

# Mining Networks

## Techniques and Challenges

---

Sam Steingold

<<http://sds.podval.org/data/clust.ps>>

# Outline

---

- The problem
  
- Mining methodology
  - Individual threat level
  - Group detection
  - Classification
  
- Evaluating results
  - L-quality
  - Traditional metrics
  - Proficiency

# Typical Problem

---

A law enforcement agency has activity logs for some suspect (and not!) individuals (their communications, travels) et al and wants to find out what the terrorist cell(s?) are and what they are planning to do.

- Figure out who is connected to whom and how strongly
- What the teams are
- What the plans are
- Who is bad and who is not

# Challenges

---

- Input data (activity logs) is noisy in many ways:
  - Some information is incorrect (no phone call was made)
  - Some information is missing (we do not know of an important trip)
  - Some names could be misspelled (there are dozens of ways to spell Qaddafi in English)
  
- Graph size may require very efficient algorithms
  - Have to resort to heuristics instead of using full search
  
- The targets are scarce
  - There are very few bad guys out there
  - Data collections methods are very powerful (produce a lot of data)

# Methodology

---

- Clean up and convert activity logs to a network (graph)
- Group individuals into teams
- Build model(s) predicting whether the team or an individual is malicious

# Data cleaning

---

- Merge individuals
  - Different spelling of the same name
  - Other aliases
- Check for conflicts
  - A person can only be at one place at one time
  - Some places can contain a limited number of individuals
- Can do much much much much much much more.....
  - Know when to stop (diminishing returns)

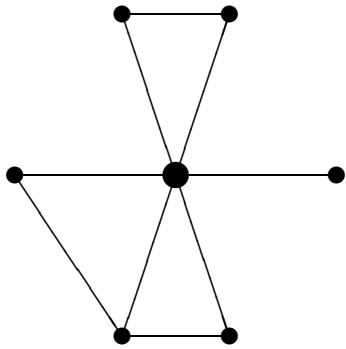
# Individual threat level

---

Clustering Coefficient of a person  $z$  with  $N$  neighbors in graph  $(V,E)$ :

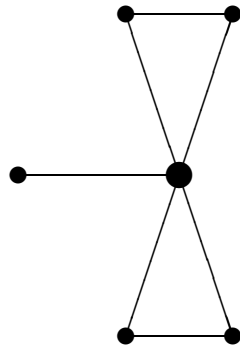
$$c = \frac{\#\{(x, y) \mid (z, x), (z, y), (x, y) \in E\}}{N(N-1)/2}$$

the actual number of triangles in which  $z$  participates  
divided by the potential number of such triangles.



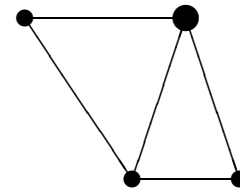
$$c = 3/15 = 0.2$$

$$t = 1.2$$



$$c = 2/10 = 0.2$$

$$t = 1.0$$



$$c = 2/3 = 0.66$$

$$t = 2$$

# Threat level $t=cN$

---

This selects people who participate in group activities as reflected in the communication network data

By combining clustering coefficients with numbers of connections we focus on the individuals that are most active and tied to active peers

Assumes that the data is not a random set, but is based on observations of suspicious individuals and activities

Helps eliminate the proverbial "pizza delivery guys"



# Distance in a graph (definition)

---

Define the distance between two individuals to be the time necessary to pass a fixed volume of information from one to the other, i.e., the inverse of the bandwidth of the total network connection along all paths between them.

More formally:

- consider the communications graph as an electrical network
- each communication is assigned conductance (inverse of resistance) proportional to its bandwidth
- the resistance between nodes satisfies the triangle inequality, is symmetric and positive

# Distance in a graph (computation)

---

- Computed using the Kirchhoff laws.
- Requires inverting a (usually sparse) adjacency matrix of size  $\#V$  (number of vertexes).

