

Supervised learning for graph-structured data

Hugh Chipman, Acadia University

joint work with Charles Bouveyron (U Paris I), Erika Nahm (DND),
François Thériège (Ottawa U)

*UBC Department of Statistics,
June 2008*



Mathematics and Statistics at Acadia

FACULTY OF PURE AND APPLIED SCIENCES



Introduction: What are “Graph-structured Data?”

An example:



- A naughty US Company
- Over 125,000 emails from 184 employees published.
- Our “raw data” consist of email “header” information (sender, recipient(s), time message sent).
- We also have a label (management/other) for each employee.
- We want to predict the label based on the transaction information.

Two approaches to supervised learning

Convert raw transactions into “features” (predictors) in two different ways:

1. **How you talk**: Derive features that summarize transaction patterns for each node (e.g. number of messages sent).
 - $n \times p$ feature matrix (if we have n nodes)
2. **Who you talk to**: Generate a “sociomatrix”, which is a $n \times n$ binary matrix Y .
 - $Y_{ij} = 1$ if nodes i and j communicate, 0 otherwise.
 - Additional processing necessary before supervised learning.

Part I: predictions using “How you talk”

Raw email data (header information)

From: hugh@enron.com

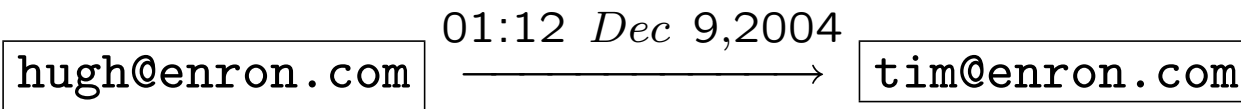
To: tim@enron.com

CC: sue@com.com, ken@enron.com

Date: 01:12 Dec 9, 2004 (ADT)

Subject: We're really in it now

Hi Tim, BlahBlahBlahBlah implicate BlahBlah criminal...



Sender	Recipient	Time (sec. since 1998)	Type
hugh@enron.com	tim@enron.com	217598400	To
hugh@enron.com	sue@com.com	217598400	CC
hugh@enron.com	ken@enron.com	217598400	CC

Approach:

- Convert email headers from 125,000 into a “feature matrix” for 184 email addresses.
- Built 23 node-centric features:
 - node-centric used here: communication density, degree, time of day, ...
 - Others possible, (graph-based, subject covariates like age)
- Usual classifiers + train/validate/test.
- Performance assessment via misclassification rates, lift curves, ROC curves.

Constructing features: The basic idea is simple

Convert raw transactions into *features* which summarize activity of each node (ie each person). Below is a toy example:

Raw data

From	To	Time	Type
hugh	tim	Thu Dec 20 1:17pm	To
hugh	sue	Thu Dec 20 1:17pm	CC
tim	sue	Thu Dec 20 5:05pm	To
tim	hugh	Thu Dec 20 5:27pm	To
hugh	tim	Sat Dec 22 7:12am	To

Processed data: the feature matrix

id	in degree	in volume	out degree	average #recip	% out weekday
hugh	1	1	2	1.5	50
tim	1	2	2	1	100
sue	2	2	0	NA	NA

Processing the Enron data:

- Raw data: 125,409 transactions, 184 email addresses
- Processed data: 184 email addresses, 23 features (84 × 23 feature matrix)
- Features used:

WeekdayIn	DaysIn	RelActivity
WeekendIn	DaysOut	InActivity
WeekdayOut	InDegree	OutActivity
WeekendOut	OutDegree	HiActivity
InVolume	Lifespan	LoActivity
OutVolume	OutRatio	DiffIn
PercentOut	AvgDegree	DiffOut
UniqueDays	AvgRecip	

- Response variable: Senior Management (Yes/No)

More on the features:

Types of information captured:

- Traffic flow on the graph: InVolume, OutVolume, PercentOut
- Graph topology (connections, ignoring traffic volume): InDegree, OutDegree
- Topology & Traffic combined: $\text{OutRatio} = \text{OutDegree} / \text{OutVolume}$
- When messages sent: WeekdayIn, WeekdayOut, WeekendIn, WeekendOut
- Daily summaries: Uniquedays (# unique days with activity), DaysIn, DaysOut, ...
- Overall existence: Lifespan = time between first & last transaction, $\text{AvgDegree} = \text{OutDegree} / \text{Lifespan}$
- Burstiness: $\text{HiActivity} = (\# \text{ gaps with duration} < 24\text{hr}) / (\text{OutVolume} - \text{LoActivity})$
- “Bulkiness” of events: AvgRecip = average number of recipients for outgoing messages.

So now we have a standard supervised learning problem.

