Chapter 10:
Data Streams

Jilles Vreeken

IRDM '15/16

15 Dec 2015

UNIVERSITÄT DES SAARLANDES

CLUSTER OF EXCELLENCE

max planck institut informatik

IRDM Chapter 10, overview

- Stream Mining
  1. Basic Ideas
  2. Uniform Sampling
  3. Membership Queries
  4. Counting Distinct Items
  5. Mining Frequent Items
  6. Mining Frequent Itemsets

You'll find this covered in
Aggarwal Ch. 12(.2)

Chapter 10: Streams

Aggarwal Ch. 12

Motivation

In stream processing

- (all) data **cannot be stored –** we can only make one pass
- analysis needs to be **online –** no time to wait for an answer
- time per update is **limited**

Many normally trivial questions become very hard

- how much traffic from/to one IP adress?
- how many distinct flows?
- what are the heavy hitters?

Stream Mining

Abstraction

- a stream $\mathcal{S}$ is a continuous sequence of **items** or **elements**

Notation

- stream $\mathcal{S} = X_1, X_2, \ldots$
  - possibly infinite
- of n observed elements, i.e. from X_1 up till X_n
- of m distinct elements

Stream Mining

Abstraction

- a stream $\mathcal{S}$ is a continuous sequence of **items** or **elements**

Problems

- maintain a uniform sample

- how many distinct items do I have in my stream?

 (6)

- give all frequent items in the stream

$\text{sup}(\cdot)$ or more: 

Approximations

It won't (always) be possible to give an exact answer

- therefore, approximations

Popular: ϵ, δ-approximations

- $P(|X - E[X]| > \epsilon) \leq \delta$
- in $1 - \delta$ of the cases we are at most ϵ off

We will see a few example stream mining algorithms

- uniform sampling
- number of distinct items
- frequent items and itemsets

Chapter 10.2:
Uniform Sampling

Aggarwal Ch. 12.2

Maintaining a uniform sample

Sampling

- stream $\mathcal{S} = X_1, X_2, X_3, X_4, \ldots$
- goal: at any time n, have a uniform sample of size k from $\{X_1, \ldots X_n\}$

Why?

A uniform sample characterises the distribution well[1]

- flexible synopsis of a database
- speeds up processing of analytical queries and data mining tasks
- enhances query optimization
- …

[1] if there is no concept drift

Reservoir Sampling

How can we get a uniform sample R of k elements over a stream $\mathcal{S}$?

- that is, how do we make sure that after n elements of $\mathcal{S}$, each of those have the **same probability** to be in R?

Reservoir Sampling, The Key Idea:

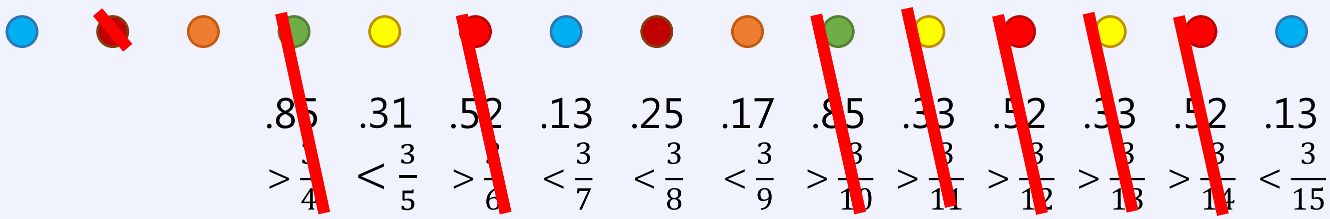
- initialise reservoir R with first k elements of $\mathcal{S}$
- insert nth element into R with probability $\frac{k}{n}$
- if successful, remove one of the k old points uniformly at random

Now, every element of $\mathcal{S}$ has the probability $\frac{k}{n}$ to be in R (!)

