Topic III.1: Swap Randomization

Discrete Topics in Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2012/13

Topic III.1: Swap Randomization

Motivation & Basic Idea

- **Permutation test** for assessing the significance of a data mining result
  - Is this itemset significant?
  - Are all itemsets that are frequent w.r.t. threshold *t* significant?
  - Is this clustering significant?
- Null hypothesis: *The results are explained by the number of 1s in the rows and columns of the data*
  - We expect binary data for now
  - Previous lecture: only number of 1s per column was fixed

Basic Setup

- Let $\boldsymbol{D}$ be n-by-m data matrix and let $\boldsymbol{r}$ and $\boldsymbol{c}$ be its row and column margins
- Let $M(\boldsymbol{r}, \boldsymbol{c})$ be the set of all n-by-m binary matrices with row and column margins defined by $\boldsymbol{r}$ and $\boldsymbol{c}$
  - Let $S \subseteq M(\boldsymbol{r}, \boldsymbol{c})$ be a *uniform* random sample of $M(\boldsymbol{r}, \boldsymbol{c})$
- Let $R(\boldsymbol{D})$ be a single number that our data mining method outputs
  - E.g. the number of frequent itemsets w.r.t. t, the frequency of an itemset I, the clustering error
- The **empirical *p*-value** for $R(\boldsymbol{D})$ being big is

$$(|\{\boldsymbol{D}' \in S : R(\boldsymbol{D}') \geq R(\boldsymbol{D})\}| + 1) \,/\, (|S| + 1)$$

Comments on Empirical *p*-value

- The **empirical *p*-value** for $R(\boldsymbol{D})$ being big is

$$(|\{\boldsymbol{D}' \in S : R(\boldsymbol{D}') \geq R(\boldsymbol{D})\}| + 1) / (|S| + 1)$$

- The +1's are to avoid having problems with 0s
- If $S = M(\boldsymbol{r}, \boldsymbol{c})$ this is an exact test
  - +1's are not needed
- The bigger the sample, the better
  - Sample size also controls the maximum accuracy
- Changing the definition for small $R(\boldsymbol{D})$ or two-tailed test is easy

Swaps

$$\begin{array}{c|ccccc|} & & A & & B & \\ \hline & & \vdots & & \vdots & \\ u & \cdots & 1 & \cdots & 0 & \cdots \\ & & \vdots & & \vdots & \\ v & \cdots & 0 & \cdots & 1 & \cdots \\ & & \vdots & & \vdots & \\ \hline \end{array} \quad \Longleftrightarrow \quad \begin{array}{c|ccccc|} & & A & & B & \\ \hline & & \vdots & & \vdots & \\ u & \cdots & 0 & \cdots & 1 & \cdots \\ & & \vdots & & \vdots & \\ v & \cdots & 1 & \cdots & 0 & \cdots \\ & & \vdots & & \vdots & \\ \hline \end{array}$$

- A **swap box** of ***D*** is a 2-by-2 combinatorial sub-matrix that is either *diagonal* or *anti-diagonal*
- A **swap** turns diagonal swap box into anti-diagonal, or vice versa
- **Theorem** [Ryser '57]. If ***A***, ***B*** ∈ *M*(***r***, ***c***), then ***A*** is reachable from ***B*** with a finite number of swaps

Generating Random Samples

- **Idea:** Starting from the original matrix, perform k swaps to obtain a random sample from $M(\boldsymbol{r}, \boldsymbol{c})$, and run the data mining algorithm with this data. Repeat.
  - The empirical p-value can be computed from the results
  - Simple
  - Requires running the data mining algorithm multiple times
    - Can be very time consuming with big data sets
- **Question:** Are we sure we get a uniform sample from $M(\boldsymbol{r}, \boldsymbol{c})$?
  - The results are not valid if the sample is not uniform
  - To ensure uniformity, we need a bit more theory…

Markov Chains and Sampling

- A **stochastic process** is a family of random variables $\{X_t : t \in T\}$
  - Henceforth $T = \{0, 1, 2, \ldots\}$ and t is called *time*
    - This is *discrete stochastic process*
- Stochastic process $\{X_t\}$ is **Markov chain** if always

$$\Pr[X_t = x \mid X_{t-1} = a, X_{t-2} = b, \ldots, X_0 = z]$$
$$= \Pr[X_t = x \mid X_{t-1} = a]$$