- Apply a favorite learning algorithms to data. We used:
 - Logistic regression
 - Decision trees
 - Random forests (Breiman 2001)
 - Gradient boosting (Friedman 2001; Freund and Schapire 1997)
- Goals:
 1. Low misclassification rate
 2. Identify relevant features (ie feature selection)
- Usual issues:
 - Model tuning via cross-validation
 - Test set for honest assessment of generalization error

Performance: Enron Data

	Misclass. (s.e.)	AUC' (s.e.)
Logistic regression (LR)	.25 (.02)	.50 (.13)
Decision trees (DT)	.31 (.02)	.35 (.11)
Random forest (RF)	.18 (.03)	.52 (.11)
Boosted trees (BT)	.22 (.02)	.45 (.14)

- For misclass, about 33% are “senior management”.
- We want large AUC'.
 - AUC' is “area under receiver operating characteristic (ROC) curve”, scaled to lie between 0 (random guessing) and 1 (perfect classification).
- All methods better than random guessing

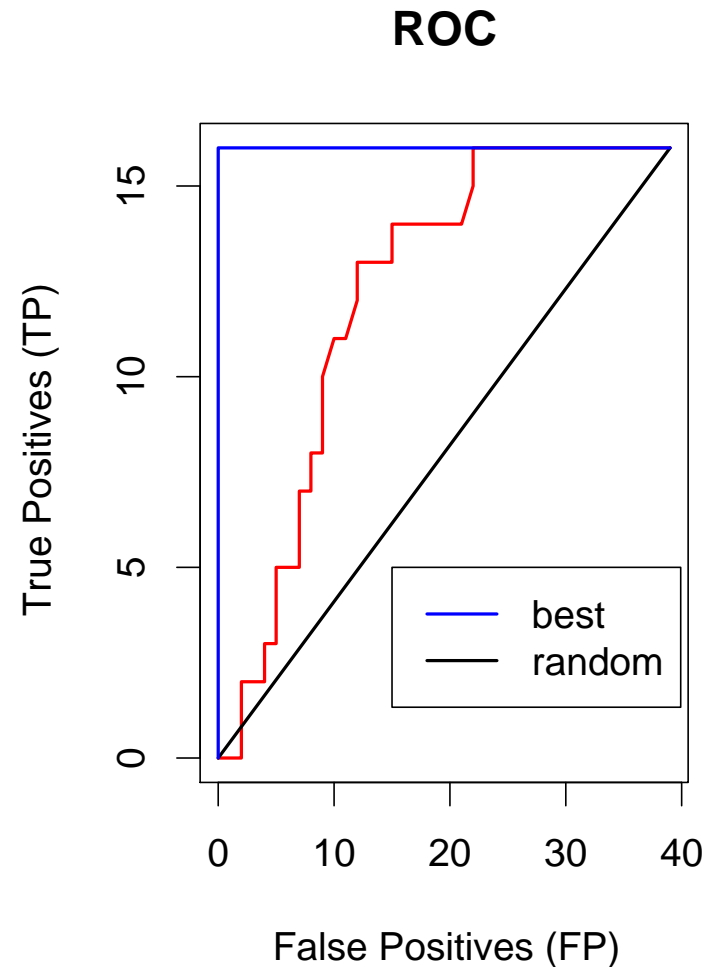
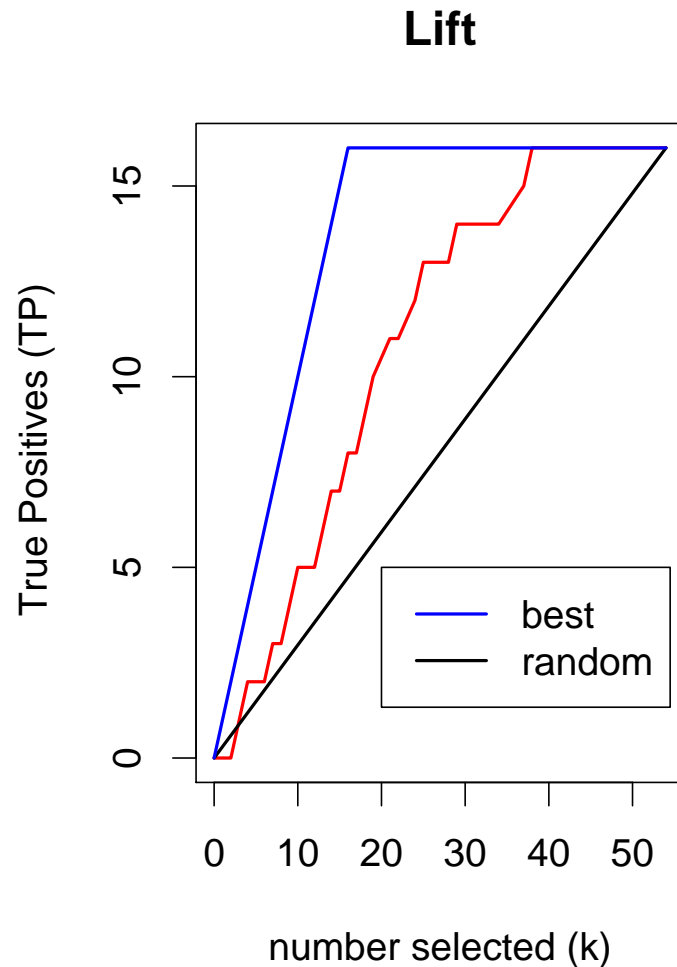
Rare target challenge...

- Another dataset
- 2.4 *million* transactions
- 21,533 nodes, but...
- **rare target:** only 652 nodes of “positive” class
- same 23 node-centric features
- find as many unknown positives as “quickly” as possible
 - produce ranking, and investigate “top-k” unknown nodes
- how to evaluate performance in view of objective and class imbalance?

(beginning of “rare target performance” digression)

Rare target performance: Lift, ROC curves

only difference in horizontal axis (“# selected” or “false pos.”)



