Learning to make predictions in graphs

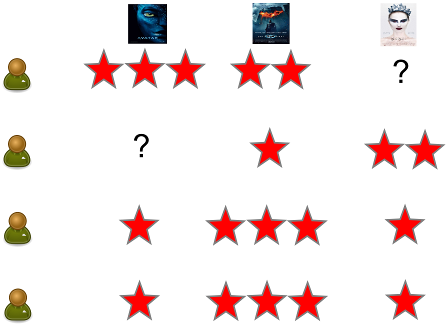
Aditya Krishna Menon
(Joint work with Charles Elkan)

ID Analytics, May 24, 2012

Outline

Movie rating prediction

- **Netflix prize**: Given users' ratings of movies they have seen, predict ratings on the movies they have not seen



- Popular solution is collaborative filtering

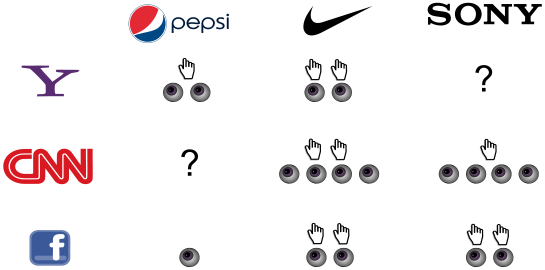
Friends in social networks

- **Recommending friends**: Given whether certain pairs of users know each other, predict whether a new pair are likely to know each other

- Popular solutions are scores computed from graph topology

Computational advertising

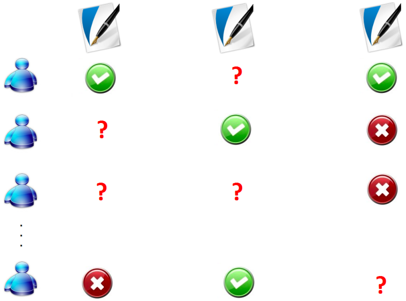
- **Response prediction**: Given ads' clickthrough rates on webpages, predict clickthrough rate for an ad on a webpage it has not been shown on



- Popular solution is supervised learning with feature engineering

Item response theory

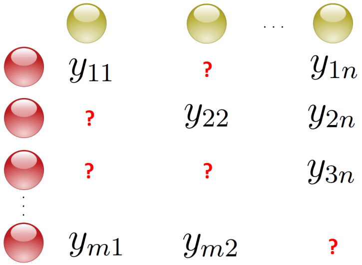
- **Exam performance**: Given student's performance on questions in an exam, predict performance on unanswered questions



- Popular solution is ideal point model

The general problem: dyadic prediction

- **General dyadic prediction**: Given labels between certain pairs of objects (or dyads), predict the labels for the unobserved dyads

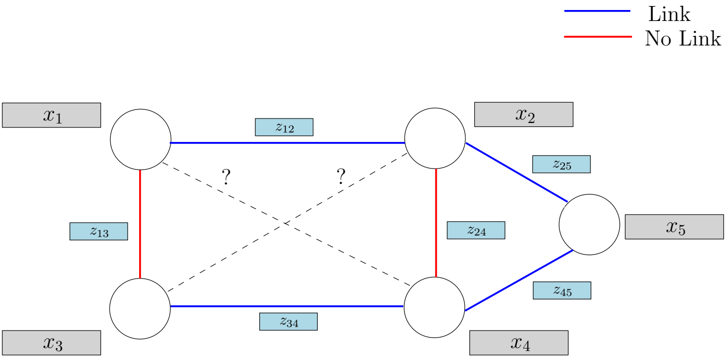


  - Possibly have feature vectors for the rows, columns, and cells
  - A type of generalized matrix completion

- Solution? Depends on problem instanation...

