A Fast Active Learning Algorithm for Link Classification

Claudio Gentile

Universita' dell'Insubria, Italy

ICTCS 2012

September 19th, 2012

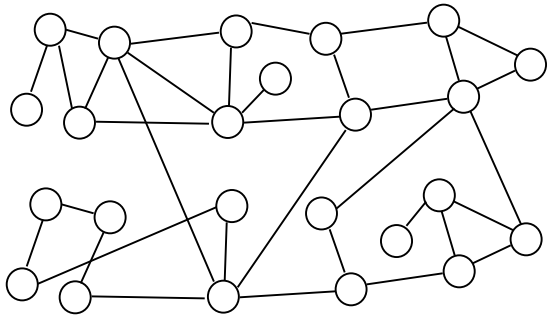
Joint with:

N. Cesa-Bianchi, F. Vitale, G. Zappella

Networked data/1: Intro/1

Networked data: Undirected, connected and (for simplicity) unweighted graph

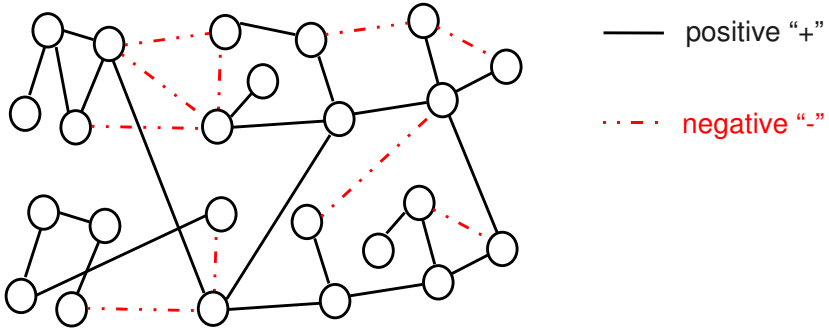
Either nodes or edges
(or both)
carry **information**



- 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 through users
  (or viceversa)

Networked data/1: Intro/2

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

Networked data/2: Learning to classify links/1

Learning: Inductive inference = inference with errors

Observe some information & infer about the rest

Relies on regularity of observed phenomenon

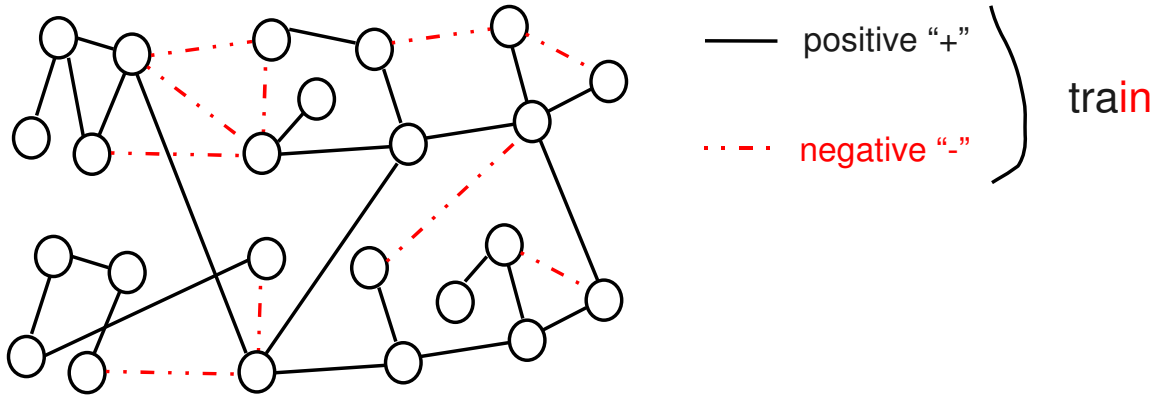
Link classification:

- observe subset of training edges
- build prediction model (= inference) for remaining ones

Restrict to $\underbrace{\text{binary classification}}_{\text{predict } + \text{ or } -}$ tasks in $\underbrace{\text{transductive}}_{\text{graph is known in advance}}$ settings