Lift, ROC curves:

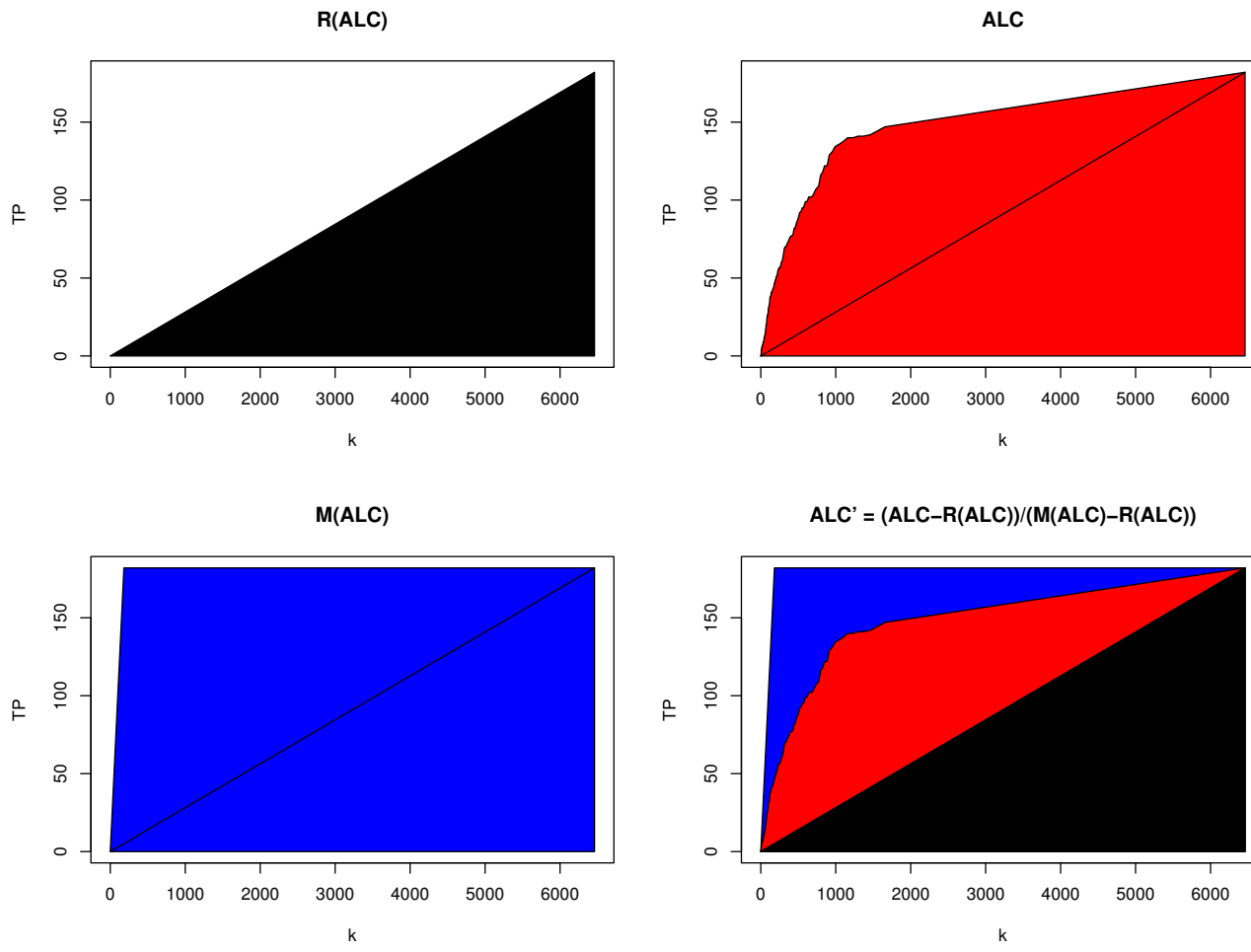
Common performance measures include:

- *AUC* (Area under ROC)
- *ALC* (Area under lift curve)

Result: $AUC' = ALC'$ (suitably normalized)

- normalization: rescale *ALC* so 0=random guessing, 1=perfect classification.
- Similar for *AUC'*.

ALC' and AUC':