# Group Detection: Objective function

---

Cohesiveness of a group of individuals:

- Describes how closely tied to each other the members of this group are

The average distance between pairs of group members: the smaller this average is, the more closely connected the members of the group are.

The inverse of this average is the cohesiveness of the group.

# Group Detection: Algorithm

---

Use greedy search to find groups of maximal cohesiveness

- start with a single individual
- first iteration:
  - add its closest neighbor
- subsequent iterations:
  - find candidates: the individuals who improve the cohesiveness of the group, and add the best one to the group
- the search is complete when there are no candidates, that is, when no remaining individuals could be added to the group to improve the group's cohesiveness

We call these groups "cohesive clusters".

To find all cohesive clusters, we run the search algorithm once for each individual in the network, where the individual serves as the starting point for the search.

# Group Detection: Refinement

---

The clustering algorithm could potentially get stuck in a local maximum where adding one person to a group does not increase the cohesiveness, but adding, two or three at a time may increase it.

- after we finish the search, we try to merge overlapping cohesive clusters looking for a cluster with a better cohesiveness than at least one component.
- when we cannot find a person that would increase the cohesiveness, we try pairs of discarded candidates from the previous iteration steps.

In practice, however, the simple search usually finds so many clusters that pruning appears a bigger issue than finding more clusters.

# Threat Detection

---

Now that we have

- cohesive clusters (closely knit groups of individuals)
- clustering coefficients (indcates whether an individual is prone to work with groups)

We can try to identify who is actually bad.

- hand-made heuristics
- prefictive modeling

# Naive Bayesian Learning

---

Bayes, 1763:

$$P(A_i|C) = \frac{P(A_i)P(C|A_i)}{\sum_{i=1}^n P(A_i)P(C|A_i)}$$

C - discrete class, vector A - attributes with discrete values

Predict C=c from A=a when P(C=c | A=a) is maximal

Assume that the components of A are independent (given C).

From Bayes:

$$P(C = c|A = a) = P(C = c) \frac{\prod_{i=1}^n P(A_i = a_i|C = c)}{P(A = a)}$$

"Maximum likelihood" probability estimate

$$P(A_i = a_i|C = c) = \frac{\text{count}(A = a \wedge C = c)}{\text{count}(C = c)}$$

# Naive Bayesian Learning (math 1)

---

Two classes (0&1),  $a$  - attribute value vector for a test example,

$$b_0 = P(C = 0), b_1 = P(C = 1) = 1 - b_0$$

$$p_{j0} = P(A_j = a_j | C = 0), p_{j1} = P(A_j = a_j | C = 1)$$

Then ( $z$  - normalizing constant):

$$p = P(C = 1 | A = a) = b_1 \prod_{j=1}^k p_{j1} / z$$

$$q = P(C = 0 | A = a) = b_0 \prod_{j=1}^k p_{j0} / z = 1 - p$$

$$\log p - \log q = \sum_{j=1}^k (\log p_{j1} - \log p_{j0}) + \log b_1 - \log b_0$$

Let  $w_j = \log p_{j1} - \log p_{j0}$ , and  $b = \log b_1 - \log b_0$

# Naive Bayesian Learning (math 2)

---

Then the log odds

$$\log \frac{1-p}{p} = - \sum_{j=1}^k w_j - b$$

$$p = \frac{1}{1 + e^{-\sum_{j=1}^k w_j - b}}$$

In general, let each  $A(j)$  take  $v(j)$  values, and let

$$w_{jj'} = \log P(A_j = a_{jj'} | C = 1) - \log P(A_j = a_{jj'} | C = 0)$$

$$P(C(x) = 1) = \frac{1}{1 + e^{-\left(\sum_{j=1}^k \sum_{j'=1}^{v(j)} I(A_j(x)=a_{jj'}) w_{jj'}\right) - b}}$$

Here  $I$  is the indicator function, i.e.,  $I(X)=1$  if  $X$  is true and  $I(X)=0$  if  $X$  is false.

