Assessing the significance of (data mining) results

- Data D, an algorithm ***A***
- Beautiful result ***A*** (D)
- But: what does it mean?
- How to determine whether the result is interesting or just due to chance?
- Significance testing, hypothesis testing
- Classical analytical results
- The multiple testing problem
- Randomization

Examples

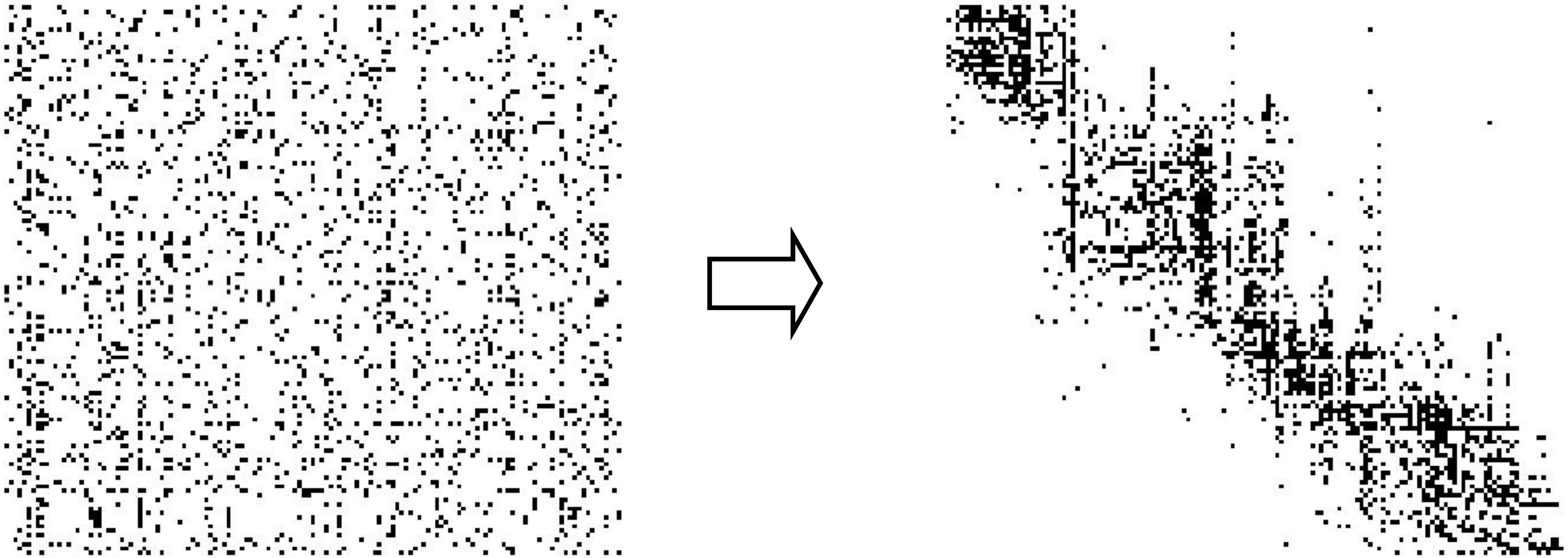
- Pattern discovery: association rules etc.
- From the data D we find a collection of nice patterns
- Significance of individual patterns is sometimes straightforward to test
- What about the whole collection of patterns? Is it surprising to see such a collection? Etc.

Examples

- Clustering, mixture modeling etc.: we always get a result
- BIC, AIC etc. can be used to test what is the ”best” number of mixture components
- But how to test if the whole idea of components in the data is good?

Examples

- Novel types of analysis: e.g., seriation in paleontology via spectral tecniques



- Looks nice: how important is the result?
  (Paleobiology 2006)

Classical methods

- Hypothesis testing
- Example: given two datasets C and D of real numbers, same number of observations
- We want to test whether the means of these samples are ”significantly” different
- Test statistic $t = (E(C) - E(D))/s$, where s is an estimate of the standard deviation
- The test statistic has (under certain assumptions) the t distribution with $2n-2$ degrees of freedom

Classical methods, cont.

- The result can be something like: ``the difference in the means is significant at the level of 0.01''
- That is, such a difference would occur by chance only in about 1 out of 100 trials of taking two samples of size |C|
- Problems
  – What if we are testing many hypothesis? (Multiple testing problem)
  – What if there is no closed form available?