  - Memory-less property
- A Markov chain is **time-homogenous** if for all t

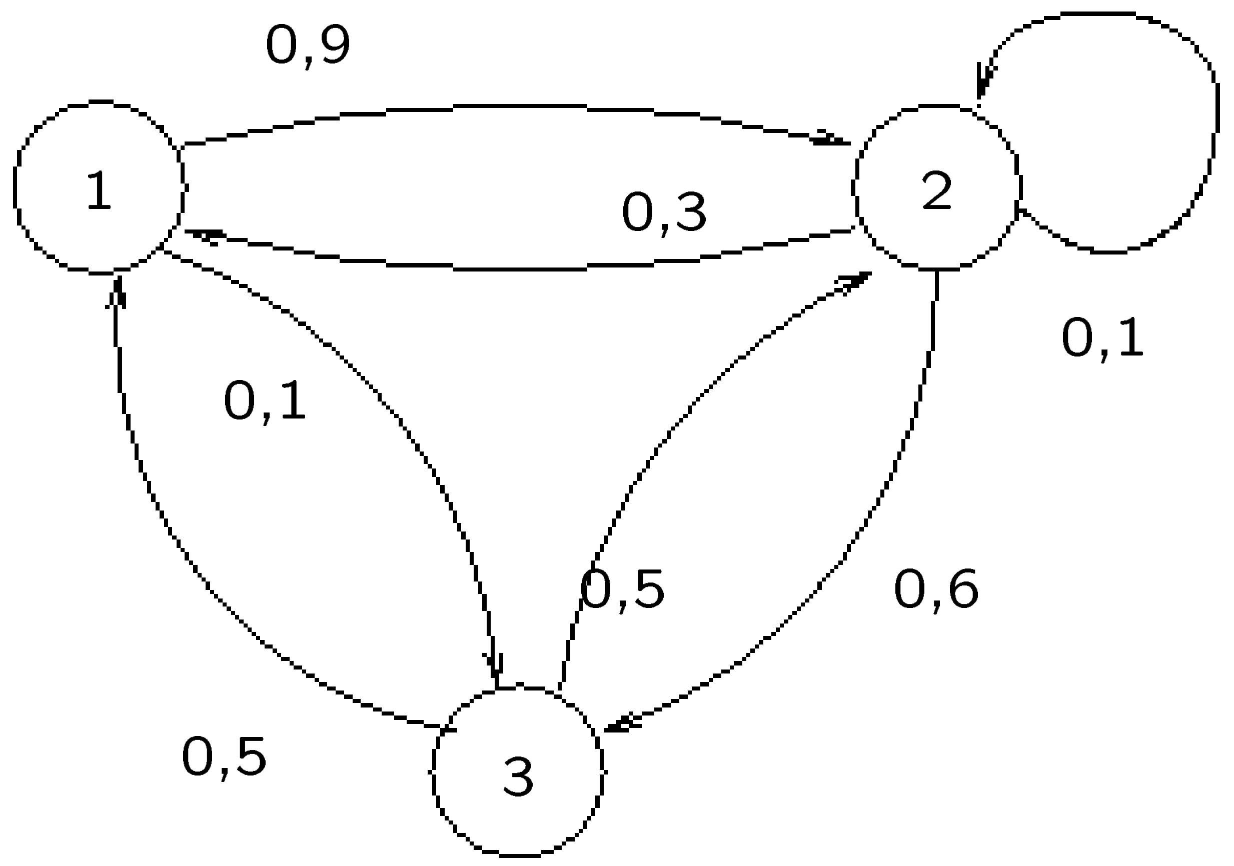
$$\Pr[X_{t+1} = x \mid X_t = y\,] = \Pr[X_t = x \mid X_{t-1} = y]$$

  - We only consider time-homogenous Markov chains

Transition matrix

- The **state space** of a Markov chain $\{X_t\}_{t \in T}$ is the countable set S of all values X_t can assume
  - $X_t \colon \Omega \to S$ for all $t \in T$
  - Markov chain is in state s at time t if $X_t = s$
  - A Markov chain $\{X_t\}_{t \in T}$ is *finite* if it has finite state space
- If Markov chain $\{X_t\}$ is finite and time-homogenous, its **transition probabilities** can be expressed with a matrix $\mathbf{P} = (p_{ij})$, $p_{ij} = \Pr[X_1 = j \mid X_0 = i]$
  - Matrix $\mathbf{P}$ is n-by-n if Markov chain has n states and it is *right stochastic*, i.e. $\sum_j p_{ij} = 1$ for all i (rows sum to 1)