A graph view: link prediction

- Given a graph G with partially labelled edges
  - Possibly have feature vectors for the nodes and edges



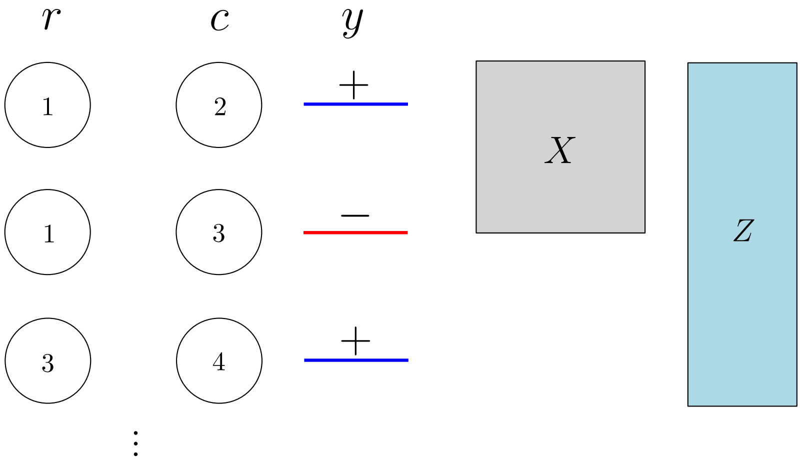
- **Task**: Predict labels for all edges
  - No-Link $\neq$ unknown label

Dyadic prediction formally - I

- **Training set**: $\{((r_i, c_i), y_i)\}_{i=1}^{N}$
  - $(r_i, c_i) \to$ dyad, $y_i \to$ label
  - $r_i \in \{1, \ldots, m\}$, $c_i \in \{1, \ldots, n\}$ represent member identities
  - $y_i \in \mathcal{Y}$, e.g. $\{1, \ldots, 5\}$
- **Auxiliary information**: $X_1 \in \mathbb{R}^{m \times d_1}$, $X_2 \in \mathbb{R}^{n \times d_2}$, $Z \in \mathbb{R}^{mn \times d_3}$
  - Feature vectors describing dyad members and the dyads
  - Optional to specify; can learn without them!
- **Output**: Mapping $f : \{1, \ldots, m\} \times \{1, \ldots, n\} \to \mathcal{Y}$
  - Given a new dyad, would like to predict the label

Dyadic prediction formally - II

- For predicting edges in an undirected graph:



Dyadic prediction: flexibility

- Dyad members (r, c)?
  - Same or different space
    - Users and movies, or users and users in a social network
  - Unique identifiers only, or feature vectors
    - Netflix prize versus computational advertising
- Labels y?
  - Unordered (nominal) or ordered
    - { Viewed, Purchased, Returned } or $\{ 1, 2, \ldots, 5 \}$
  - Single or multi-label
    - { Friend, Colleague, Family Member }
- Structure of data?
  - Undirected or directed graph
    - Friendships versus trust relations in a social network

This talk

- A general dyadic prediction model that learns latent features
  - Associate a "fingerprint" with each dyad/node in the graph
  - Linking behaviour $\rightarrow$ interaction of these fingerprints
- Main focus:
  - Modelling disparate problems in a single framework
  - Adapting to problem-specific constraints
  - Focus on objectives other than accuracy

Outline

Can logistic regression do the job?

- Recall the logistic regression model:

$$\Pr[y = 1|x; w] = \sigma(w^T \phi(x)),$$

  where $\sigma(z) = 1/(1 + e^{-z})$, and $\phi(\cdot)$ is some transformation
- Can we just let $x = (r, c)$ and be done?
  - Theoretically no, because the iid assumption fails
  - But what exactly goes wrong?

A limitation of logistic regression - I

- Recall that r, c are identities of the dyad members
  - Or in a graph, the source and destination
  - Must represent them in a way logistic regression understands
- A sensible encoding is a one-hot (one-of-K) scheme
  - The resulting features will be

$$x = \begin{bmatrix} e_r & e_c \end{bmatrix}$$

    where e_k is the standard bitvector

$$e_k = \begin{bmatrix} 0 & 0 & \ldots & 1 & 0 & \ldots & 0 \end{bmatrix}$$

A limitation of logistic regression - II

- The logistic regression model will be

$$\Pr[y = 1|x] = \sigma(\alpha_{r(x)} + \beta_{c(x)})$$

  i.e. will comprise source- and destination-specific biases
- Consider the ranking over destinations for any two source nodes r_1, r_2
  - This will be independent of the parameters $\alpha_{r_1}, \alpha_{r_2}$
  - $\implies$ all source nodes will induce the same ranking over destinations!