Multiple testing problem

- Compute correlations of a random 1000x100 matrix

```
>> d = rand(1000,100);
>> [cc,pp] = corrcoef(d); % correlations and their
        significances
>> sum(pp(:)<0.01)
98
```

- 98 correlations had a significance value (p-value) less than 0.01
- p value 0.01: such a correlation occurs by random with probability 0.01
- 10,000 correlations (or 100x99 /2), so about 0.01 x 10,000\$ correlations should have such a p value

Bonferroni correction

- When testing for significance of B hypothesis, the significance values should be multiplied by B

```
sum(10000*pp(:)<0.01)
ans =  0
```

- Typically the hypotheses are somehow correlated
- Bonferroni correction is too conservative
- Extreme example: if all the hypotheses are actually the same
- Difficult to count the number of ``independent hypotheses'': how many did we have in the correlation example?

Randomization methods

- Goal in assessing the significance of results: could the result have occurred by chance
- Randomization methods: create datasets that somehow reflect the characteristics of the true data

Randomization

- Create randomized versions from the data D
- $D_1, D_2, \ldots, D_k$
- Run the algorithm $\mathcal{A}$ on these, producing results $\mathcal{A}(D_1), \mathcal{A}(D_2), \ldots, \mathcal{A}(D_k)$
- Check if the result $\mathcal{A}(D)$ on real data is somehow different from these
- **Empirical *p*-value**: the fraction of cases for which the result on real data is (say) larger than $\mathcal{A}(D)$
- If this is small, there is something in the data

Questions

- How is the data randomized?
- Can the sample $\{D_1, D_2, \ldots, D_k\}$ be computed efficiently?

- Can the values $\{\mathcal{A}(D_1), \mathcal{A}(D_2), \ldots, \mathcal{A}(D_k)\}$ be computed efficiently?

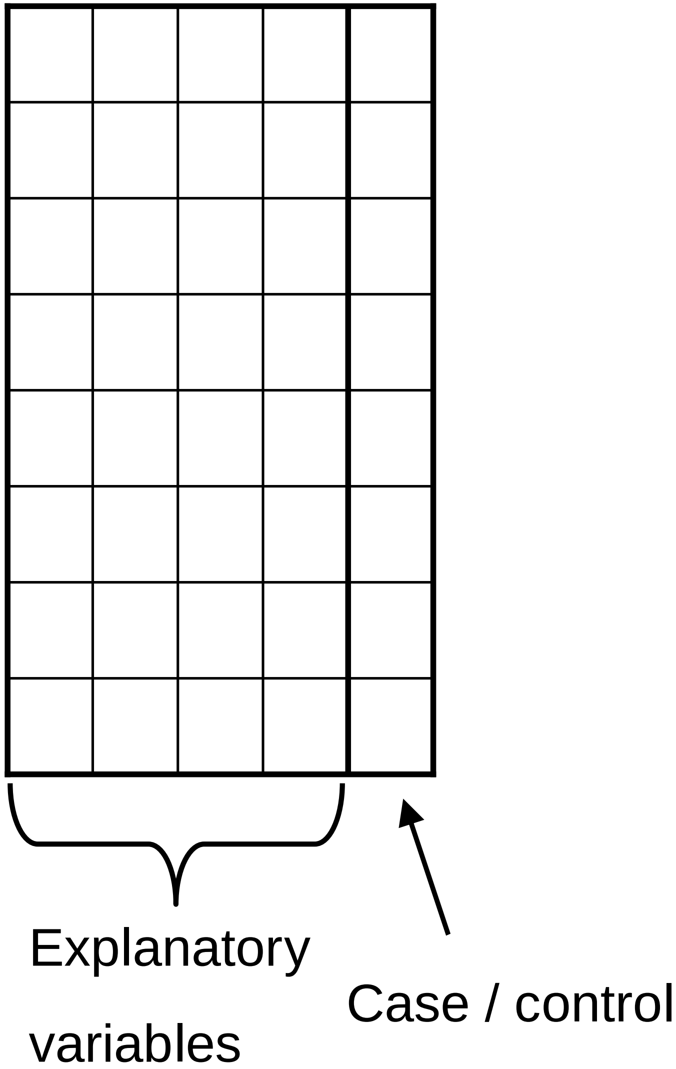
How to randomize?