Example Markov chain



$$P = \begin{pmatrix} 0 & 9/10 & 1/10 \\ 3/10 & 1/10 & 6/10 \\ 1/2 & 1/2 & 0 \end{pmatrix}$$

Classifying the states

- State i can be *reached* from state j if there exists $n \geq 0$ such that $(\mathbf{P}^n)_{ij} > 0$
  - $\mathbf{P}^n$ is the nth exponent of $\mathbf{P}$, $\mathbf{P}^n = \mathbf{P}\times\mathbf{P}\times\cdots\times\mathbf{P}$
- If i can be reached from j and vice versa, i and j *communicate*
  - If all states $i, j \in S$ communicate, Markov chain is **irreducible**
- If the probability that the process visits a state i infinitely many times is 1, then state i is **recurrent**
  - State is **positive recurrent** if the estimated return time to it is finite
  - Markov chain is recurrent if all of its states are

More classifying of the states

- State i has **period** k if any return to i must occur in time that is multiple of k:

  $$k = \gcd\{n : \Pr[X_n = i \mid X_0 = i] > 0\}$$

  - State i is **aperiodic** if it has period $k = 1$; otherwise it is **periodic** with period k
  - Markov chain is aperiodic if all of its states are
- State i is **ergodic** if it is aperiodic and positive recurrent
  - Markov chain is ergodic if all of its states are

Two important results for finite MCs

Lemma. Every finite Markov chain has at least one recurrent state and all of its recurrent states are positive recurrent.

Corollary. Finite, irreducible, and aperiodic Markov chain is ergodic.

Stationary distributions

- If $\boldsymbol{\pi}$ is such that $\pi_i \geq 0$ for all i, $\sum_i \pi_i = 1$, and
$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$$
then $\boldsymbol{\pi}$ is the **stationary distribution** of the Markov chain
- Let $h_{ii} = \sum_{t\geq 1} t\Pr[X_t = i \text{ and } X_n \neq i \text{ for } n < t \mid X_0 = i]$ be the estimated return time to state i

Theorem. If Markov chain is finite, irreducible, and ergodic, then
1. it has an unique stationary distribution $\boldsymbol{\pi}$
2. for all i and j, $\lim_{t\to\infty} (\mathbf{P}^t)_{ji}$ exists and is the same for all j
3. $\pi_i = \lim_{t\to\infty} (\mathbf{P}^t)_{ji} = 1/h_{ii}$

More on stationary distributions

- If Markov chain has a stationary distribution, then the probability that the chain is in state *i* after long-enough time is independent of the starting time but depends only on the stationary distribution
- Aperiodicity is not necessary condition for stationary distribution to exist, but then the stationary distribution will not be the limit of transition probabilities
  - Two-state chain that always switches the state has stationary distribution (1/2, 1/2), but the transitions look either (1, 2, 1, 2, ...) or (2, 1, 2, 1, ...) depending on the starting state

Markov Chain Monte Carlo Method

- The **Markov Chain Monte Carlo** (MCMC) method is a way to sample from probability distributions
- Each possible sample is a state in a Markov chain
- Each state has a **neighbour structure** giving the transitions in the chain
- The chain is build so that its stationary distribution is the desired distribution to sample from
- After *burn-in* period, the chain is well-mixed, and we can sample by taking every *n*th state

Uniform Stationary Distribution

- **Lemma.** Consider a Markov chain with a finite state space. Let $N(x)$ be the set of neighbours of state x, let $N = \max_x |N(x)|$, and let $M \geq N$. Define the transition probabilities by

$$\mathbf{P}_{xy} = \begin{cases} 1/M & \text{if } x \neq y \text{ and } y \in N(x), \\ 0 & \text{if } x \neq y \text{ and } y \notin N(x), \\ 1 - N(x)/M & \text{if } x = y. \end{cases}$$

If this chain is irreducible and aperiodic, then the stationary distribution is the uniform distribution.