Networked data/2: Learning to classify links/2

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

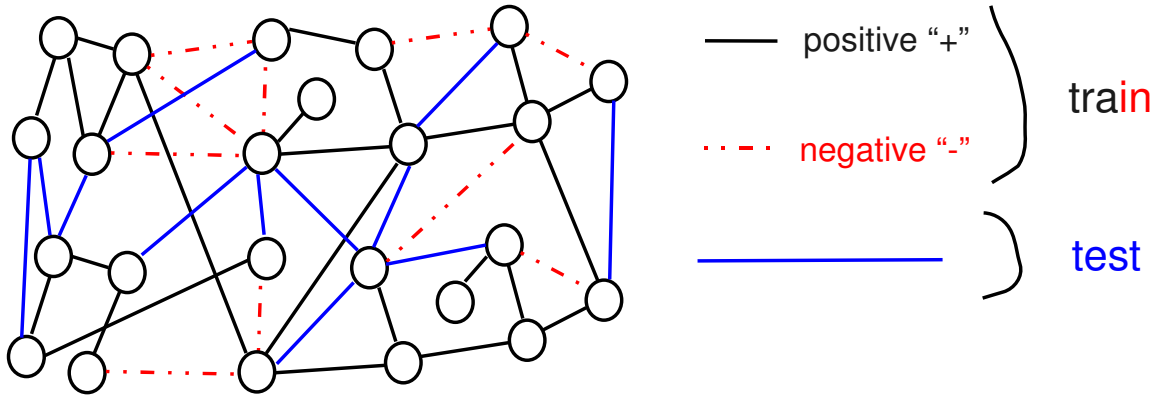


- Graph structure encodes prior knowledge
- Need to decide on inductive bias (underlying regularity) to bet on
- want both **accuracy** and **scalability**

Note: Different from link prediction !

Networked data/2: Learning to classify links/2

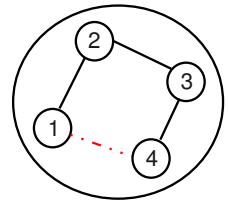
Task: Learn edge classifier in network where links with positive (+) and negative (−) signs exist, based on training subset



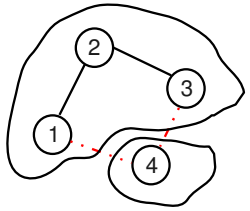
- Graph structure encodes prior knowledge
- Need to decide on inductive bias (underlying regularity) to bet on
- want both **accuracy** and **scalability**

Note: Different from link prediction !

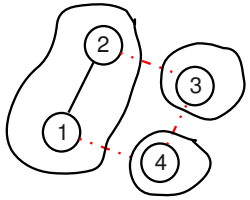
Link classification: Task & inductive bias/1



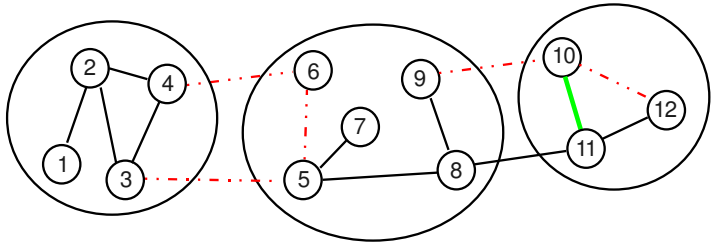
Bad cycle
Cost of partition = 1



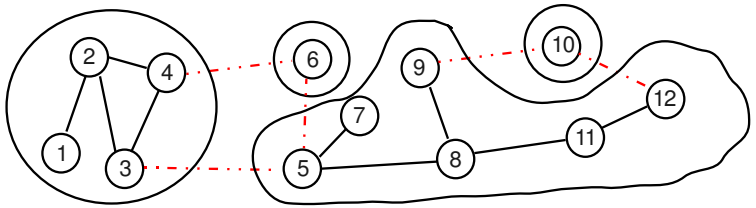
Cost of partition = 0



Cost of partition = 0
Cost of 2-partition = 1



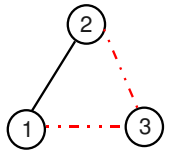
Cost of partition = 3
Two bad cycles
Cost of optimal partition = 1



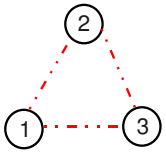
No bad cycles

- Inductive bias: Minimum cost over node partitions tends to be small
- Sometimes called clustering with side info [D+99,BDo+99,...]