- How are the datasets D_i generated?
- Randomly from a ”null model” / ”null hypothesis”

Example: randomizing the status variable

- Lots of variables on cases and controls in an epidemiological study
- A: computes the best model for explaining the cases, e.g., a decision tree
- A(D) has a score of some type
- Null model: the cases have nothing to do with the result
- Generate pseudocases and pseudocontrols by randomizing the status variable
- See if the score of A(D) is better than for the pseudocontrols

Example: randomizing the status variable



Find a rule for case / control status and compute its quality

Create pseudocases/pseudocontrols by randomizing the last column

Find a rule for pseudocases/pseudocontrols and compute its quality

Iterate

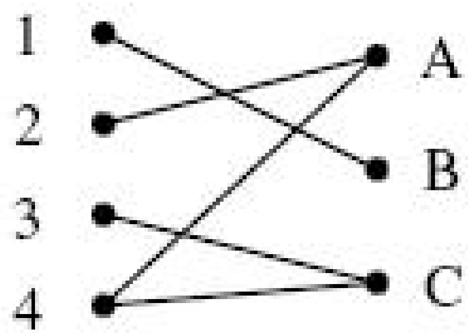
If many of the rules for pseudocase/ control are as good as the rule for the true case / control variable, then the real rule is not significant

0-1 data

- Market basket data: customers and products
- Documents and words
- Regions and species (presence/absence)
- Fossil sites and genera
- ...
- Bipartite graph: observations and variables

0-1 data

$$\begin{array}{c|ccc|} & A & B & C \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \\ 4 & 1 & 0 & 1 \\ \hline \end{array}$$



How to randomize 0-1 data?

- Simple method: randomize each variable (column) independently
- Null model: the variables are independent
- E.g., for testing co-occurrence this yields tests equivalent to traditional tests

- Sometimes this is not the best possible test

Example

X	Y
1	1
1	1
1	1
1	1
1	1
1	1
1	0
1	0
0	1
0	1
0	0
0	0
...	...

Strong correlation between X and Y
Randomization shows that this is not likely to arise by chance

For testing the correlation or co-occurrence counts of the data of X and Y only the columns of X and Y have an impact

Two examples

X	Y
1	1
1	1
1	1
1	1
1	1
1	1
1	0
1	0
0	1
0	1
0	0
0	0

X	Y
1	1
1	1
1	1
1	1
1	1
1	1
1	0
1	0
0	1
0	1
0	0
0	0

Two examples

X	Y	$\dots$
1	1	0 0 1 0 0 1 1
1	1	1 1 0 0 1 0 0
1	1	0 0 0 1 0 1 1
1	1	0 1 1 0 1 0 1
1	1	0 1 0 0 0 0 1
1	1	1 0 1 0 0 1 0
1	0	0 0 0 1 1 0 0
1	0	0 1 1 0 0 0 1
0	1	0 0 1 1 0 0 0
0	1	1 0 0 1 0 0 1
0	0	0 0 1 0 1 0 0
0	0	0 1 1 0 1 0 0

X	Y	$\dots$
1	1	1 1 1 1 1 1 1
1	1	1 1 1 1 1 1 1
1	1	0 1 1 1 1 1 1
1	1	1 1 1 1 1 1 1
1	1	1 1 1 1 0 1 1
1	1	1 1 1 1 1 1 1
1	0	
1	0	
0	1	
0	1	
0	0	
0	0	

X and Y are correlated just because both tend to be 1 in the dense rows

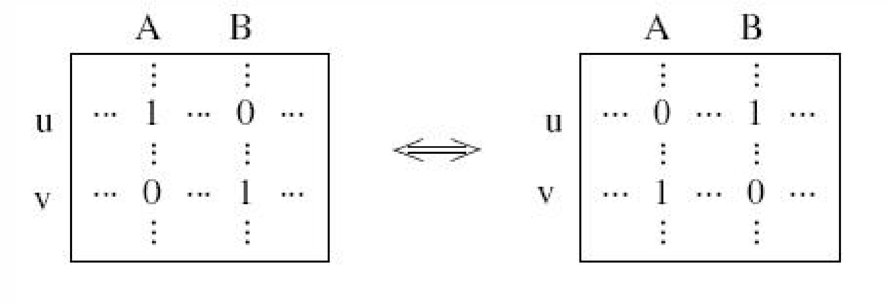
Swap randomization

- 0-1 data: *n* rows, *m* columns, presence/absence
- Randomize by generating random datasets with the same row and column margins as the original data

- Originally from ecology
- Assessing data mining results using swap randomization, Aris Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas, ACM KDD 2006, ACM TKDD, to appear.