The Metropolis Algorithm

- The **Metropolis algorithm** is a general technique to transform any irreducible Markov chain into a time-reversible chain with a required stationary distribution
  - A Markov chain is *time-reversible* if $\pi_i \boldsymbol{P}_{ij} = \pi_j \boldsymbol{P}_{ji}$
- Let $N(x)$, N, and M be as in previous slide, and let $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ be the desired stationary distribution.
  - Let

$$\mathbf{P}_{xy} = \begin{cases} 1/M \min\{1, \pi_y/\pi_x\} & \text{if } x \neq y \text{ and } y \in N(x), \\ 0 & \text{if } x \neq y \text{ and } y \notin N(x), \\ 1 - \sum_{y \neq x} \mathbf{P}_{xy} & \text{if } x = y. \end{cases}$$

  - If the chain is aperiodic and irreducible, the stationary distribution is the desired one

Notes on the Metropolis Algorithm

- Two-step process: each neighbour is selected with probability $1/M$, and accepted with probability π_y/π_x
  - To obtain uniform distribution, only the first step is needed
- We do not need to have the transition matrix defined explicitly
  - E.g. inifinite state space
  - Even with finite chains, MCMC methods can be faster than solving the stationary distribution first
- Slightly more general method is known as the *Metropolis–Hastings algorithm*

The Metropolis–Hastings Algorithm

- A generalization of the Metropolis algorithm
- Suppose we have a Markov chain with transition matrix $\boldsymbol{Q}$
- We generate a new chain where we move from state x to state y with probability $\min\left\{\frac{\pi_y \mathbf{Q}_{yx}}{\pi_x \mathbf{Q}_{xy}}, 1\right\}$ and otherwise stay still
- This new chain will have the desired stationary distribution

Besag–Clifford Correction

- The subsequent states in Markov chains are dependent
  - Subsequent samples in Metropolis are dependent, too
  - No problem if we have long-enough (mixing time) gaps between samples
    - But mixing time is hard to estimate…
- In Besag–Clifford correction, we first run the chain s steps backward and then from there k times s steps forward
  - The original data and random samples are exchangeable
  - Time-reversible chains: backward = forward

Swap-Randomization for Binary Data

- To obtain the uniform samples from $M(\boldsymbol{r}, \boldsymbol{c})$, we use an MCMC method
  - The states of the chain are the matrices in $M(\boldsymbol{r}, \boldsymbol{c})$
  - The neighbours of $\boldsymbol{X}$ are the matrices $\boldsymbol{Y} \in M(\boldsymbol{r}, \boldsymbol{c})$ that are reachable from $\boldsymbol{X}$ with a single swap
  - But the resulting chain does not have uniform stationary distribution
- To ensure the uniform distribution, we have two options
  - Add multiple self-loops so that each state has the same degree
  - Use the Metropolis–Hastings algorithm

Gionis, Mielikäinen & Mannila 2007

Self-Loops

- In every state $\boldsymbol{X}$, we select u.a.r. two elements (i, j) and (k, l) of the matrix $(i \neq k, j \neq l)$ such that $\boldsymbol{X}_{ij} = \boldsymbol{X}_{kl} = 1$
- If the selected elements are corners of a swap box, we perform the swap
  - Swap box if $\boldsymbol{X}_{il} = \boldsymbol{X}_{kj} = 0$
- Otherwise, we stay at $\boldsymbol{X}$ but consider this a step
- This chain has uniform stationary distribution because each state has equivalent degree
  - Each self-loop is counted separately
- This chain has long burn-in time

Metropolis–Hastings

- Let $N(\boldsymbol{X})$ be the number of neighbours of matrix $\boldsymbol{X}$
- For Metropolis–Hastings, we select $Y \in N(\boldsymbol{X})$ u.a.r. and make the transition with probability $\min\{N(\boldsymbol{X})/N(\boldsymbol{Y}), 1\}$
  - To select $\boldsymbol{Y}$, we use rejection sampling
    - Try random pairs (i, j), (k, l) and return the first that defines a swap box
- Metropolis–Hastings probably converges faster than the self-loop method
  - But it needs to know the size of the neighbourhood