How do we fix the problem? - I

- Maybe supervised learning is not a good way to think of the problem?
- Many intuitive alternate schemes have been proposed:
  - Count # of common neighbours
  - Multiply degrees of the nodes
  - Count # of paths of certain length between nodes
  - ...
- But these only exploit topological structure
  - How to also look at features for node and edges?
  - Further, scoring based on some fixed criterion

How do we fix the problem? - II

- Ranking problem would disappear if we could take cross features
- Turns out the naïve solution will give us:

$$\Pr[y = 1|x] = \sigma(\gamma_{r(x)c(x)})$$

  i.e. single parameter for every (source, node) dyad!
  - Memorizes training data
  - Cannot generalize to unseen dyad
- What sort of model lets us get around this?

Outline

Our starting point: log-linear models

- We'll describe a log-linear model for dyadic prediction
- Simple, flexible framework
  - Models probabilities of labels given examples
    - Useful for taking actions based on predictions
  - Labels can be nominal or ordered
    - Applicable to a range of tasks
  - Can integrate identity- and feature-information
    - Exploit all available information

The log-linear framework

- A log-linear model for inputs $x \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ assumes

$$p(y|x;\theta) \propto \exp\left(\sum_{j=1}^{J} \theta_j f_j(x,y)\right)$$

  for a weight vector θ, and feature functions $f_j : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- Letting $x = (r, c)$, resulting probability model is

$$p(y|(r,c);\theta) \propto \exp(W_{rc}^{y})$$

  for some tensor $W \in \mathbb{R}^{m \times n \times |\mathcal{Y}|}$

The log-linear framework

- A log-linear model for inputs $x \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ assumes

$$p(y|x;\theta) \propto \exp\left(\sum_{j=1}^{J} \theta_j f_j(x,y)\right)$$

  for a weight vector θ, and feature functions $f_j : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- Letting $x = (r, c)$, resulting probability model is

$$p(y|(r,c);\theta) \propto \exp(W^y_{rc})$$

  for some tensor $W \in \mathbb{R}^{m \times n \times |\mathcal{Y}|}$
  - **Problem**: W^y_{rc} not defined for unobserved (r, c) pairs!
  - Ends up memorizing the training set

Factorizing interaction weights

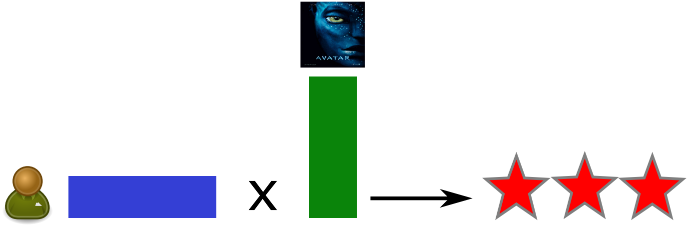
- **Solution**: Factorize the interaction weights W:

$$W^y_{rc} \approx (\alpha^y_{r:})^T \Lambda^y \alpha^y_{c:} = \sum_{k,k'=1}^{K} \lambda^y_{kk'} \alpha^y_{rk} \alpha^y_{ck'}$$

  for some fixed $K \in \mathbb{Z}_+$
  - Ties together parameters, prevents memorization
- Interpretation of parameters for a fixed r, c, y:
  - $\alpha \rightarrow$ latent features ("fingerprint") for each dyad member (node in the graph)
  - $\Lambda \rightarrow$ scaling factors for each latent dimension

Latent feature modelling

- Associate a latent vector with each dyad member
  - Movies with indie-aesthetic, rich orchestration, based on a book, . . .
- Label $= f$(similarity of corresponding latent vectors)