Basic idea

- Maintains the degree structure of the data
- Such datasets can be generated by **swaps**



- Simple algoritmic results and experiments è
- The method can be used for moderately large datasets
- Empirical results on significance of data mining results

Fixed margins

- Null hypothesis: the row and column margins of the data are fixed
- What structure is there in the data?
- ... When we take the marginals for granted?

X	Y	...
1	1	0 0 1 0 0 1 1
1	1	1 1 0 0 1 0 0
1	1	0 0 0 1 0 1 1
1	1	0 1 1 0 1 0 1
1	1	0 1 0 0 0 0 1
1	1	1 0 1 0 0 1 0
1	0	0 0 0 1 1 0 0
1	0	0 1 1 0 0 0 1
0	1	0 0 1 1 0 0 0
0	1	1 0 0 1 0 0 1
0	0	0 0 1 0 1 0 0

Significant co-occurrence of X and Y

X	Y	...
1	1	1 1 1 1 1 1 1
1	1	1 1 1 1 1 1 1
1	1	0 1 1 1 1 1 1
1	1	1 1 1 1 1 1 1
1	1	1 1 1 1 0 1 1
1	1	1 1 1 1 1 1 1
1	0	0 0 0 1 1 0 0
1	0	0 1 1 0 0 0 1
0	1	0 0 1 1 0 0 0
0	1	1 0 0 1 0 0 1
0	0	0 0 1 0 1 0 0

No significant co-occurrence of X and Y

Why keep margins fixed?

- The marginal information is known; what else is there?
- Explaining the data by telling first the simple things
- Power-law distributions for degrees etc.: they will have an effect

Problem

- Given a 0-1 dataset D
- Generate random datasets having the same row and column margins as D
- Related to generating contingency tables with given margings & computing permanents

Questions

- D_1, D_2, ...,D_k, where each D_i has the same margins as D

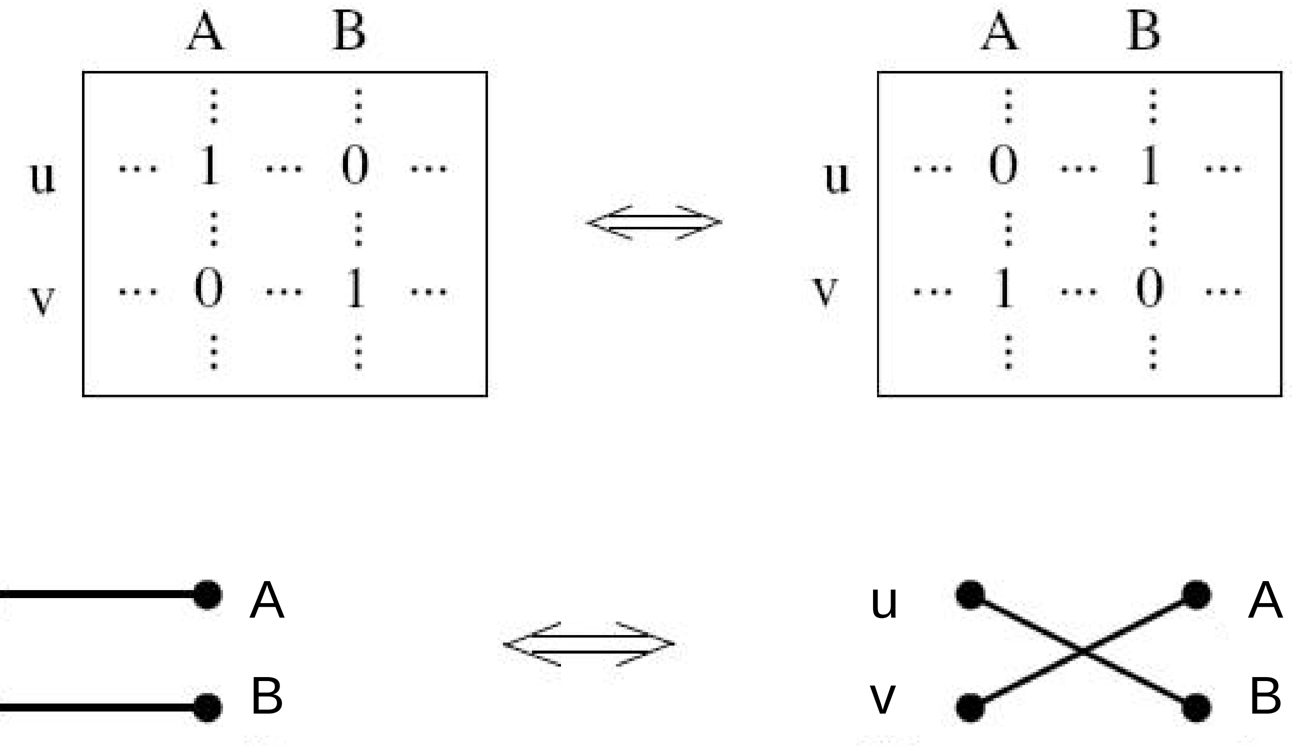
- Can these be computed efficiently?