Counting the Neighbours

- **Theorem.** The number of neighbours of $\boldsymbol{X}$ is

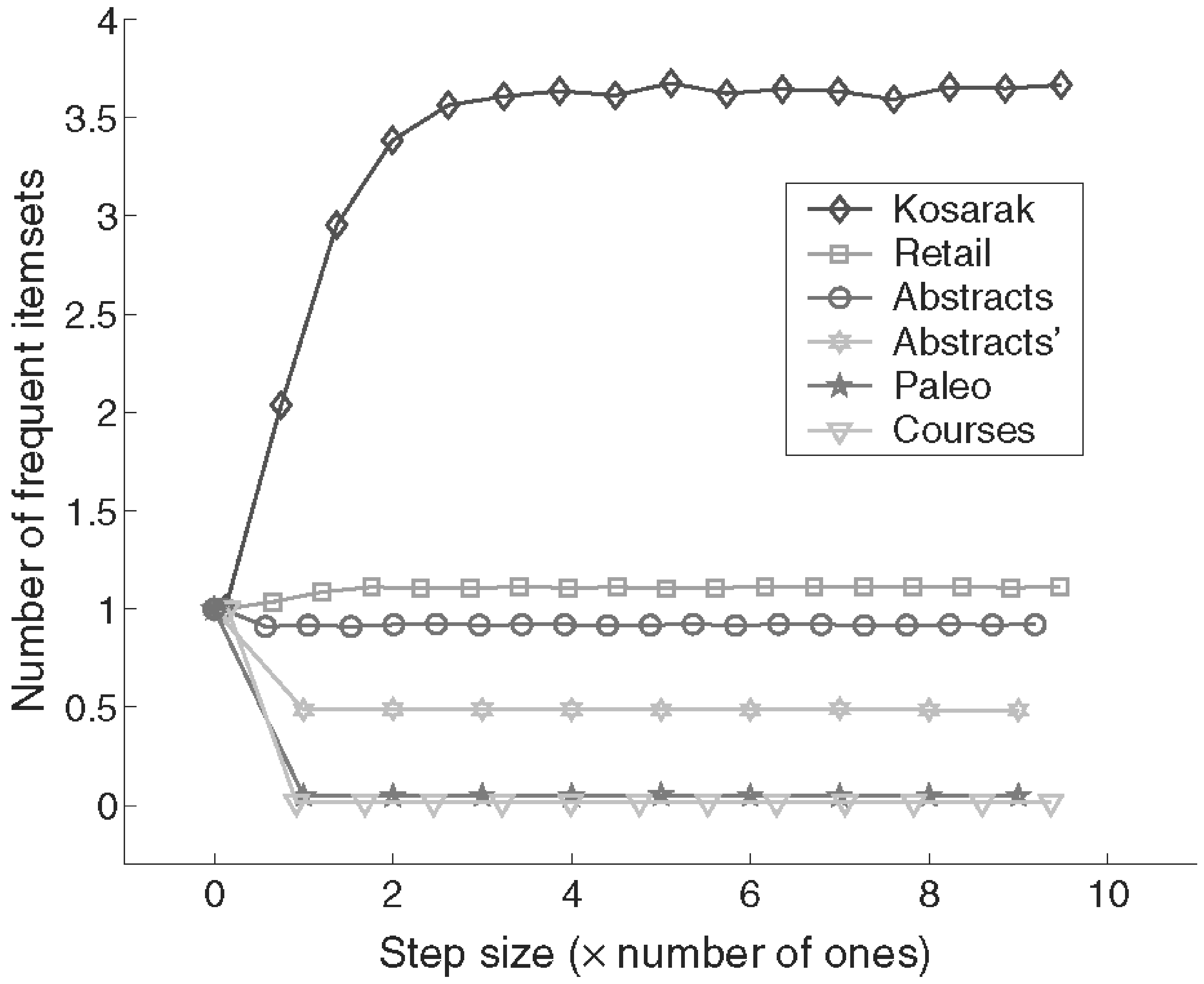
  $$N(\boldsymbol{X}) = J(\boldsymbol{X}) - Z(\boldsymbol{X}) + 2K_{22}(\boldsymbol{X}),$$

  where
  - $J(\boldsymbol{X})$ is the number of pairs (i, j), (k, l) with distinct i, j, k, and l such that $\boldsymbol{X}_{ij} = \boldsymbol{X}_{kl} = 1$
    - All potential swap boxes
  - $Z(\boldsymbol{X})$ is the number of "Z-structures": distinct i, j, k, and l such that $\boldsymbol{X}_{ij} = \boldsymbol{X}_{kl} = \boldsymbol{X}_{kj} = 1$
    - Non-swap boxes
  - $K_{22}(\boldsymbol{X})$ is the number of 2-by-2 all-1s submatrices of $\boldsymbol{X}$
    - $Z(\boldsymbol{X})$ removes some non-swap boxes multiple times