  - Identities of dyad members influence label
  - $\sim$ SVD with missing data

Incorporating explicit features

- If the dyad (r, c) has a feature vector $s_{rc} \in \mathbb{R}^d$, we use:

$$p(y|(r,c);\theta) \propto \exp((\alpha_r^y)^T \Lambda^y \alpha_c^y + (v^y)^T s_{rc})$$

  - Multinomial logistic regression with s_{rc} as feature vector
- Latent and explicit features complement each other
  - If e.g. user has no ratings $\rightarrow$ ignore latent features, just use feature weights

The LFL model: definition

- Resulting model with latent and explicit features:

$$p(y|(r,c);\theta) \propto \exp((\alpha_r^y)^T \Lambda^y \alpha_c^y + (v^y)^T s_{rc})$$

- Think of $\alpha^y \in \mathbb{R}^{(m+n)\times K}$ as latent feature vectors
  - $K = \#$ of latent features
  - Call this the latent feature log-linear or LFL model [Menon and Elkan, 2010a]

Training the LFL model - I

- For training set $\mathcal{T}$, training objective is

$$\min_{\alpha,\Lambda,v} \sum_{((r,c),y)\in\mathcal{T}} -\log p(y|\alpha_r, \alpha_c, \Lambda, v) + \lambda\Omega(\alpha, \Lambda, v)$$

  - Only models observed entries
    - No attempt to impute the missing entries when training
  - ℓ_2 regularization to prevent overfitting
- # of parameters $= (m+n) \cdot K \cdot |\mathcal{Y}|$
  - From a graph POV, linear in the # of nodes

Training the LFL model - II

- Optimize by block coordinate descent
  - Fix all other parameters, optimize for α_1; repeat for α_2, α_3, and so on
  - Each optimization is parallelizable across rows/columns

- Time to train $= |\mathcal{O}| \cdot K \cdot \mathcal{Y} \cdot \#$ of iterations
  - From a graph POV, linear in the # of labelled edges

Outline

Probability model for undirected graphs

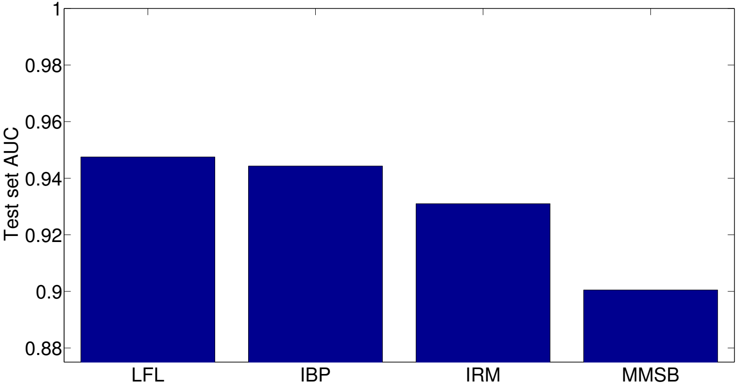
- Recall that the basic LFL model was

$$p(y|(r,c);\theta) \propto \exp((\alpha_r^y)^T \Lambda^y \alpha_c^y + (v^y)^T s_{rc})$$

- Directed graph $\rightarrow \Lambda^y$ some general, asymmetric matrix
- Undirected graph $\rightarrow \Lambda^y$ some diagonal matrix
  - $\sim$ eigendecomposition

Experimental results: unordered link prediction

- Results on Alyawarra dataset, comprising kinship relations ($\{$ Brother, Sister, Father, $\ldots \}$) between 104 people
- Our model outperforms Bayesian models for relational data



Probability model for binary edge weights

- Commonly studied setting involves binary edge weights
- Here, the LFL model reduces to

$$p(y|(r,c);\theta) = \sigma(\alpha_r^T \Lambda \alpha_c + x_r^T W x_c + v^T z_{rc})$$

for sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$, node features $x_r \in \mathbb{R}^d$ and edge features $z_{rc} \in \mathbb{R}^{d'}$