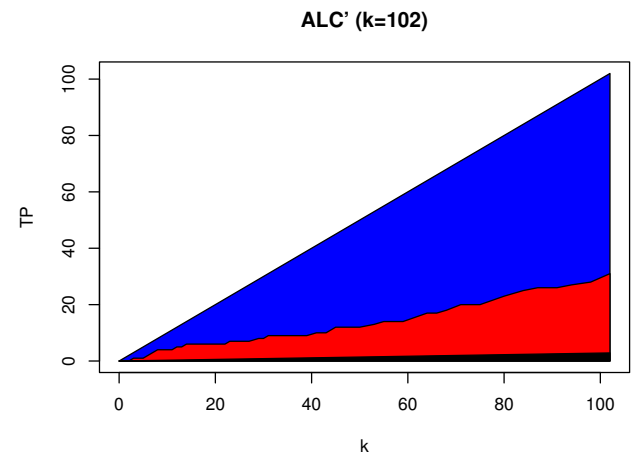
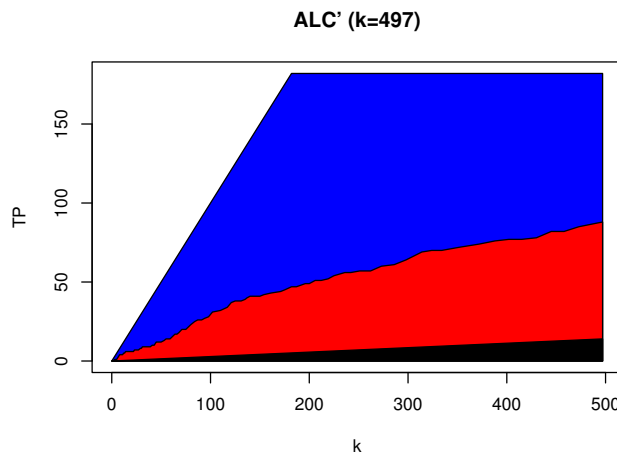
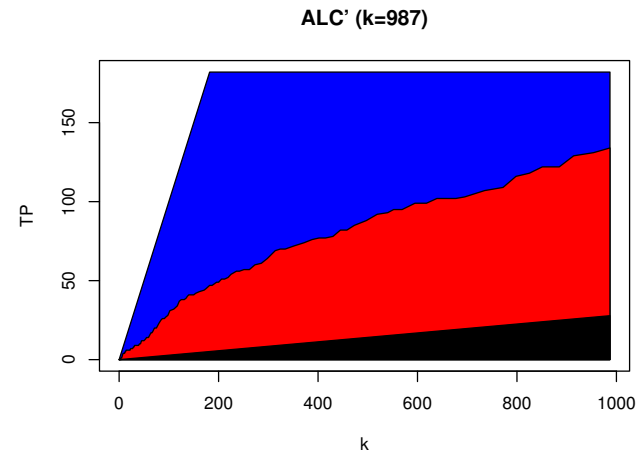
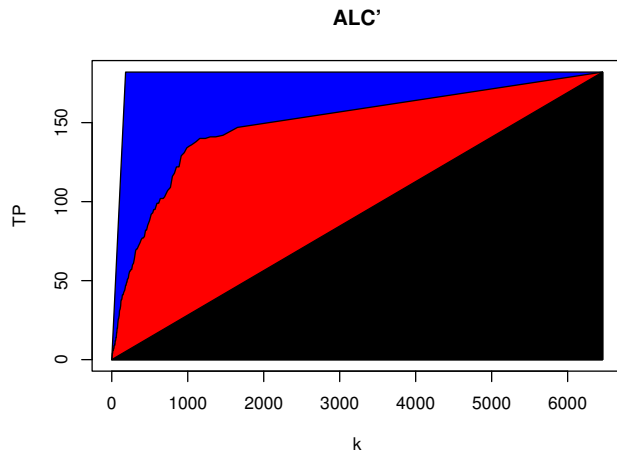
In lower right plot,

$$ALC' = \frac{\text{red}}{\text{red} + \text{blue}} = \frac{\text{improvement over random}}{\text{best possible}}$$

Improvement: truncated ALC

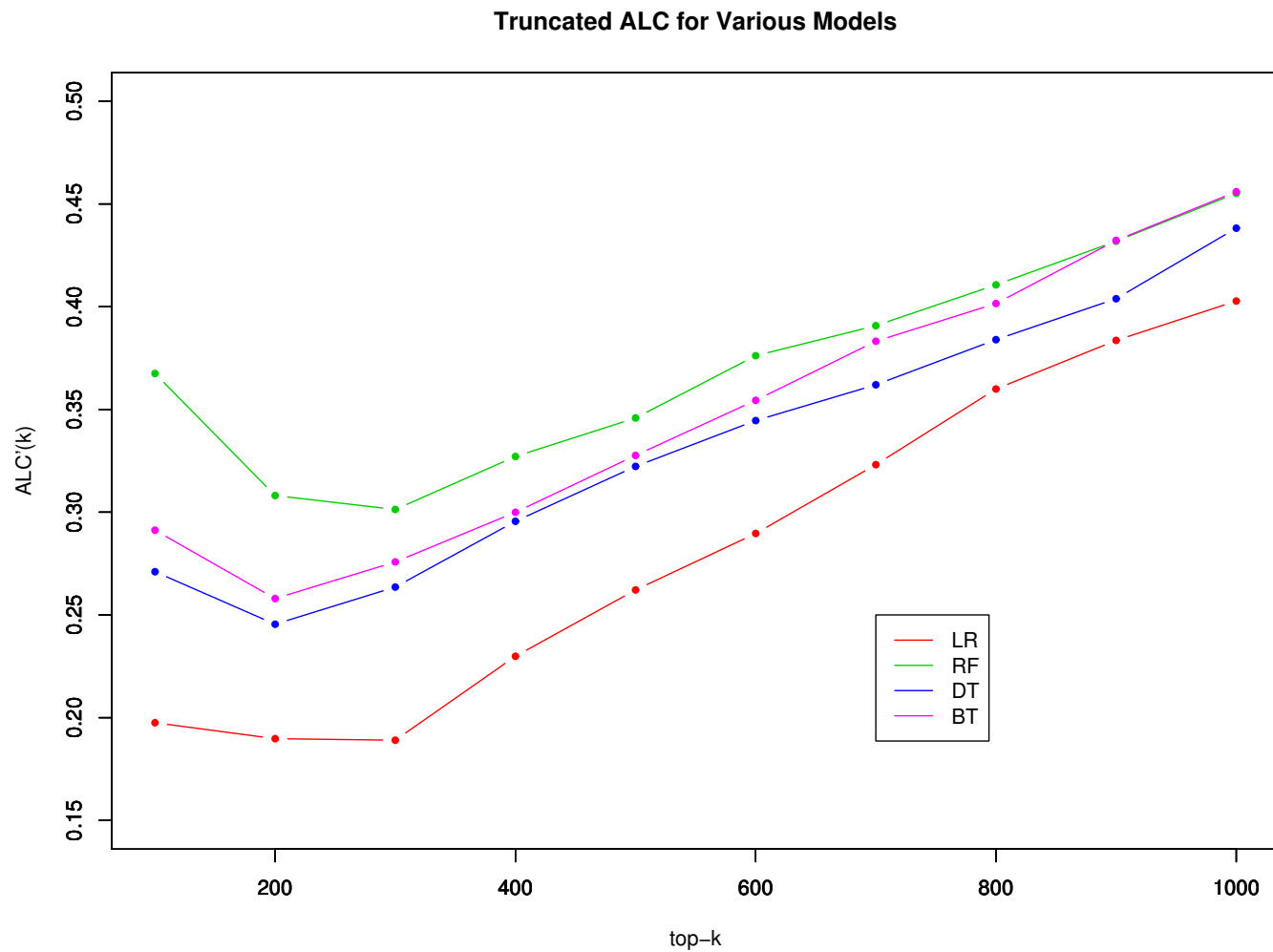
- AUC truncation problematic since horizontal axis (FP) is random.
- ALC'_k : normalize & truncate ALC, selecting top k cases.
- Various truncations shown on next slide.