Updating the Neighbour Count

- **Theorem.** If we know $N(\boldsymbol{X})$ and $\boldsymbol{Y}$ is obtained from $\boldsymbol{X}$ with a single swap, then we can compute $N(\boldsymbol{Y})$ by
$$N(\boldsymbol{Y}) = N(\boldsymbol{X}) - \Delta Z + 2\Delta K_{22},$$
where ΔZ is the change in number of Z-structures and ΔK_{22} is the change in number of 2-by-2 all-1s submatrices.
- The change can be computed in time $\min\{n, m\}$
  - Thus, the convergence is probably faster, but each step costs considerably more than with self-loops

Mixing Times for Self-Loop



Gionis, Mielikäinen & Mannila 2007

Numerical Data

- Swap randomization *per se* works only for binary data
- It can be extended to handle real-valued data
- Two different tasks (null hypotheses):
  - Approximately the same value distributions on rows and columns
  - Approximately the same mean and variance on rows and columns
- The algorithms are based on the Metropolis algorithm
  - The neighbourhood is based on different local changes

Ojala et al. 2009

Local Changes

- One-element changes
  - Replace a value
  - Add another value
- Four-element changes
  - Rotate
    - If a = a’ and b = b’, equals to swap
  - Mask
    - Preserves row and column sums

$$\begin{array}{c|ccccc|}
 & & j_1 & & j_2 & \\
\hline
 & & \vdots & & \vdots & \\
i_1 & \dots & a & \dots & b & \dots \\
 & & \vdots & & \vdots & \\
i_2 & \dots & b' & \dots & a' & \dots \\
 & & \vdots & & \vdots & \\
\hline
\end{array}
\Longrightarrow
\begin{array}{c|ccccc|}
 & & j_1 & & j_2 & \\
\hline
 & & \vdots & & \vdots & \\
i_1 & \dots & b' & \dots & a & \dots \\
 & & \vdots & & \vdots & \\
i_2 & \dots & a' & \dots & b & \dots \\
 & & \vdots & & \vdots & \\
\hline
\end{array}$$

Rotate

$$\begin{array}{c|ccccc|}
 & & j_1 & & j_2 & \\
\hline
 & & \vdots & & \vdots & \\
i_1 & \dots & +\alpha & \dots & -\alpha & \dots \\
 & & \vdots & & \vdots & \\
i_2 & \dots & -\alpha & \dots & +\alpha & \dots \\
 & & \vdots & & \vdots & \\
\hline
\end{array}$$