(Aggarwal Ch. 2.4.1)

Example: reservoir sampling

For example, for the following stream of data



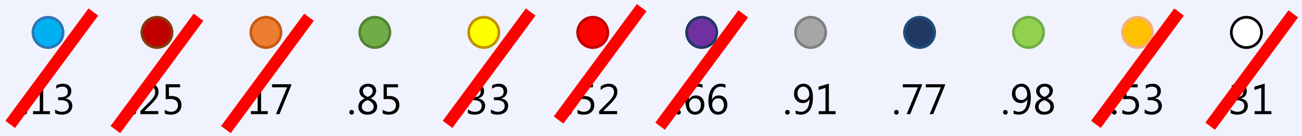
we maintain the following reservoir of size 3

Min-Wise Algorithm

The min-wise algorithm is even simpler

- we maintain a sample R of k elements
- at every time point i draw a random number in $[0,1]$ and only keep the objects of the highest k draws

For example, for $k = 4$



- at any time, every point has the **same chance** to be in the top-k

Concept is simpler than reservoir sampling, but (slightly) more costly

Concept Drift

The process generating a stream is seldom stationary

- when the distribution of the stream changes, we call this concept drift
- uniform sample may be **stale**

To have a relevant sample, we need a recency bias

- a bias function gives higher sample probabilities to recent elements
- most commonly, we use an exponential bias function

$$f(r,n) = e^{-\lambda(n-r)}$$

Synopses for Massive-Domains

In certain settings, not just the **number of data points** is a problem, but also the **size of the domain**

Storing even simple summary statistics, such as

- set membership determination,
- distinct element counts,
- (frequent) item counts,

become challenging w.r.t. space constraints.

For example, we often deal with **pairs** of identifiers

- e.g. such as *from* and *to* email or ip-addresses.
- for a 10^8 unique addresses, there are 10^{16} unique pairs (!)

Chapter 10.3:
Membership Queries

Aggarwal Ch. 12.2.2

Bloom filters

Given an element ●, has it ever occurred in the stream?

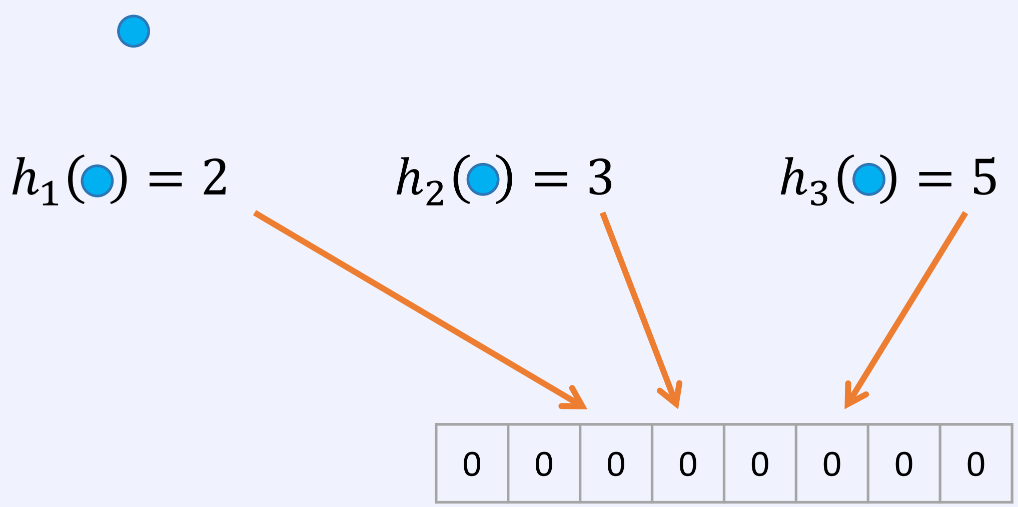
- no false negatives, probabilistic guarantee on false positives
- using only $O(k)$ space

A **bloom filter** is an array B of k bits,
together with w indepdent hash functions,
each of which of type $h : U \to \{0,1,2, \dots, k-1\}$

- initialise B to all 0's
- item ● enters at time t
  - **for** $j = 1$ to w **do**
    - update $h_j(●)^{th}$ element of B to 1
- when membership of element ● is queried
  - return 1 if all $h_j(●)^{th}$ elements of B are set to 1 for all $j = 1$ to w