# Bayesian Classifier

---

- Equivalent to a perceptron with a sigmoid activation function and sparse encoding of attributes (a separate input for each possible value of each attribute).
- A non-parametric, non-linear extension of logistic regression, which, in turn, is equivalent to a perceptron with a sigmoid activation function and a single input node for each attribute (attribute values are encoded with their magnitude).

# Boosting (Freund & Schapire, 1995)

---

- Learn several classifiers
- Each classifier up-weights previously misclassified examples
- The final classifier outputs a weighted sum of all the classifiers

## Elkan, 1997:

---

- Boosted Naive Bayesian classifier is representationally equivalent to a multilayer perceptron with a single hidden layer.

# Advantages

---

- Low computational complexity:  $O(T \cdot e \cdot f)$  where
  - $T$  is the number of boosting rounds
  - $e$  is the number of examples
  - $f$  is the number of attributes (features)
  
- Real-time model updates
  
- Interpretable results
  - offers new statistically significant actionable insights
  
- Excellent performance
  - Fast training
  - Fast scoring
  - Good model quality

# Disadvantages

---

- The assumption of attribute independence is usually false
- The score returned by the model is usually either almost 0 or almost 1
  - Use selective classifier (keep a few best attributes)
  - Use special magic to convert scores to probability estimates
- Requires discrete attributes, thus necessitates a binning process
  - Best is "equal height" binning
  - Slow ( $e \cdot \log(e)$ ), compared to the actual training
  - Complicates real-time adaptive modeling: may require periodic re-binning
- Neural networks and SVM often exhibit (insignificantly) better model quality (at the price of slow training)

# Evaluating Prediction Results

---

Results can be

- numeric scores
  - use cut-off to convert them to...
  
- binary predictors

# Model quality: Lift curve

---

Let Cumulative Percent Hits  $CPH(M,p)$  be the percentage of targets in the first  $p\%$  of the list sorted by decreasing score of model  $M$ . Define

$$\text{lift}(M, p) = \frac{CPH(M, p)}{p}$$

- "lift curve" is the CPH vs percentile graph
  - (AKA receiver operating characteristic - ROC curve)
  
- If model A lift curve dominates (is strictly above) model B lift curve, then for any fixed cost/benefit matrix and for any  $p$ , the benefit of selecting the top  $p$  observations using the model A will be higher than the benefit of selecting the top  $p$  observations using B.