Dealing with class imbalance

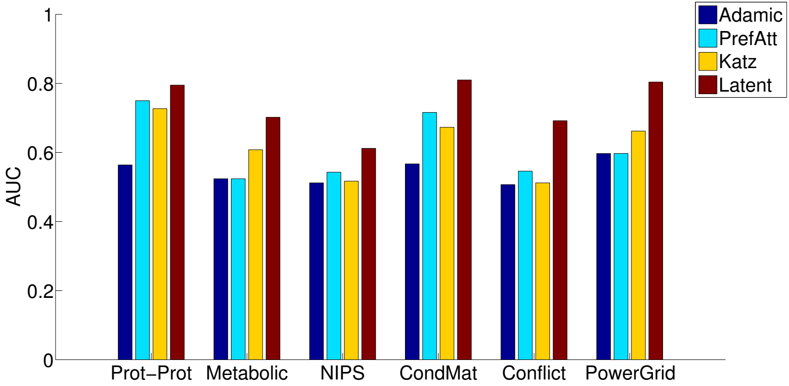
- Challenge: class imbalance
  - Vast majority of node-pairs do not link with each other
- To overcome imbalance, optimize latent features to maximize convex approximation to AUC [Menon and Elkan, 2011]:

$$\min_{\alpha,\Lambda,W,v} \sum_{(i,j,k)\in\mathcal{D}} \ell(\hat{G}_{ij} - \hat{G}_{ik}, 1) + \Omega(\alpha, \Lambda, W, v)$$

where $\mathcal{D} = \{(i,j,k) : G_{ij} = 1, G_{ik} = 0\}$

Experiments on binary link prediction - I

- Datasets from various applications of link prediction:
  - **Computational biology**: Protein/metabolic networks
  - **Citation network**: NIPS authors, condensed matter physicists
  - **Other**: Military disputes, US electric powergrid
- Latent features → directly predictive of link behaviour:



Experiments on binary link prediction - II

- Fewer observed edges $\implies$ unsupervised performance $\approx$ random
- Latent features still manage to be reasonably predictive

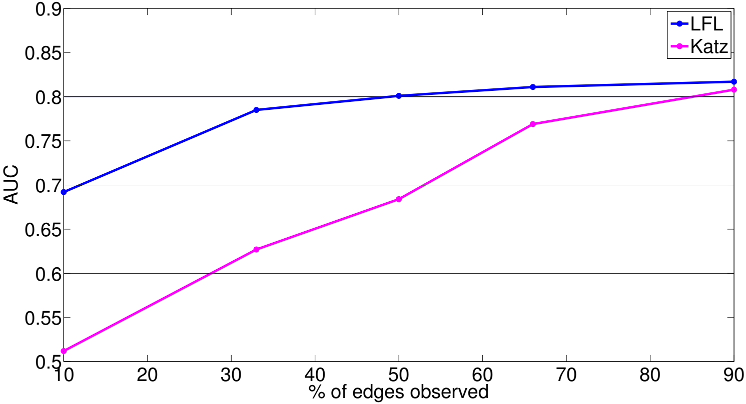
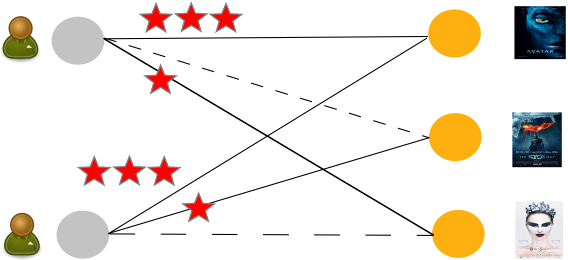


Figure: Results on dataset of military conflict relationships.