Example: Bloom filters

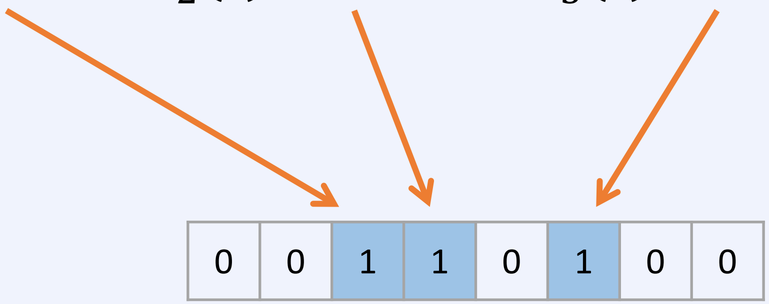
Suppose a bloom filter B
of $k = 8$ bits and 3 hash functions
and the following stream of elements



Example: Bloom filters

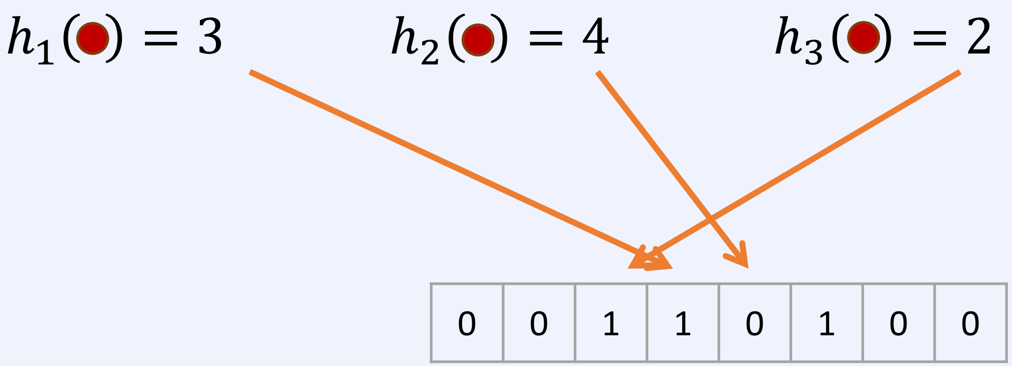
Suppose a bloom filter B
of $k = 8$ bits and 3 hash functions
and the following stream of elements

$h_1(\bullet) = 2$ $h_2(\bullet) = 3$ $h_3(\bullet) = 5$



Example: Bloom filters

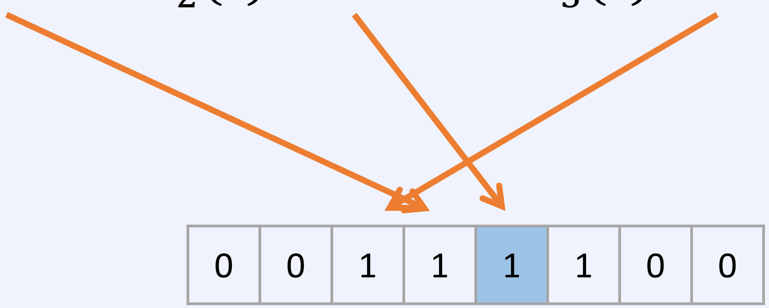
Suppose a bloom filter B
of $k = 8$ bits and 3 hash functions
and the following stream of elements



Example: Bloom filters

Suppose a bloom filter B
of $k = 8$ bits and 2 hash functions
and the following stream of elements

$h_1(\bullet) = 3$ $h_2(\bullet) = 4$ $h_3(\bullet) = 2$

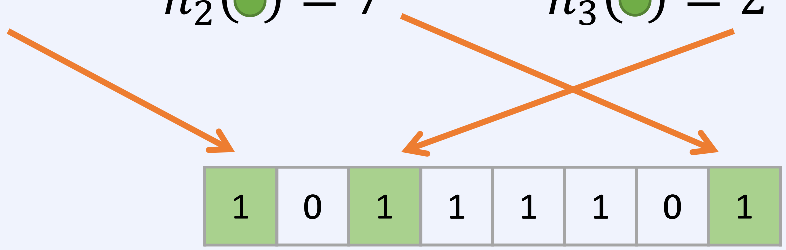


Example: Bloom filters

Suppose a bloom filter B
of $k = 8$ bits and 3 hash functions
after the following stream of elements