Improvement: truncated ALC



ALC'_k results: rare target ex.

ALC'_k as a function of truncation point k :



ALC'_k results: rare target ex.

Significance? Derive $\text{Var}(ALC'_k)$

- Here, $\text{se}(ALC'_k) \leq 0.023$
⇒ all quite significant.

(end of “rare target performance” digression)

Part II: predictions using “Who you talk to”

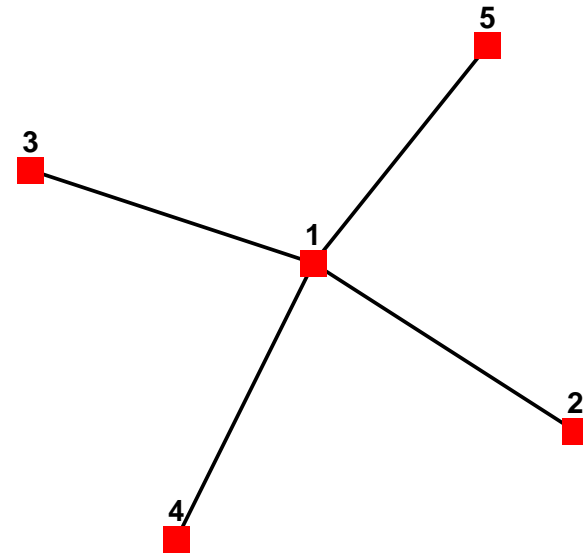
The representation of a social network

A social network is usually represented by a $n \times n$ sociomatrix Y :

- $Y_{ij} = 1$ if a tie (or edge) exists between the nodes i and j ,
- $Y_{ij} = 0$ otherwise.

$$Y = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

(a) sociomatrix



(b) social network

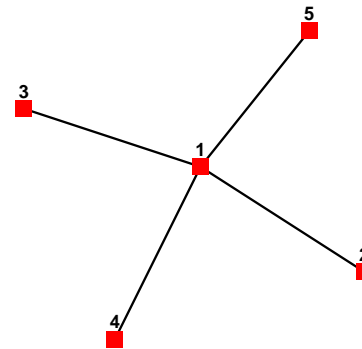
For our problem, we generate Y_{ij} by thresholding the number of $i \leftrightarrow j$ transactions.

Supervised learning for a social network

- In addition to the sociomatrix Y , we observe a response variable (e.g. a label) for each node in the network.
- Goal: predict label using sociomatrix information.
 - That is, predict label on the basis of “who talks to who”.
- In example below, suppose node 1 had label 1, all other nodes had label 0.

$$Y = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

(a) sociomatrix



(b) social network

Supervised learning for a social network

Example: Add-Health data

- 69 students
- Grades 7-11
- Observe “friendship” relations in form of sociomatrix Y
- Predict grade using information contained in Y

Supervised learning for a social network

Problem: can't use sociomatrix Y directly.

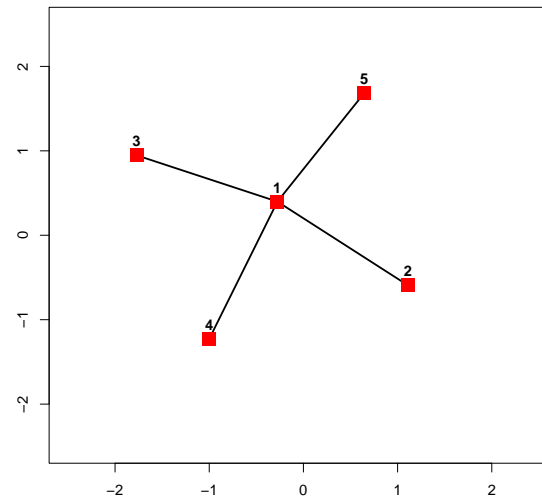
- For n labeled individuals, sociomatrix Y is $n \times n$.
⇒ Too many predictor variables.
- As new test points are added, the dimension of Y grows.

Solution:

- Learn a low-dimensional latent space “projection” of Y .
- New test points can be projected into the same latent space.