- $\mathcal{A}(D_1), \mathcal{A}(D_2), \ldots, \mathcal{A}(D_k)$

- Can these be computed efficiently?

How to generate 0-1 datasets with the given margins?

- Swaps:



Swaps: Ryser’s result (1957)

- Every two datasets with the same margins can be converted to each other by a set of swaps

Randomization (again)

- Create randomized versions from the data D

  $D_1, D_2, \ldots, D_k$

- Run the algorithm $\boldsymbol{A}$ on these, producing results $X_1 = \boldsymbol{A}(D_1), X_2 = \boldsymbol{A}(D_2), \ldots, X_k = \boldsymbol{A}(D_k)$

- Compute the empirical p-value: the fraction of cases for which the result on real data is (say) larger than $\boldsymbol{A}(D)$

Generating datasets with fixed margins

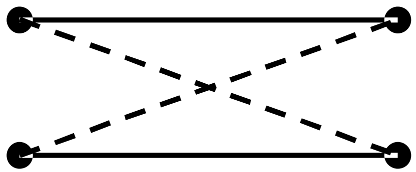
- MCMC approach
- Start from the data $D_0 = D$
- Apply random swaps repeatedly

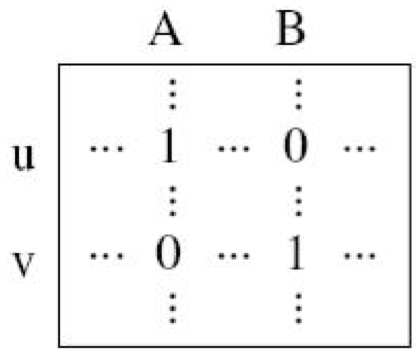
$$D_{i+1} = rswap(D_i)$$

- Do enough swaps
- Then D_i will be a random dataset with the same margins as D
- MCMC state space: all datasets with the same margins as the original data

First attempt

- Start from the data D
- Pick a random **swappable pair** of edges and swap them



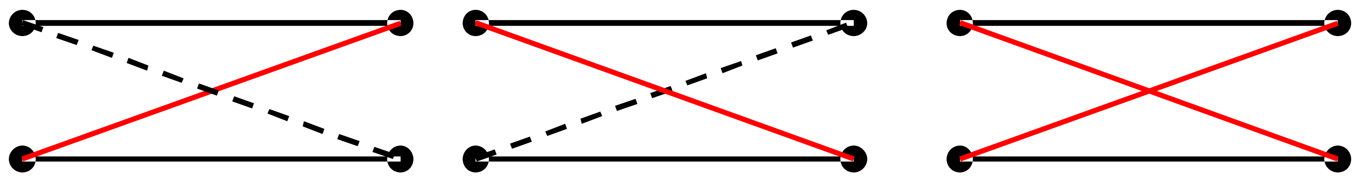


- Random walk on the state space
- Does not produce a uniform distribution
- The state space has nodes with different degrees (datasets with the given margins but different number of possible swaps)

Second method: self-loop

- Make the nodes of the state space to have equal degree
- Introduce self-loops

- Select random pair of edges
  - If not swappable, stay in the current node (but report this as an element of the chain)



Third method: Metropolis-Hastings

- Achieve uniform sampling by accepting swaps according to the degrees of the states

- At each step compute the number S(D_i) of neighbors in the state space of the current dataset

- I.e., the number of datasets with the same margins as the original dataset and one swap away from the current one

Metropolis-Hastings

- Let *D* be the current dataset and let *D’* be one obtained by selecting a random swappable pair of edges

- Accept the move from *D* to *D’* with probability

$$\min(1, \frac{S(D)}{S(D')})$$

What can be said about the algorithms?