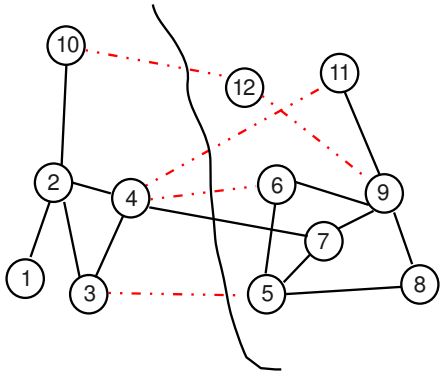
Link classification: Task & inductive bias/2



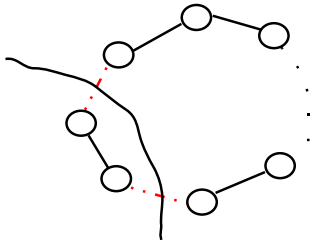
2-partitions:
"The enemy of my enemy
is my friend"



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

Many learning protocols

- Standard statistical batch passive:

  train $\Longrightarrow$ build static prediction model for remaining items $\Longrightarrow$ 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 (**covered here**):

  alg. chooses training set $\Longrightarrow$ build static prediction model for remaining items $\Longrightarrow$ test

  Count total no. of mistakes on test

Can turn online performance to batch performance via reductions

Link classification: Correlation clustering (CC) index

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

CC index:

$\Delta(Y) =$ Minimum cost over all partitions of V
$=$ smallest no. of edges to drop to eliminate all bad cycles
(obtaining cost $= 0$)

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

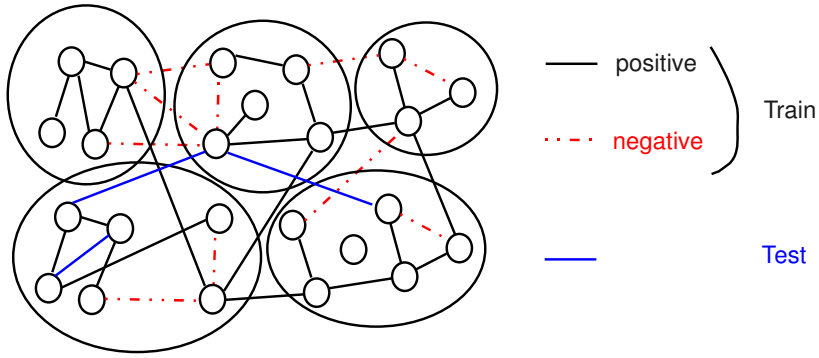
$\Delta(Y) \leq \Delta_2(Y)$

- (CC) index Δ (or Δ_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 $\Longrightarrow$ relaxations !

Link classif. : E.g. (Statistical) batch passive [CGVZ12]

Combines off-the-shelf techniques:

- Random draw of training edges
- Computes partition approximating Δ (or Δ_2) on training graph
- use that partition to predict test edges [EYP09]



E.g. :

- train and test of comparable size
- train by approximating Δ on training graph within $\log |V|$ factor [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 Δ_2

Signed graph Laplacian:

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

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

Minimal eigenvector heuristic:

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

Remarks:

- Works fine but lacks theoretical motivation
- Many variants (e.g., k smallest, matrix exponentials, ...)
- Scaling issues ($|V|^2$ to $|V|^3$)

Link classification: Active (on Δ_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 α-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 Δ_2)/2: Relaxations

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

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

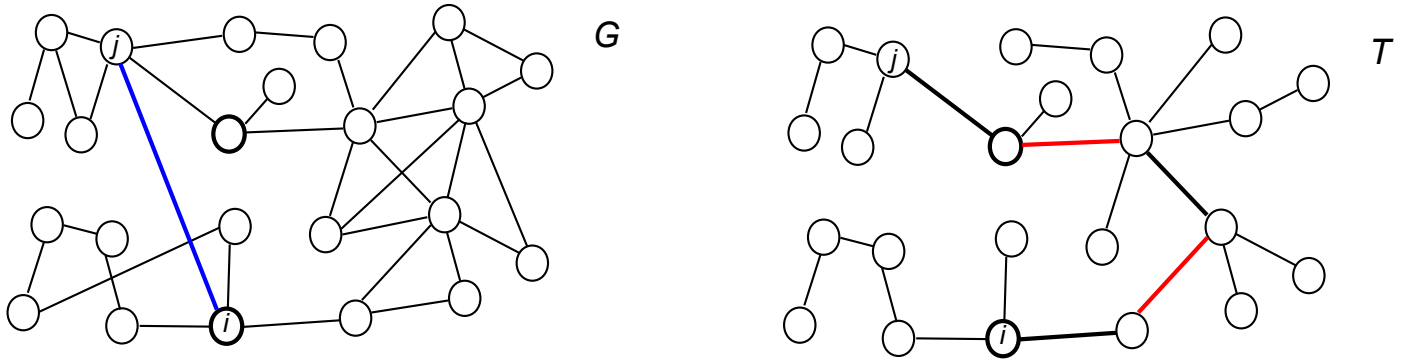
- Notice that $E[\Delta_2(Y)] \leq p\,|E| \implies$ replace $\Delta_2(Y)$ by $p\,|E|$
- Previous lower bound still holds:

$$E[M] \geq (1-\alpha)\,p\,|E|$$

- Greatly simplifies design of algorithms
- Main concern becomes predicting via short paths

Link classification: Active (on $\triangle_2$)/3: Algorithms/1

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



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

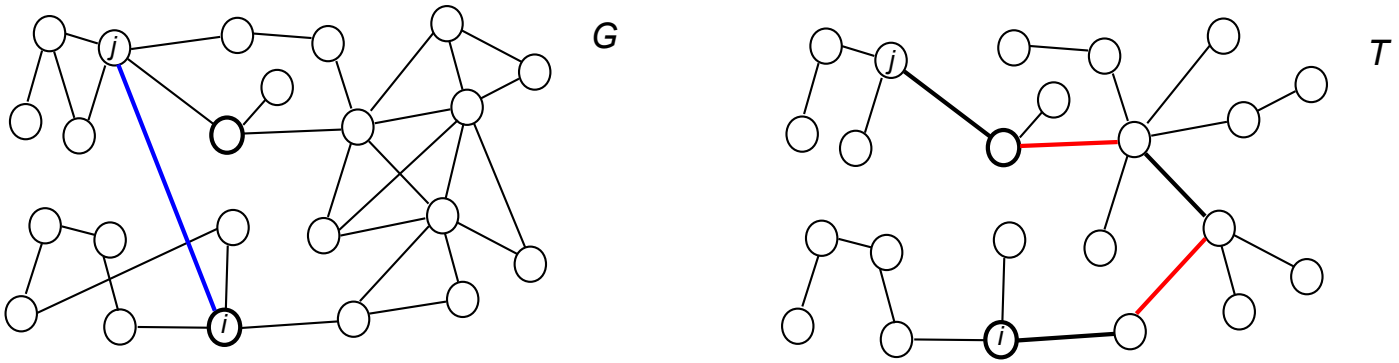
If $G=(V,E)$ has small diameter d_G

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

Drawbacks: i) only $|V|-1$ edges, ii) What if d_G not small

Link classification: Active (on $\triangle_2$)/3: Algorithms/2

Low-stretch spanning trees (metric sparsifiers)



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

$|E|\log|V|$-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

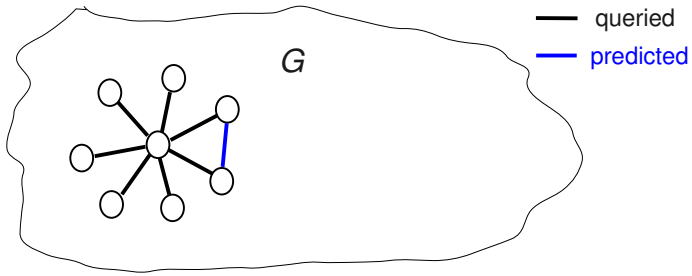
$$\Longrightarrow E[M] \leq (\log^2|V|\log\log|V|)\, p\,|E|$$

Drawbacks: i) only $|V|-1$ edges, ii) not easy to implement

Link classification: Active (on $\triangle_2$)/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:

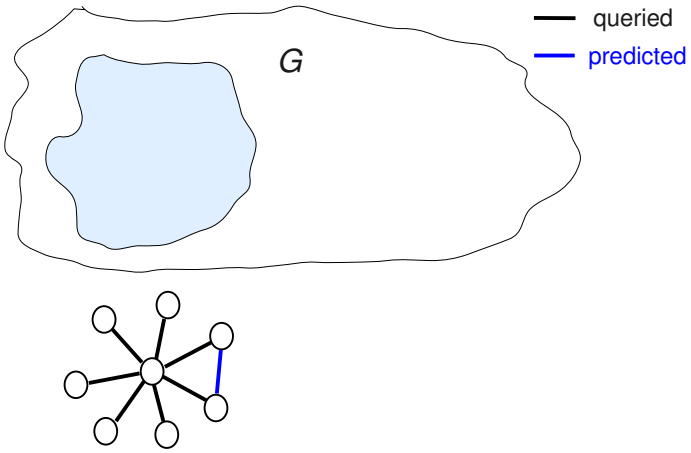
$$\text{Querying } (|V|/k)^{3/2} \text{ edges} \implies E[M] \leq k\,p\,|E|$$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality

Link classification: Active (on $\triangle_2$)/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:

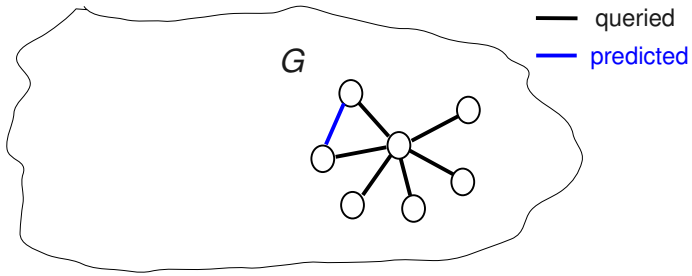
$$\text{Querying } (|V|/k)^{3/2} \text{ edges} \implies E[M] \leq k\,p\,|E|$$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality

Link classification: Active (on $\triangle_2$)/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:

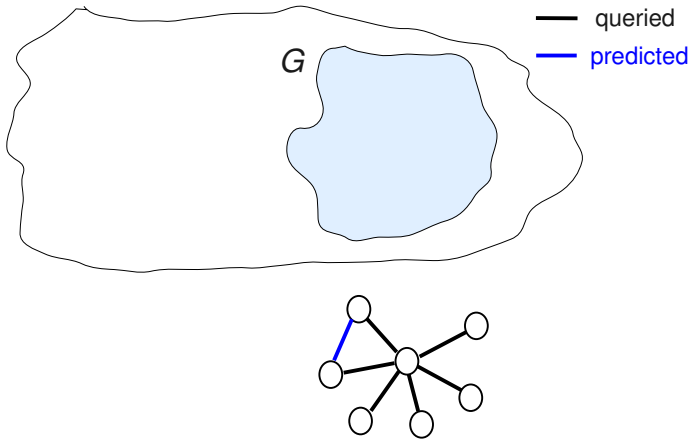
$$\text{Querying } (|V|/k)^{3/2} \text{ edges} \implies E[M] \leq k\,p\,|E|$$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality

Link classification: Active (on $\triangle_2$)/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:

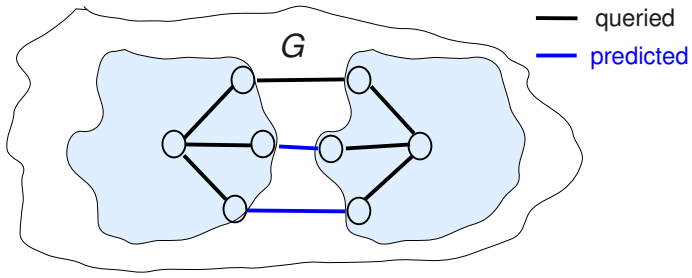
$$\text{Querying } (|V|/k)^{3/2} \text{ edges} \implies E[M] \leq k\,p\,|E|$$

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality

Link classification: Active (on $\triangle_2$)/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:

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

- Querying $O(|V|)$ gets $k = |V|^{1/3}$ optimality
- Querying $O(|V|^{3/2})$ gets $k = O(1)$ optimality

Link classification: Active (on $\triangle_2$)/3: Algorithms/4

Treeletstar's computational aspects :

- (Amortized) running time for predicting whole test set:

$$|E| + (|V|/k) \log(|V|/k)$$

- Memory space : Linear in $|E|$

Conclusions and research directions

- Flavor of this research activity
- 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, weighted graphs
  - Active with more than two clusters