Now, for membership query of element ●

$h_1(●) = 0$ $h_2(●) = 7$ $h_3(●) = 2$



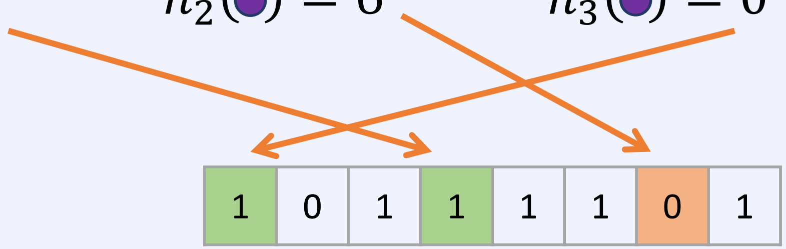
All $h_j(●)$ in B are 1, so answer is yes

Example: Bloom filters

Suppose a bloom filter B
of $k = 8$ bits and 3 hash functions
after the following stream of elements

While for membership query for element ●

$h_1(●) = 3$ $h_2(●) = 6$ $h_3(●) = 0$



Not all $h_j(●)$ in B are 1, so answer is **no**

Bloom filters, analysis

An **upper bound** for the probability of giving a **false positive answer** is related to number of bits k of the filter and number w of hash functions

$$P = \left[1 - \left(1 - \frac{1}{k}\right)^{wn}\right]^{w}$$

For very few or many hash functions performance deteriorates.
Optimum is at $w = k \cdot \ln(2)/n$. We can rewrite to

$$P = 2^{-k \cdot \ln(2)/n}$$

For which k/n is most important. This means the length of the bloom filter should be proportional to the number of distinct elements in $\mathcal{S}$.

Chapter 10.3:
Counting Distinct Items

Aggarwal Ch. 12.2.2.2

The number of distinct items

How to estimate the number of distinct items m, if there are too many of them to keep in memory?

Naive solution

- store all elements
- requires $O(m)$ space for m distinct elements

Can we do better using approximations?

- what can we do with only $O(\log n)$ space?
  - n is an upper bound for m

The number of distinct items

How to estimate the number of distinct items m,
if there are too many of them to keep in memory?

Observation:

If $h(\cdot)$ is a hash function: every $X_i \rightarrow [0,1]$ (u.a.r if $h(\cdot)$ does its job well)
then by maintaining $\min\{h(X_1), h(X_2), \ldots, h(X_n)\}$, we have

$$E[\min\{h(X_1), h(X_2), \ldots, h(X_n)\}] = 1/(1+m)$$

In other words:

the minimal hash gives an estimate of the number of distinct items!

This is called the **min-hash algorithm**. To decrease its variance, we average over (many) (independent) hash functions.

(Flajolet-Martin'85, Alon-Matias-Szegedy'96)

Example: min-hash

For example, for the following stream of data

.13 .25 .17 .85 .33 .52 .13 .25 .17 .85 .33 .52 .33 .52 .13

we get the above stream of hash values

The minimum observed hash value, $\min h(X_i) = h(\bullet) = .13$

by which we can estimate m: $\frac{1}{1+m} = 0.13,$ $m = \frac{1}{0.13} - 1$

Averaging over independent trials makes the result more accurate

Distinct elements – even less space

$$x = 0.\underbrace{000000}_{zeroes(x)}1100101$$

We can store $minHash$ approximately

- store the minimal count of **trailing zeroes**
  - needs only $O(\log \log n)$ bits in worst case
    - $\log \log n$ is an upper bound on m

Algorithm DISTINCT

- initialisation
  - $minZ = 0$
  - hash function $h : U \to [0,1]$
- item ● enters at time t
  - **if** $h(●) < 1/2^{minZ}$, **then** $minZ = zeroes(h(●))$
- when the distinct element count is needed, return 2^{minZ}

(Flajolet-Martin'85, Alon-Matias-Szegedy'96)