Outline

Bipartite link prediction: movie recommendation

- Recall the movie recommendation problem:



- This is a type of bipartite link prediction

Probability model for bipartite graphs

- Recall that the basic LFL model was

$$p(y|(r,c);\theta) \propto \exp((\alpha_r^y)^T \Lambda^y \alpha_c^y + (v^y)^T s_{rc})$$

- We will drop Λ, and consider separate vectors for the two sets:

$$p(y|(r,c);\theta) \propto \exp((\alpha_r^y)^T \beta_c^y + (v^y)^T s_{rc})$$

- Movie recommendation example:
  - Each user/movie has a collection of weights, representing characteristics for different ratings
  - Characteristics that make user rate 1 star $\neq$ those that make him rate 5 stars

Prediction and training: unordered versus numeric

- Unordered ratings $\rightarrow$ train to optimize log-likelihood
- Not desirable for numeric ratings
  - Difference between 1 and $5 \neq$ difference between 4 and 5
- Better alternative is to predict:

$$\hat{Y}_{rc} = \mathbb{E}[y] = \sum_{y=1}^{|\mathcal{Y}|} y p(y|(r,c);\theta)$$

and optimize using e.g. MSE

  - Expectation acts like a "summary function"
  - Standard latent feature model $\rightarrow$ single factorization

Assessing uncertainty

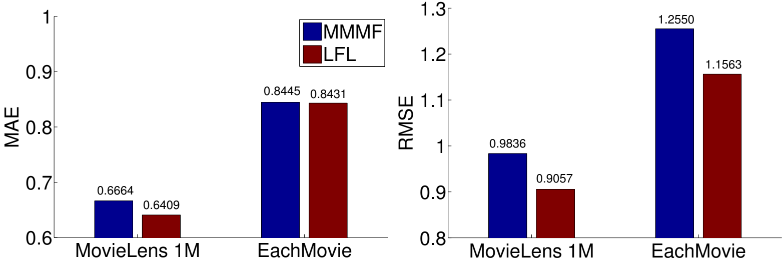
- For numeric ratings, we can also compute

$$\sigma_{rc}^2 = \mathbb{E}[y^2] - (\mathbb{E}[y])^2$$
$$= \sum_y y^2 p(y|(r,c);\theta) - \left(\sum_y y p(y|(r,c);\theta) \right)^2$$

- Quantifies estimated uncertainty of prediction
  - Could be combined with business rules
  - e.g. Protein-protein interaction: confidence in predicted link $<$ cost threshold $\implies$ do not run expensive test

Experiments on collaborative filtering - I

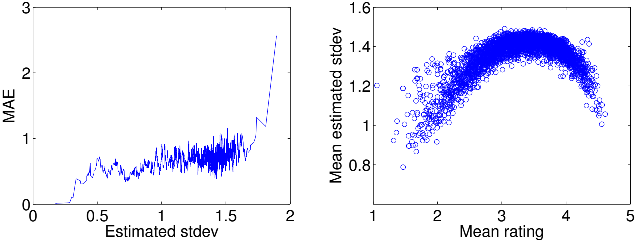
- Results on 1M MovieLens (6040 users, 3952 movies, 1 million ratings of 1-5 stars) and EachMovie (36,656 users, 1628 movies, 2.6 million ratings of 1-6 stars)



- Despite being more general, the LFL model is competitive with, yet faster than, the MMMF method [Rennie and Srebro, 2005]

Experiments on collaborative filtering - II

- Estimated uncertainty correlates with observed test set errors and average rating of movie:

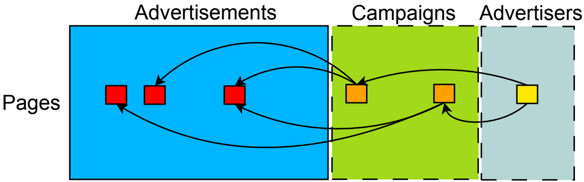


Lowest variance	Highest variance
Kazaam	Grateful Dead
Lawnmower Man 2: Beyond Cyberspace	The Rescuers
Problem Child 2	Prizzi's Honor
Meatballs III	Homeward Bound: The Incredible Journey
Pokemon the Movie 2000	The Fly