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(a) Input sociomatrix



(b) Learned latent space

The representation of a social network

Want to develop a probability model $P(Y)$

- We could assume Y_{ij} is independent of the other ties in the network:

$$P(Y|\theta) = \prod_{i \neq j} P(Y_{ij}|\theta).$$

- This independence assumption might be unreasonable.
- Think of a “friendship relation” $i \leftrightarrow j$
 - * Suppose $1 \leftrightarrow 2$ and $2 \leftrightarrow 3$ (i.e. $Y_{12} = 1, Y_{23} = 1$.)
 - * Then $1 \leftrightarrow 3$ (i.e. $Y_{13} = 1$) is more probable, since 1 knows 2 and 2 knows 3.
- For some forms of model (and corresponding θ), the independence assumption be reasonable (see next slide).

The LSN model

Hoff *et al.* (2002) proposed the **Latent Social Network (LSN) model** which takes the following form:

$$\text{logit}(P(Y_{ij} = 1|\theta)) = \alpha - \|Z_i - Z_j\|,$$

where:

- $\text{logit}(P) = \log(P/(1 - P))$,
- $\theta = \{\alpha, Z_1, \dots, Z_n\}$ are the parameters,
- α is the prior probability of an existing link,
- Z_i is the p -dimensional latent position of node i .

The model hypothesizes the existence of a p -dimensional “latent social space”. Two nodes that are “close” in this space have a large probability of a tie.

Coming back to the **independence** question, we are saying that **conditional on latent positions** the ties are independent.

Learning the latent representation

- Note that the positions (the Z_i 's) are *not* observed.
- We must estimate (learn) these parameters.
- This is similar to “multidimensional scaling” (MDS)
- The parameters can be estimated by maximizing the likelihood:

$$\log(L(\theta)) = \sum_{i \neq j} \left[y_{ij} (\|Z_i - Z_j\| - \alpha) - \log(1 + \exp(\|Z_i - Z_j\| - \alpha)) \right].$$

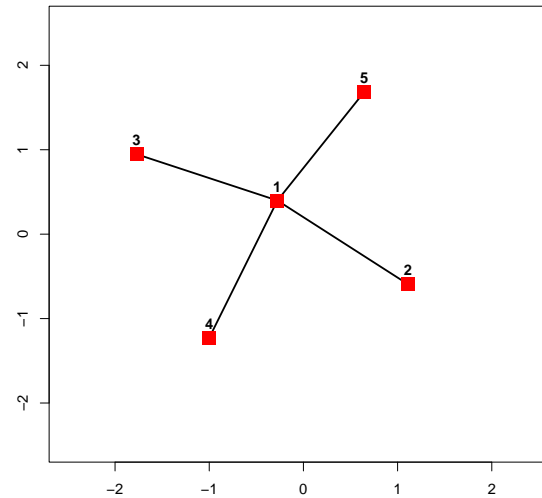
Learning the latent representation

Instead of using the MCMC procedure of Hoff *et al.* (2002), we use a faster approach:

1. Find a starting solution using MDS.
2. Maximize the likelihood on $\theta = \alpha, Z_1, \dots, Z_n$, using a stochastic optimization method (simulated annealing).

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 \\ \begin{array}{c} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array} & 1 & 0 & 0 & 0 & 0 \\ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} & & 1 & 0 & 0 & 0 \\ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} & & & 1 & 0 & 0 \\ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} & & & & 1 & 0 \\ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} & & & & & 1 \end{pmatrix}$$

(a) Input sociomatrix

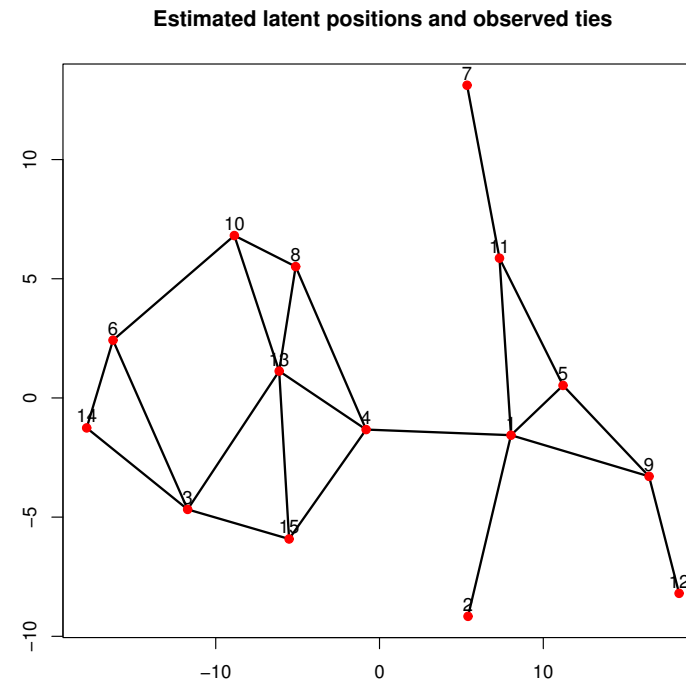
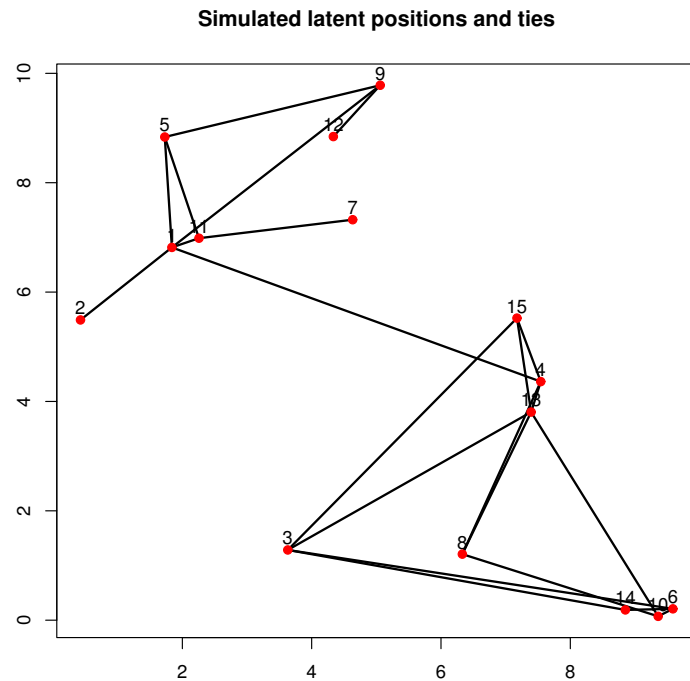