Chapter 10.4:
Mining Frequent Items

Aggarwal Ch. 12

Identifying frequent items

Counting every item is impossible

- e.g. all pairs of people that phone each other

Beforehand we do not know the frequent combinations

Example:

30 items: (8) (6) (5)
all others have support 3

For min-freq σ =20%, and need to be reported

(here σ is a **minimal frequency threshold**, as absolute support is useless in an infinite stream)

Superset of the frequent items

We consider an algorithm that finds a **superset** of the σ-frequent items:

- initialisation: no item has a counter
- item 🔴 enters at time t
  - **if** 🔴 has a counter, then *counter*(🔴) + +
  - **else**
    - *counter*(🔴) = 1
    - *start*(🔴) = t
  - **for all** other counters 🟢 **do**
    - **if** $\frac{counter(🟢)}{t - start(🟢) + 1} < \sigma$ **then**
      - delete *counter*(🟢), *start*(🟢)
- when the frequent items are needed, return all items with a counter

(here σ is a **minimal frequency threshold**, as absolute support is useless in an infinite stream)

Example – frequent items in a stream

$\sigma = 20\%$

	start	#	(freq)
	1	1	(100%)

Example – frequent items in a stream

$\sigma = 20\%$

	start	**#**	**(freq)**
	1	1	(100%) (50%)
	2	1	(100%)

Example – frequent items in a stream

$\sigma = 20\%$

	start	#	(freq)
	1	1	(50%) (20%)
	2	1	(100%) (25%) (50%)
	3	2	(66%)
	4	1	(50%)

Example – frequent items in a stream

$\sigma = 20\%$

	start	#	(freq)
	~~1~~	~~1~~	~~(20%) (17%)~~
	2	1	(25%) (20%)
	3	2	(66%) (50%)
	4	1	(50%) (33%)
	6	1	(100%)

Example – frequent items in a stream

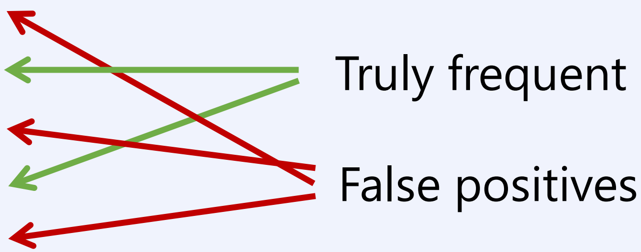
$\sigma = 20\%$

	start	#	(freq)
	2	2	(25%)
	3	2	(29%)
	6	1	(25%)
	8	2	(100%)

Example – frequent items in a stream

$\sigma = 20\%$

	start	#	(freq)
●	16	3	(20%)
●	17	4	(29%)
●	27	1	(25%)
●	8	6	(26%)
●	19	3	(25%)

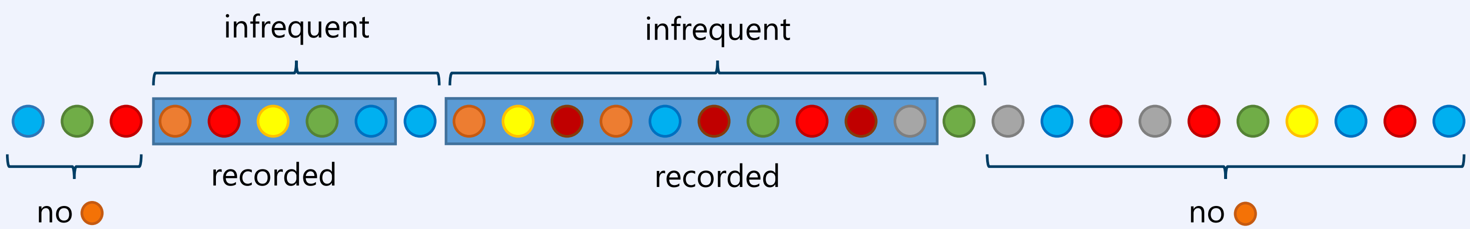


Why does it work

If ● is not recorded, ● is not frequent in the stream

Imagine marking when ● was recorded:

- if ● occurs, recording starts
- only stopped if ● becomes infrequent since start of recording



Whole stream can be partitioned into parts in which ● is not frequent → ● is not frequent in the whole stream

Algorithm is called **lossy counting**

(Manku & Motwani, 2002)

Space requirements

What is the space complexity of lossy counting?