Figure: Result on MovieLens 1M

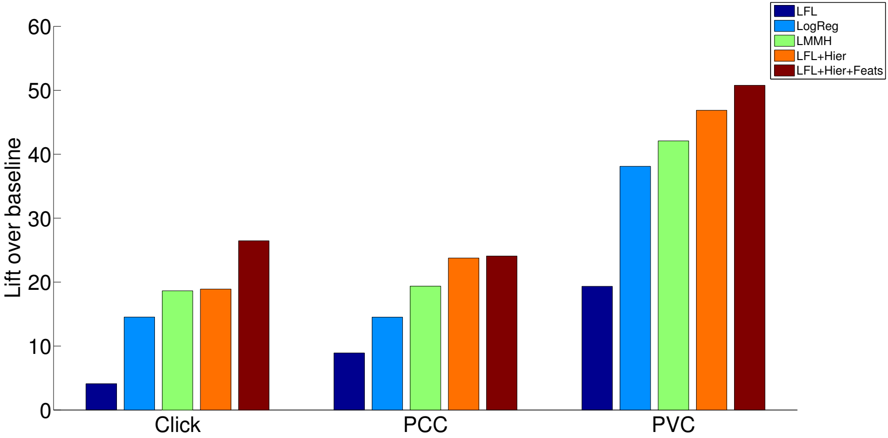
Coping with extreme sparsity

- In applications like response prediction for ads, labels are especially scarce
- Can use explicit features to pool together labels and estimate latent features at a coarser granularity



Experiments on response prediction

- Results on three large Yahoo! advertising datasets
- Latent feature gives lifts over state-of-the-art methods



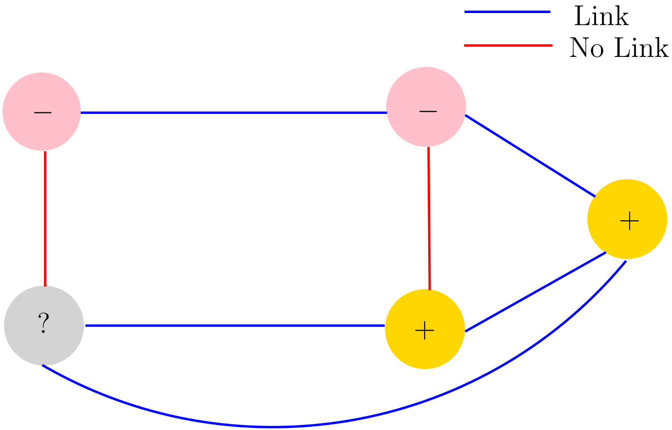
Outline

Dyadic label prediction

- Given dyadic relationships + labels for some dyad members, predict labels for all dyad members [Menon and Elkan, 2010b]

Graph perspective: within-network classification

- **Input**: Graph $G = (V, E)$, labels for subset $V' \subseteq V$ of nodes
- **Output**: Predicted labels for all nodes in $V - V'$
  - Called the within-network classification problem

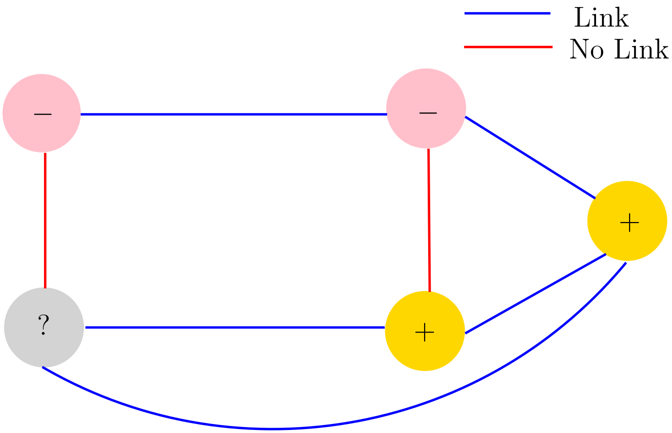


Dyadic label prediction formally

- **Training set**: $\{((r_i, c_i), y_i)\}_{i=1}^{N} + \{(r_j, z_j)\}_{j=1}^{N'}$
  - Now the rows, say, have additional labels
- **Output**: $f : \{1, \ldots, m\} \to \mathcal{Z}$
  - Optionally, predict missing dyadic relations too
  - We allow $\mathcal{Z} = \{0, 1\}^L$, i.e. multi-label prediction
- Numerous applications:
  - Will a user respond to an ad campaign based on his movie preferences?
  - Is a user "suspicious" by virtue of his links?
  - ...