(b) Learned latent space

Learning the latent representation: An example

Left: Simulated data, with “true” positions in latent space and simulated ties.

Right: Estimated positions



Is this “overfitting” or just a good representation of the sociomatrix?

A supervised classification approach

Recall that we want to predict a label for each node.

Our latent supervised classification (LSC) approach consists of:

1. learning phase :

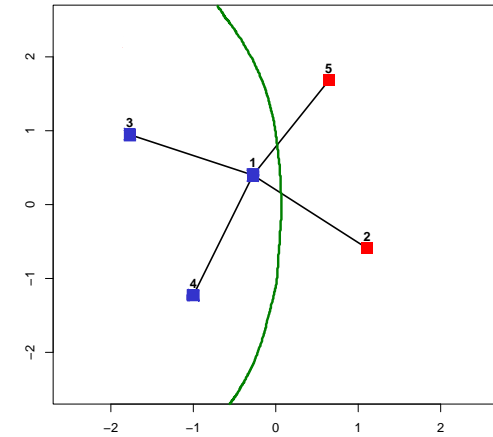
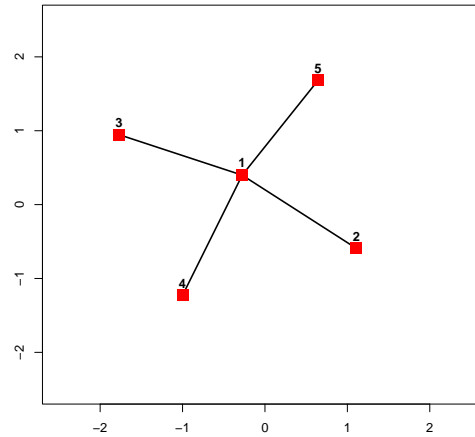
- build a latent representation of the network using the LSN model,
- learn a supervised classifier in the latent space.

2. classification phase :

- new network nodes are **projected** into the latent space,
- new nodes are then assigned to one of the classes.

Learning phase of LSC

$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$



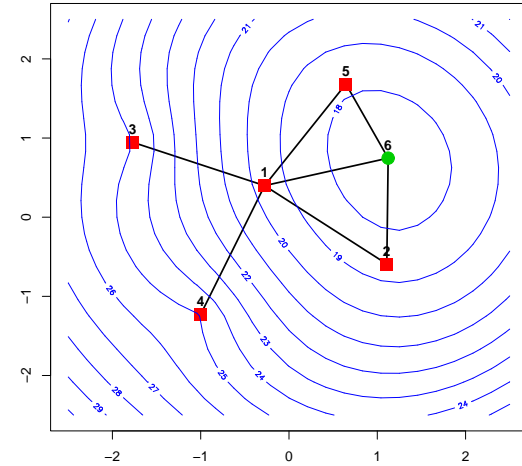
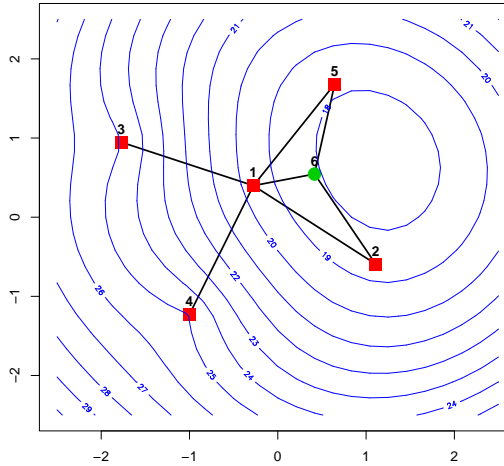
(a) Input sociomatrix (b) Learned latent space (c) supervised classifier

In the learning phase:

1. The latent space associated with the network is built via maximum likelihood estimation.
2. A supervised classifier is then learned in the latent space.
 - Any classifier can be used, although radial symmetry desirable.