- it reports a superset of all frequent items, how large can it be?

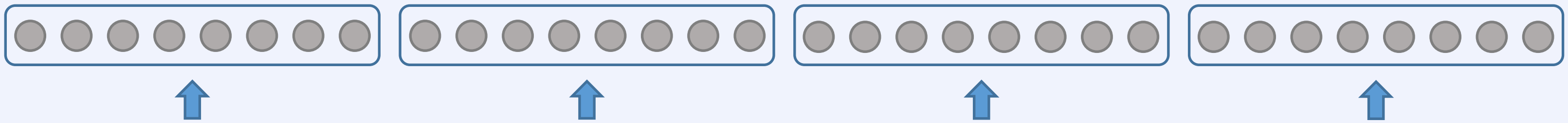
Let n be the length of the stream,
σ the minimal frequency threshold, and $k = 1/\sigma$

When is item ● in the summary?

- if it appears once among the last k items
- if it appears twice among the last $2k$items
- … if it appears x times among the last xk items
- … if it appears σn times among last n items

Space requirements (2)

Divide stream in blocks of size $k = 1/\sigma$



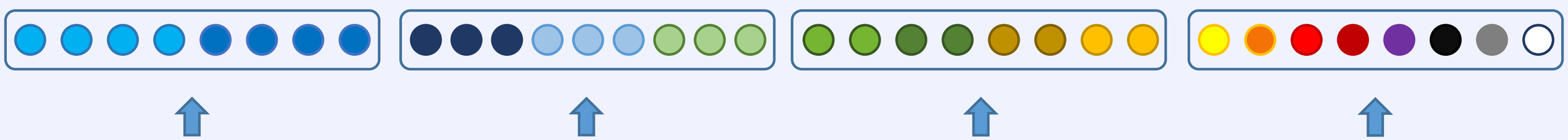
k candidates; each requires 4 occurrences

k candidates; each requires 3 occurrences

k candidates; each requires 2 occurrences

k candidates; each requires 1 occurrence

Constellation with maximum number of candidates:



$k/4$ different each appears 4 times

$k/3$ different; each appears 3 times

$k/2$ different; each appears 2 times

k different each appears 1 time

Space requirement (3)

Hence, the total space requirement is

$$\sum_{i=1}^{n/k} \frac{k}{i} \approx k \log\left(\frac{n}{k}\right)$$

Recall that $k = 1/\sigma$

- so, the worst case space requirement is $\frac{1}{\sigma}\log(n\sigma)$

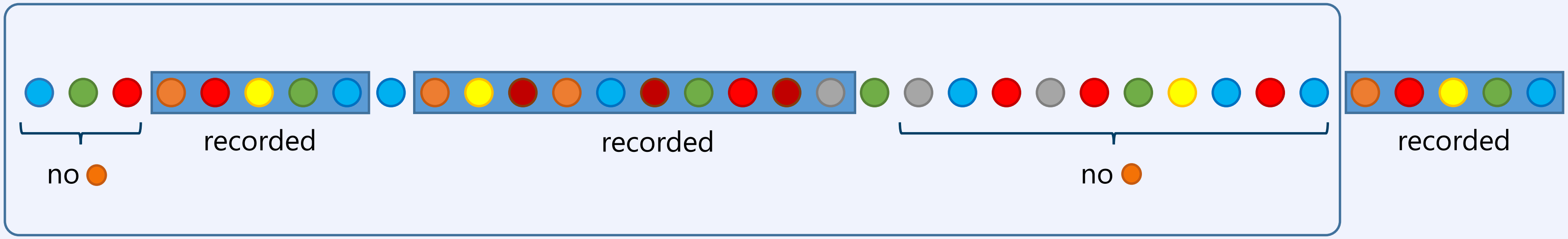
Guarantees?

