  | 565900  | 26.2\%  | 27.4   |
| MOVIELENS   | 6040   | 824818  | 12.6\%  | 273.2  |

Spectral (but passive) baseline: ASymExp \[\text{KLB09}\]

Train sign matrix \(Y_{\text{train}}\):

\[
Y_{\text{train}} \approx U_z \exp(D_z) U_z^\top, \quad z = 1, 5, 10, 15
\]

Predict \(Y_{i,j}\) by \(\text{sign}(U_z \exp(D_z) U_z^\top)\)
Link classification: Some experiments/2

DELTA0

DELTA100

DELTA250

MOVIELENS

SLASHDOT

EPINIONS
Conclusions and research directions

- Models
  \[ \Rightarrow \text{characterization} \]
  \[ \Rightarrow \text{principled algs. can be designed if problem is crisp enough} \]
  (better avoid matrices anyway ...)

- Research directions:
  - Typical real-world topologies (e.g. preferential attachment, ...)
  - Connection to link prediction (e.g. ”Kats rule” in unsigned graphs) [C+11]
  - Combine sources of info (overlapped networks, node & links, ...)
  - Directed graphs (both node and link classification)