- Mixing time is an open problem
- Self-loop: how much time does it use in a state before moving out?
- M-H: how difficult is it to to compute *S(D)*?

Self-loop

- **Lemma**: View the dataset *D* as a graph *(V,E)*. For the class of graphs where the maximum degree is $o(|E|)$, the time spent in each state is constant.

- (Unswappable pairs are relatively rare.)

Idea of analysis of `Self_loop`

- Consider state $G = (U, V, E)$ of $\mathcal{M} = (\mathcal{S}, \mathcal{T})$
- Number of swappable pairs $d(G)$
- Expected time staying in G is $\frac{|E|^2}{d(G)}$
- Probability of ending up in G is $\frac{d(G)}{2|\mathcal{T}|}$
- Thus, expected time staying in a state is

$$T = \sum_{G \in \mathcal{S}} \frac{|E|^2}{d(G)} \cdot \frac{d(G)}{2|\mathcal{T}|} = \frac{|E|^2}{2} \cdot \frac{|\mathcal{S}|}{|\mathcal{T}|}.$$

- Proving that $|\mathcal{T}|/|\mathcal{S}| = \Omega(|E|^2)$ for certain families of graphs gives expected time constant

Self-loop

- **Corollary**. If the degree sequence follows a power law with exponent > 2, then the expected time spent in a state is constant.

Metropolis-Hastings

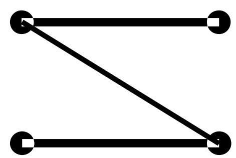
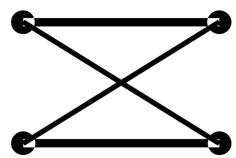
- How difficult is it to compute *S(D)*, the number of possible swaps that can be done on data *D*?

- Can this number be computed incrementally?

- If we know *S(D)* and *D'=swap(D)*, can *S(D')* be computed efficiently?

Computing S(D)

- Lemma:

- *S(D)* =  -  + 2 

- Number of swappable pairs is
  #pairs of edges - #Z-structures + 2 K(2,2)

Computing S(D’) from S(D)