Classification phase of LSC

$$\begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$



(a) Input sociomatrix

(b) Initial position

(c) Final position

In the classification phase:

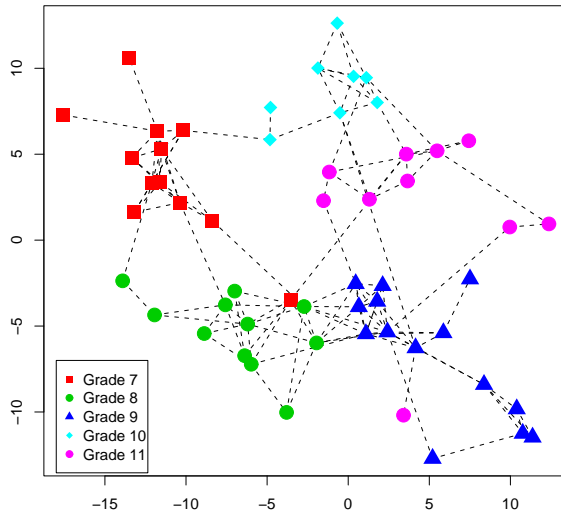
1. The latent position of the new node is estimated:
 - by maximizing the likelihood as a function of the new position(s) (i.e. new Z 's).
 - This preserves locations of existing nodes.
2. The new node is assigned to one of the k classes:
 - using its position in the latent space
 - according to the classification rule

Experimental study: the Add-health dataset

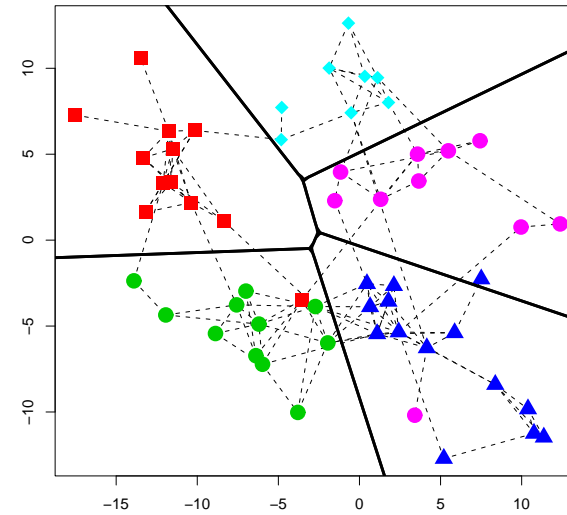
In this data, we consider

- 69 students
- Grades 7-11
- Use observed “friendship” relations to construct latent space
- Predict grade on basis of latent space representation.
- Test set: position new kids in latent space based on their friendship relations, then predict using supervised model.

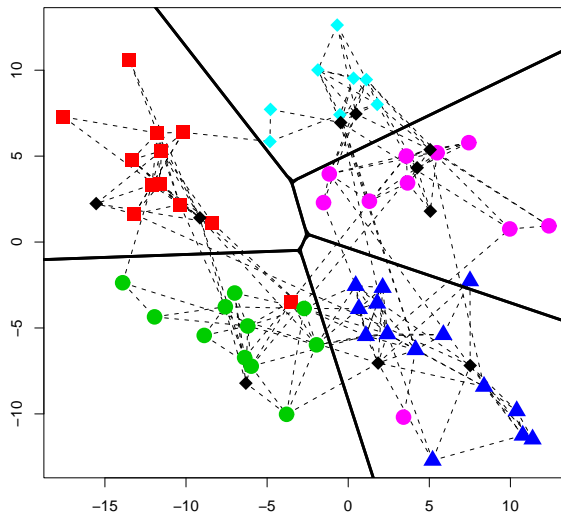
Experimental study: the Add-health dataset



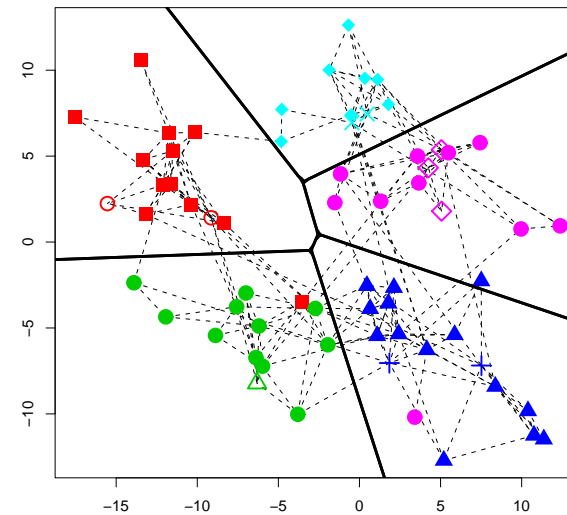
(a) Learned latent space



(b) Learned classifier (LDA)



(c) Projection of a new node



(d) Classification of a new node

Experimental study: the Add-health dataset

Classification accuracy (test set):

Dimension p	2	3	4	5
LSC+SVM	0.90	0.86	0.87	0.89
LSC+LDA	0.86	0.89	0.89	0.91
KNN-based	—————0.90—————			

Notes:

1. Test-set results are averages for 25 replicates of train/test splitting.
2. KNN-based method doesn't use LSN. It uses all "neighbours" that have a tie with the node we want to predict.
3. Most standard errors are 0.025, so differences are not significant.
4. Increasing dimension might improve accuracy.

Conclusion and further work

We proposed two supervised classification approaches:

- Transactions → “how you talk” features → supervised learning.
- Transactions → sociomatrix → latent space → supervised learning.

Combining approaches is an obvious next step.

More questions than answers: other future work and ideas

- Train/test division is tricky: for the initial learning of the latents space, we “lose” the edges between train and test nodes.
 - At the transaction level this complicates things.
- Specific questions for “latent space” approach:
 - The positioning of new nodes in an existing space could be an interesting diagnostic for studying dynamic networks.
 - Is the probability model any better than multidimensional scaling?
 - Use of frequency and/or directional information on edges rather than just 0/1 data (Poisson latent social network).