Suppose we want to know the frequency up to a factor ϵ

- same algorithm, yet use ϵ as minimal frequency threshold
- report all items with count $\geq (\sigma - \epsilon)n$

Guaranteed: true frequency in the interval

$$\left[\frac{count}{n}, \frac{count}{n} + \epsilon\right]$$



Fewer than ϵn occurrences of

Summarising Lossy Counting

Worst case space consumption

- only $1/\epsilon \log(n\epsilon)$

Comes with guarantees

- with 100% certainty, the relative error for all σ-frequent itemsets is ϵ

Performs very well in practice

- and, can be optimised further
- e.g. only check if item is frequent every $\frac{1}{\epsilon}$ steps

Chapter 10.4:
Mining Frequent Itemsets

Aggarwal Ch. 12.3

What about frequent **itemsets**?

Mining frequent items is nice, but what about **patterns**?

- that is, what if we want to discover more than just frequent elements?

Solution 1: Sampling

- use reservoir sampling to maintain a reservoir of transactions
- mine frequent patterns on the sample whenever needed
- can deal with concept drift

Solution 2: Lossy Counting

- mine frequent patterns on as many segments as memory permits
- consider these **patterns** as elements, lossy-count their frequency
- without tricks will result in many false positives
- cannot deal with concept drift

Stream of Conclusions

Stream mining is exciting

- computing even trivial things becomes challenging

Surprisingly many things can be done

- especially by smart sampling and hashing

Relatively under-used in data mining

- tricky to get meaningful, sufficiently tight guarantees

Lots of potential if you get it right

Wrapping It Up

Jilles Vreeken

IRDM '15/16

15 Dec 2015

UNIVERSITÄT DES SAARLANDES

CLUSTER OF EXCELLENCE

max planck institut informatik

What did we do?

- Data Preprocessing
- Association Patterns
- Clustering
- Classification
- Sequences
- Graphs

Take Home: **overall**

Overview of the **core** topics in data mining

somewhat biased sample – by interest and available time

I wanted to give a general picture of
what data mining is, what makes it special,
and why it's so important to know

Key Take-Home Message

Data mining is **descriptive** not **predictive**
the goal is to give you insight into your data,
to offer (parts of) candidate hypotheses,
what you do with those is up to you.

Take Home: Pre-processing

It's a dirty job,
and **you** have to do it,
and do it **well**, or else no
meaningful results can be discovered.

Take Home: **Patterns**

Pattern mining aims
to provide a **simple** descriptions
of the **structures** that your data
exhibits **locally**.

Take Home: Clusters

Clustering aims to **group**
similar data points together;
infinitely many ways to define **similar**,
you have to choose **carefully**
for your domain.

Take Home: Classification

Classification aims to **predict** the label of a data point.

As **insight** is our first class citizen, we prefer methods that are **easy to inspect** over **accuracy**.

Take Home: **Structured Data**

Structure in data,
such as for **sequences** or **graphs**,
make many tasks much **more** ~~difficult~~ interesting
e.g. counting support, distance, prediction.

Lots of potential for new mining methods.

Take Home: **Streams**

Many instances where you simply **cannot store or process** all the data all the time.

Smart synopses give **accurate results** with **probabilistic guarantees**

Lots of potential for new mining methods

Take Home: Exploratory

Exploratory data analysis
wandering around your data,
looking for interesting things,
without being asked questions
you cannot know the answer of.

Questions like:
What distribution should we assume?
How many clusters/factors/patterns do you want?
Please parameterize this Bayesian network?

Things to do

Master thesis projects

- in principle: yes!
- in practice: depending background, motivation, interests, and grades – plus, on whether I have time
- interested? mail me

Student Research Assistant positions

- in principle: maybe…
- in practice: depends on background, grades, and in particular your **motivation** and **interests**
- interested? mail me, include CV and grades

Teach us More!