# Lift Table example

---

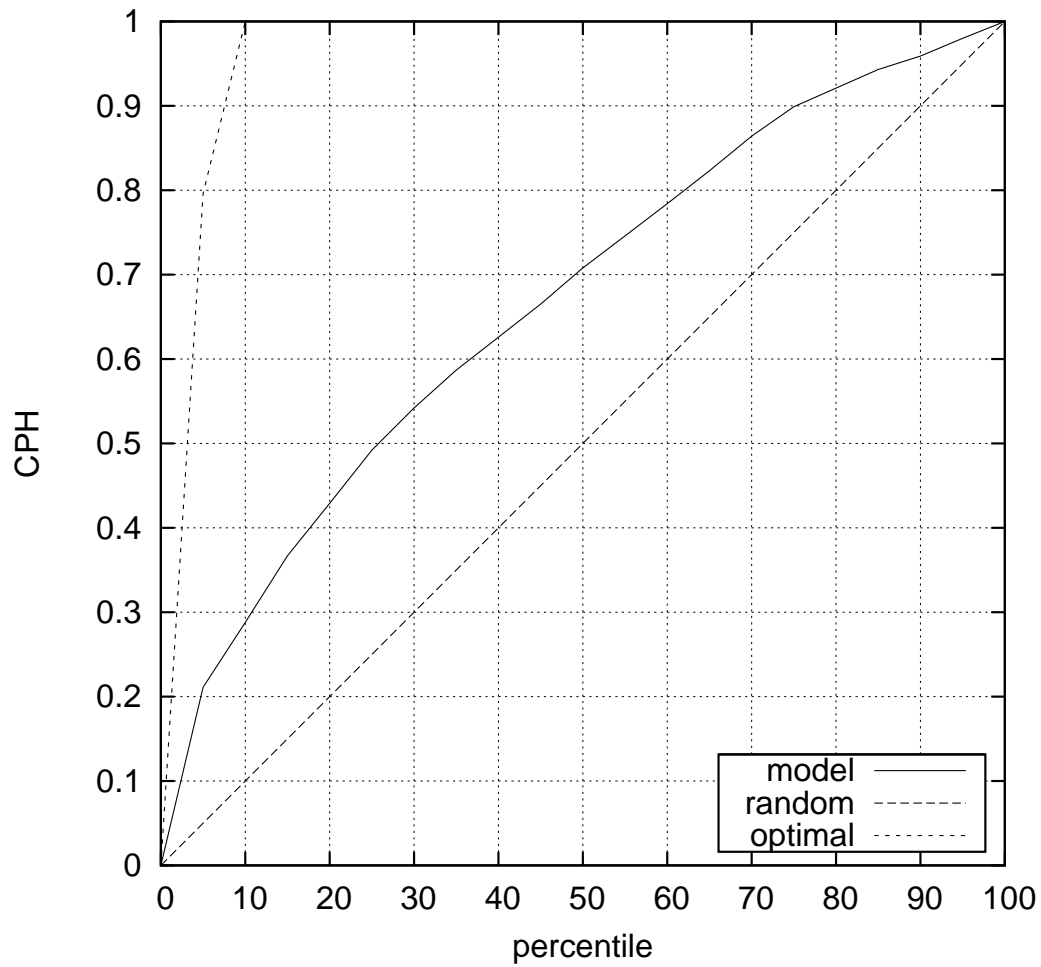
%	recs	hits	hits%	lift	model CPH	optimal CPH
5	1045	277	26.51	4.22	0.211	0.796
10	2090	378	18.09	2.88	0.288	1
15	3135	481	15.34	2.44	0.367	1
20	4180	563	13.47	2.15	0.429	1
25	5225	646	12.36	1.97	0.492	1
30	6270	711	11.34	1.81	0.542	1
35	7315	770	10.53	1.68	0.587	1
40	8360	821	9.82	1.56	0.626	1
45	9405	872	9.27	1.48	0.665	1
50	10450	929	8.89	1.42	0.708	1
55	11495	979	8.52	1.36	0.746	1
60	12540	1029	8.21	1.31	0.784	1
65	13585	1080	7.95	1.27	0.823	1
70	14630	1134	7.75	1.23	0.864	1
75	15675	1180	7.53	1.20	0.899	1
80	16720	1208	7.22	1.15	0.921	1
85	17765	1237	6.96	1.11	0.943	1
90	18810	1258	6.69	1.07	0.959	1
95	19855	1286	6.48	1.03	0.980	1
100	20900	1312	6.28	1	1.000	1

## legend:

- % - the percentile (= CPH of the "random" model)
- recs - the number of records
- hits - the number of targets correctly identified
- hits% - the percentage of hits, which is the number of hits divided by the number of records
- lift - the lift, equal to hits%/b
- model CPH - the cumulative percent hits for the given model
- optimal CPH - the cumulative percent hits for the ideal model

# Lift Curve example

---



# Sum of Cumulative Percent Hits

---

$$\text{SumCPH}(M) = \int_0^1 \text{CPH}(M, p) dp$$

$$\text{SumCPH}(\text{random}) = 1/2$$

$$\text{SumCPH}(\text{ideal}) = 1-b$$

SumCPH of a predictive model will be between these two values

If  $\text{SumCPH} < 1/2$ , just use the opposite of the model!