- *D’* is obtained from *D* by one swap

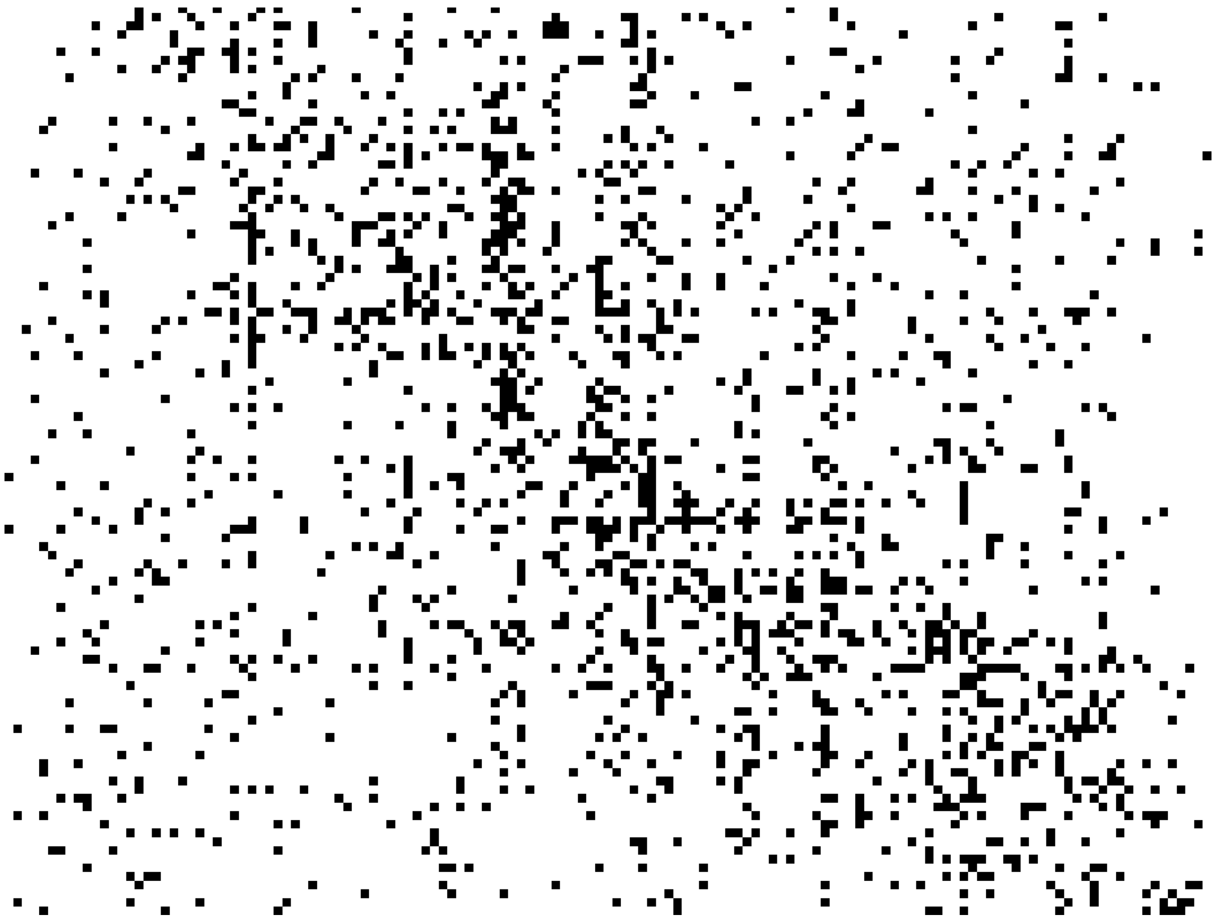
- *S(D’)*: the number of pairs of edges does not change
- Difference in the number of Zs can be computed in constant time
- Difference in the number of *K(2,2)*s can be computed in linear time (no. of columns)
- Corollary: M-H can be implemented efficiently

Comparison

- Self-loop seems to be as good as Metropolis-Hastings, or better

Example

1000 attempted swaps (self-loop)



10000 attempts

100000 attempts

Empirical results: datasets

Dataset	# of rows	# of cols	# of 1's	dens. (%)
Abstracts	128820	25335	10449902	0.32
Abstracts′	128803	5918	7150992	0.94
Courses	2405	5021	65152	0.54
Kosarak	990002	41270	8019015	0.02
Paleo	124	139	1978	11.48
Retail	88162	16470	908576	0.06

Sparse data!