Well, ok… let me advertise

Topics in Algorithmic Data Analysis

together with Pauli Miettinen

Advanced Lecture

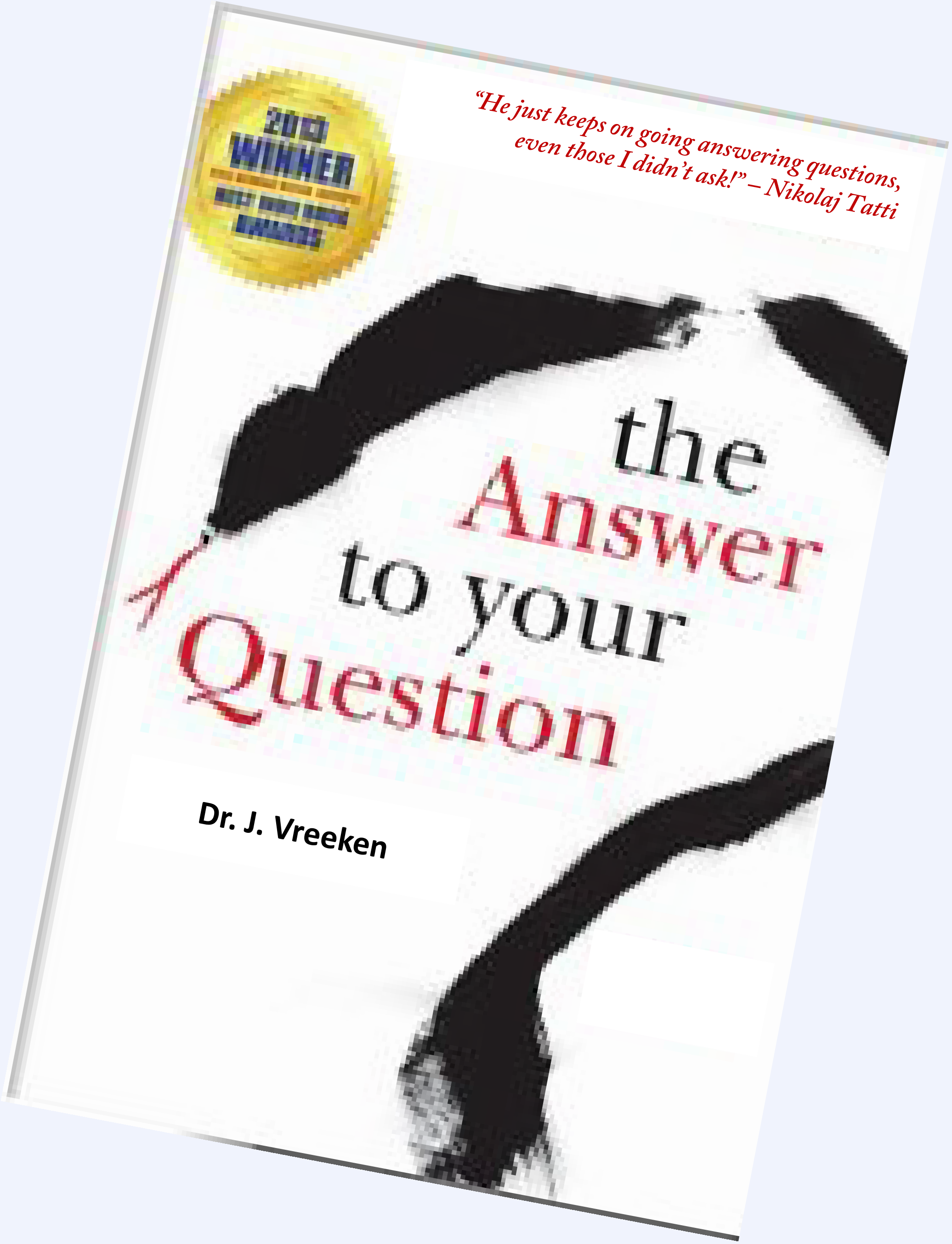
6 ECTS

In addition, Hoang Vu Nguyen and me will
likely teach a seminar next semester

Options include:

Causal Inference	(seminar+lectures)
Information Theory and Data Mining	(seminar+lectures)

Question Time!



Conclusions

This conclude
the DM part of IRDM'15.
I hope you enjoyed the ride so far.

Happy Holidays!

Thank you!

This conclude s
the DM part of IRDM'15.
I hope you enjoyed the ride so far.

Happy Holidays!