# SumCPH is bad!

---

- SumCPH is not intuitive - the meaning of 0.69 is not clear
- SumCPH by itself does not tell us how close it is to the maximum
- The higher the base rate, the lower is the maximum possible SumCPH and the measure of lift quality should account for it

# Lift quality

---

Normalize SumCPH so that the ideal model will be 1 and the random 0:

$$\begin{aligned} \text{L-quality}(M) &= \frac{\text{SumCPH}(M) - \text{SumCPH}(\textit{Random})}{\text{SumCPH}(\textit{Best}) - \text{SumCPH}(\textit{Random})} \\ &= \frac{2\text{SumCPH}(M) - 1}{1 - b} \end{aligned}$$

L-quality(random) = 0

L-quality(ideal) = 1

L-quality of a predictive model will be between these two values:

L-quality(above) = 35.6%

# Evaluating Detection: Discrete (table)

---

		<b>C</b> (correct classification)	
		yes	no
<b>D</b> (detected classification)	yes	<b>tp</b> (true positive)	<b>fp</b> (false positive)
	no	<b>fn</b> (false negative)	<b>tn</b> (true negative)

Need to look at all four of tp,fp,fn,tn to fully assess performance.

# Evaluating Detection: Discrete (metrics)

---

Accuracy =  $(tp+tn)/(tp+fp+fn+tn)$

- Often meaningless, especially when the targets are scarce
  - "San Diego weather forecast"

Utility =  $cost(tp)*tp/N + \dots$

- Have to know the dollar value/cost of each quadrant

Other quality measures evaluate a particular aspect of the result table:

- Precision =  $tp/(tp+fp) = P(C=T|D=T)$ 
  - what percentage of suspects are guilty
- Recall =  $tp/(tp+fn) = P(D=T|C=T)$ 
  - what percentage of bad guys are arrested
- Specificity =  $tn/(tn+fp) = P(D=F|C=F)$ 
  - what percentage of the innocents are not harrassed

and are usually insufficient alone,

especially when the base rate (prevalence) is low

# Evaluating Detection: Requirements

---

We need a single real parameter, which

- summarizes the result table
- is sensitive to low values of  $tp$  or  $tn$
- can be interpreted mathematically

E.g., the harmonic average of the recall and precision:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F-score satisfies the first two requirements,  
but what exactly does it measure?

# Evaluating Detection: Proficiency

---

The new information-theoretic performance measure satisfies all requirements:

- proficiency =  $I / H$
- $I = I(C;D)$  is the mutual information between the correct classification and the predicted classification, i.e., the amount of information that the detector outputs provide about the true states
- $H = H(C)$  is the entropy of the correct classification, i.e., the amount of information the detector would have to provide if it were to achieve ideal error-free performance

# Proficiency = I / H

---

$$I = \sum_{j=1}^2 \sum_{k=1}^2 P_{cd}(j, k) \log_2 \frac{P_{cd}(j, k)}{P_c(j)P_d(k)}$$

$$H = - \sum_{j=1}^2 P_c(j) \log_2 P_c(j)$$

## Properties:

- 0 when C & D are independent ( $P_{cd}=P_cP_d$ )
- 1 when  $fn=fp=0$
- Constrains both precision and recall, e.g.,
  - proficiency = 0.5 implies recall, precision  $\geq 0.55$

# Conclusions

---

We developed a methodology for threat discovery based on

- group detection
- individual threat scores
- predictive modeling

and for result evaluation based on

- lift curve analysis
- information-theoretic performance metrics

# References

---

- J.V. White, S. Steingold, C. Fournelle. “Performance Metrics for Group-Detection Algorithms”. Presented at Interface 2004, Baltimore, MD, May 29, 2004.
- S. Steingold, C. Fournelle, J.V. White. “Clustering and Threat Detection”. Presented at 2005 AAI Spring Symposium on AI Technologies for Homeland Security, Stanford University, March, 2005.
- G. Piatetsky-Shapiro, S. Steingold "Measuring Lift Quality in Database Marketing", SIGKDD Explorations, Vol. 2:2, (2000), 81-86

# The End

---