Mask

Ojala et al. 2009

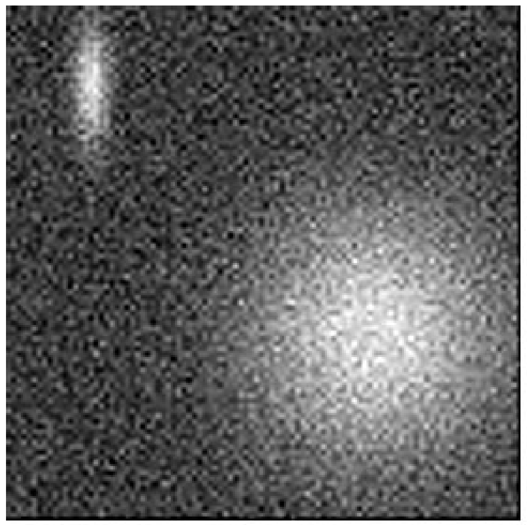
Acceptance Probability

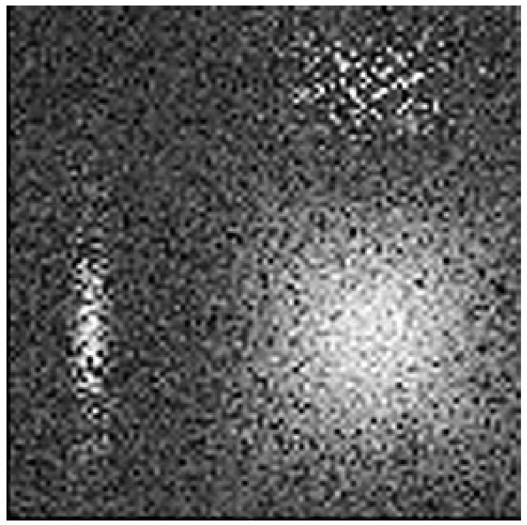
- The Metropolis algorithm performs the local change and accepts the result with a certain probability
- If $\boldsymbol{X}$ is the original matrix, and $\boldsymbol{Y}$ is the result, we accept with probability $c \times \exp\{-wE(\boldsymbol{X}, \boldsymbol{Y})\}$, where
  - c is a normalization constant
  - w is a weight parameter
  - $E(\boldsymbol{X}, \boldsymbol{Y})$ is a distance measure between $\boldsymbol{X}$ and $\boldsymbol{Y}$
    - Depends on the task
    - Further away the result is from the original, the less likely it is to be selected

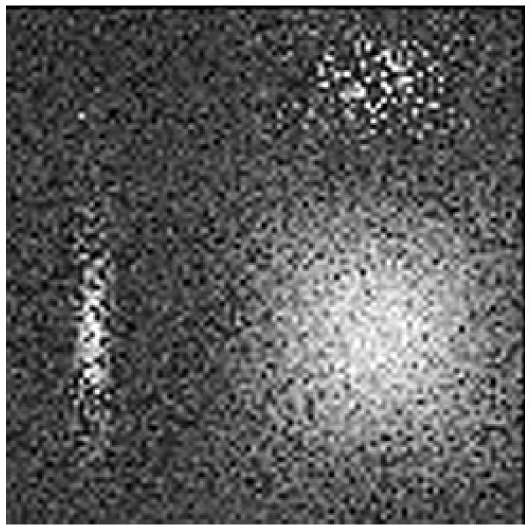
Distance Measures

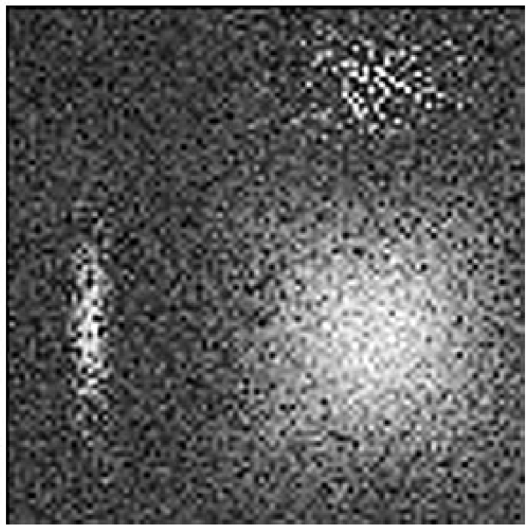
- For having approximately the same value distributions, we need to measure the distance of these distributions
  - L_1 norm between the observed unnormalized cdf's
  - Faster method: compare histograms
- For approximately the same mean and variance, that's what we must measure
  - $|s|(|\mu - \mu'| + |\sigma - \sigma'|)$, where
    - $|s|$ is the number of distinct values
    - μ and μ' are the means of the original and transformed matrix
    - σ and σ' are the standard deviations of the original and transformed matrix

Example


(a) Original


(b) *GeneralMetropolis* with `Resample` and difference measure in distributions


(c) *General Metropolis* with `Mask` and difference measure in means and variances


(d) *SwapDiscretized*

Ojala et al. 2009

Some Notes

- Masking seems to be a good local modification
- Computing the L1 in cdf’s is very slow
  - Approximation using histograms doesn’t hamper the results
- Cannot handle missing values
- Is not good with cases where columns are in different scales
  - E.g. temperature and rainfall; blood pressure and height
  - A method to handle these is presented by Ojala (2010)

Feedback from Topic II Essay

- *Metro Maps of Science* was the most popular choise by far
  - *Applications of Frequent Subgraph Mining* was the other one selected
  - Surprising, as I thought the *MMoS* as the hardest option
- Overall quality keeps on increasing, great work!
  - And also the requirement level increases a bit…
- Once again: if you use figures or tables directly from some other paper, you must cite the source **in the caption** of the said table or figure