“Reduction” to a dyadic prediction problem - I

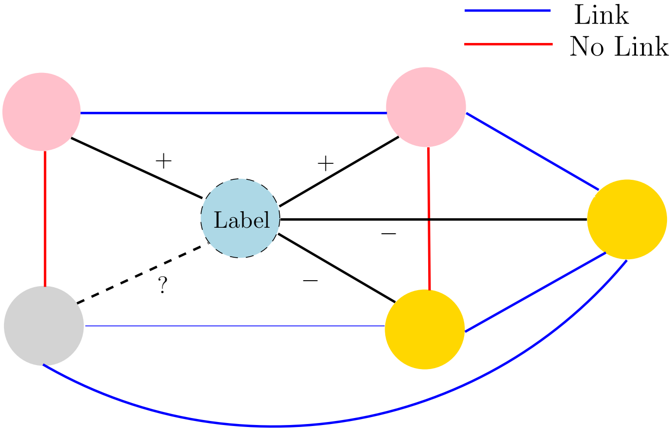
- Naïve solution: create a synthetic node for each label
  - Connected to each node, edge annotated by the label value



- Now learn latent features for the labels, reconstruct as normal

"Reduction" to a dyadic prediction problem - I

- Naïve solution: create a synthetic node for each label
  - Connected to each node, edge annotated by the label value



- Now learn latent features for the labels, reconstruct as normal

“Reduction” to a dyadic prediction problem - II

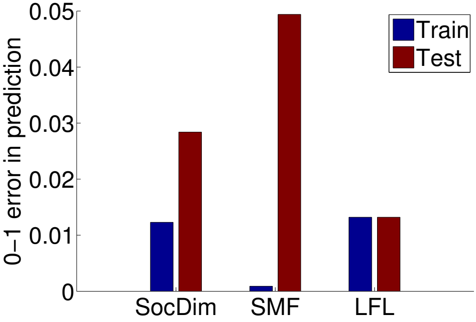
- **Issue**: Generally, our final goal is predicting the labels
  - Reconstructing the dyadic relationships is a means to that end
- Weight the loss on “label nodes” to reflect this:

$$\text{Objective} = \text{Loss}(\text{Nodes}) + \mu\text{Loss}(\text{Labels})$$

  - μ represents tradeoff between supervision and label accuracy
  - Must be careful not to overfit on the labels

Experimental results: `senator`

- Data comprises "Yea"/"Nay" votes of 101 senators concerning 315 bills
- Label $=$ whether senator is a Republican/Democrat
- LFL does best on this dataset:



Outline

Conclusion

- Predicting labels in graphs is an important and far-reaching problem
- Latent features seem to be a promising solution
  - Scalable
  - Accurate
  - Incorporate multiple sources of information

References I

Menon, A. K. and Elkan, C. (2010a).
A log-linear model with latent features for dyadic prediction.
In *ICDM 2010*, pages 364–373, Washington, DC, USA. IEEE Computer Society.

Menon, A. K. and Elkan, C. (2010b).
Predicting labels for dyadic data.
Data Mining and Knowledge Discovery (ECML/PKDD special issue), 21:327–343.

Menon, A. K. and Elkan, C. (2011).
Link prediction via matrix factorization.
Submitted to ECML/PKDD '11.

Rennie, J. D. M. and Srebro, N. (2005).
Fast maximum margin matrix factorization for collaborative prediction.
In *ICML '05*, pages 713–719, New York, NY, USA. ACM.

Sarkar, P., Chen, L., and Dubrawski, A. (2008).
Dynamic network model for predicting occurrences of salmonella at food facilities.
In *BioSecure '08*, pages 56–63, Berlin, Heidelberg. Springer-Verlag.