Generating the different datasets

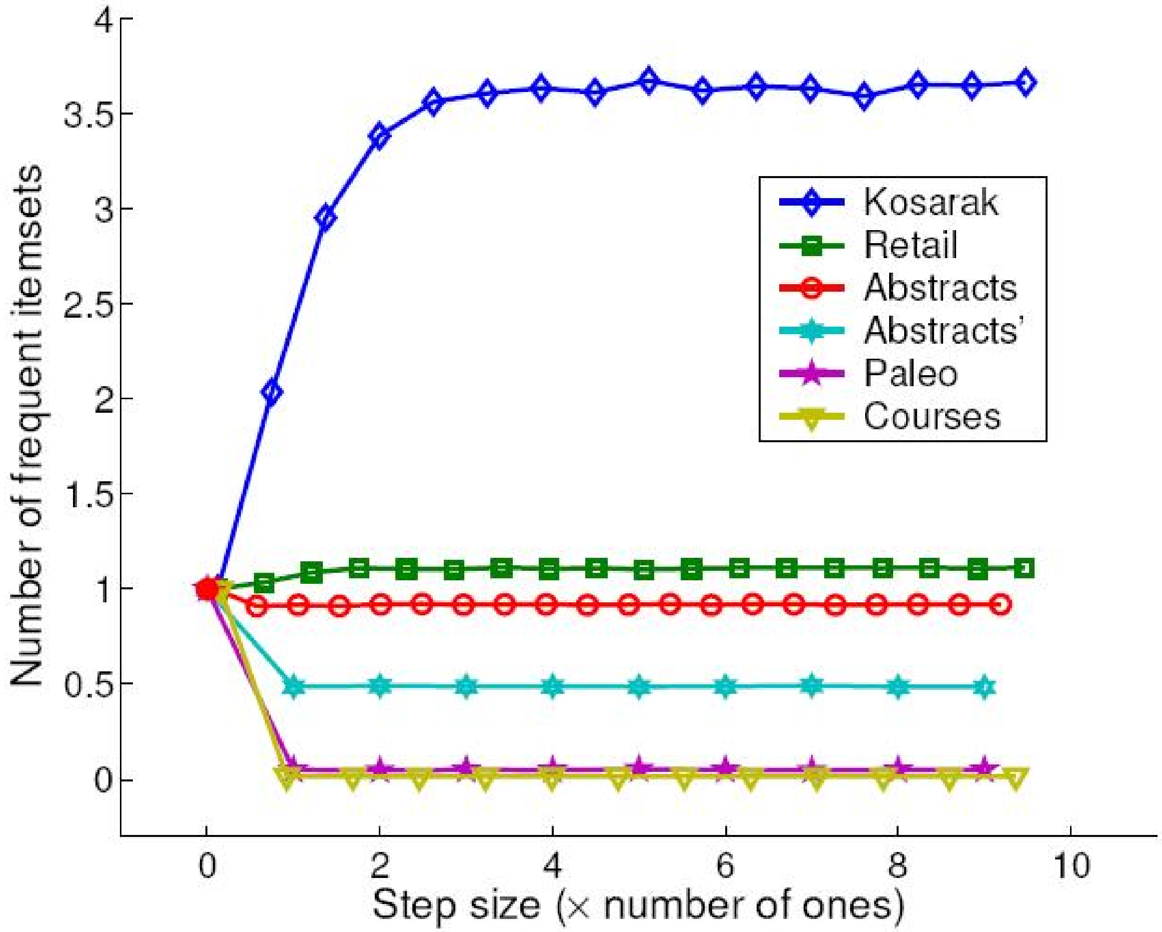
- Start from D
- Run k swaps, obtain D'
- Do N times
  - Start from D'
  - Run k swaps: result D_i

- Why this way?
- D is at the same situation as the D_i 's

Convergence

- Hard to say anything on this theoretically
- Compute different types of aggregate statistics and see when they stabilize
- E.g, the number of frequent itemsets or clustering error
- Observation: about 2 x the number of 1s in the dataset seems to be sufficient for stabilization

Convergence



Running time for the swaps

Clock time to perform ($5 \times$ the number of 1's) swaps

Dataset	time
Abstracts	12m 53s
Abstracts′	9m 11s
Courses	3.3s
Kosarak	8m 38s
Paleo	0.100s
Retail	1m 1.5s

– Perl implementation, 3GHz Pentium, 2GB memory

Clustering

Dataset	k	E	mean	std	Z	p
S1	10	1777.3	3669.9	11.1	170.43	0.01
S2	10	4075.4	4084.4	11.6	0.77	0.22
COURSES	10	17541.6	24405.1	30.2	227.09	0.01
PALEO	10	1040.7	1401.7	4.8	74.74	0.01
RETAIL	10	23920.9	24086.0	135.2	1.22	0.10

Error in clustering
Of the real data

Mean error in clustering
of the swapped data

Frequent sets

Dataset	$\lvert X \rvert$	$\lvert \mathcal{F} \rvert$	$\lvert \mathcal{F}_s \rvert$
ABSTRACTS, $(\sigma = 5000)$	≥ 2	1128	1004.8(4.8)
	≥ 3	226	188.7(2.5)
ABSTRACTS$'$, $(\sigma = 600)$	≥ 2	4854	839.5(19.2)
	≥ 3	223	0.0(0.0)
COURSES, $(\sigma = 400)$	≥ 2	9687	442.2(12.5)
	≥ 3	9412	259.7(11.4)
KOSARAK, $(\sigma = 5000)$	≥ 2	1436	5644.5(60.8)
	≥ 3	977	5013.8(59.5)
PALEO, $(\sigma = 7)$	≥ 2	2828	266.7(14.8)
	≥ 3	2058	9.8(5.4)
RETAIL, $(\sigma = 200)$	≥ 2	1384	1616.1(12.3)
	≥ 3	489	569.0(9.1)