Sampling for Frequent Itemset Mining

prof. dr Arno Siebes

Algorithmic Data Analysis Group
Department of Information and Computing Sciences
Universiteit Utrecht

Why Sampling?

To check the frequency of an itemset

- we have to make a scan over the database

If in our big data context

- the database is too large to fit in main memory
- whatever smart representation we can come up with

such scans are time-consuming

- disks – including SSD's – are orders of magnitude slower than memory
  - which is orders of magnitude slower than cache

In other words

- mining on a sample will be orders of magnitude faster

In this lecture we discuss

- Hannu Toivonen, Sampling Large Databases for Association Rules, VLDB 96

Mining from a Sample

If we mine a sample for item sets, we will make mistakes:

- we will *find* sets that *do not hold* on the complete data set
- we will *miss* sets that *do hold* on the complete data set

Clearly, the probability of such errors depend on the size of the sample.

Can we say something about this probability and its relation to the size?

Of course we can, using Hoeffding bounds.

Binomial Distribution and Hoeffding Bounds

An experiment with two possible outcomes is called a Bernoulli experiment. Let's say that the probability of *success* is p and the probability of *failure* is $q = 1 - p$.
If X is the random variable that denotes the number of successes in n trials of the experiment, then X has a *binomial distribution*:

$$\mathbb{P}(X = m) = \binom{n}{m} p^m (1-p)^{n-m}$$

In n experiments, we expect pn successes, How likely is it that the measured number m is (many) more or less? One way to answer this question is via the *Hoeffding bounds:*

$$\mathbb{P}(|pn - m| > \epsilon n) \leq 2e^{-2\epsilon^2 n}$$

Or (divide by n)

$$\mathbb{P}(|p - \tfrac{m}{n}| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

Sampling with replacement

Let

- p denote the support of Z on the database.
- n denote the sample size.
- m denote the number of transactions in the sample that contain all items in Z.

Hence $\hat{p} = \frac{m}{n}$ is our sample-based estimate of the support of Z.

The probability that the difference between the true support p and the estimated support $\hat{p}$ is bigger than ϵ is bounded by

$$\mathbb{P}(|p - \hat{p}| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

The Sample Size and the Error

If we want to have:

$$\mathbb{P}(|p - \hat{p}| > \epsilon) < \delta$$

(estimate is probably (δ) approximately (ϵ) correct).

Then, we have to choose n such that:

$$\delta \geq 2e^{-2\epsilon^2 n}$$

Which means that:

$$n \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$$

Example

To get a feeling for the required sample sizes, consider the following table:

ϵ	δ	n
0.01	0.01	27000
0.01	0.001	38000
0.01	0.0001	50000
0.001	0.01	2700000
0.001	0.001	3800000
0.001	0.0001	5000000

From One to All

So, what we now have is that for one itemset I and a sample S:

$$\mathbb{P}\left(|supp_D(I) - supp_S(I)| \geq \epsilon\right) \leq 2e^{-|S|\epsilon^2}$$

Since there are a priori $2^{|\mathcal{I}|}$ frequent itemsets, the union bound gives us

$$\mathbb{P}\left(\bigvee_I |supp_D(I) - supp_S(I)| \geq \epsilon\right) \leq 2^{|\mathcal{I}|}2e^{-|S|\epsilon^2}$$

So, to have this probability less then δ we need

$$|S| \geq \frac{1}{\epsilon^2}\left(|\mathcal{I}| + \ln(2) + \ln\left(\frac{1}{\delta}\right)\right)$$

Which can be a pretty big number, given that $|\mathcal{I}|$ can be rather large

Two Types of Errors

As we already noted

- there will be itemsets that are frequent on the sample but not on the database
- just as there will be itemsets that are not frequent on the sample but that are frequent on the database

Clearly, the first type of errors is easily corrected

- just do one scan over the database with all the frequent itemsets you discovered

The second type of error is far worse.

So, the question is

- can we mitigate that problem?

Lowering the Threshold

If we want to have a low probability (say, μ) that we miss item sets on the sample, we can mine with a lower threshold t'. How much lower should we set it for a given sample size?

$$\mathbb{P}(p - \hat{p} > \epsilon) \leq e^{-2\epsilon^2 n}$$

Thus, if we want $\mathbb{P}(\hat{p} < t') \leq \mu$, we have:

$$\begin{aligned}\mathbb{P}(\hat{p} < t') &= \mathbb{P}(p - \hat{p} > \overbrace{p - t'}^{\epsilon}) \\ &\leq e^{-2(p-t')^2 n} = \mu\end{aligned}$$

Which means that

$$t' = p - \sqrt{\frac{1}{2n} \ln \frac{1}{\mu}}$$

In other words, we should lower the threshold by $\sqrt{\frac{1}{2n} \ln \frac{1}{\mu}}$

Mining Using a Sample

The main idea is now:

- Draw (with replacement) a sample of sufficient size
- Compute the set FS of all frequent sets on this sample, using the lowered threshold.
- Check the support of the elements of FS on the complete database

This means that we have to scan the complete database only once.

Although, taking the random sample may require a complete database scan also!

Did we miss any results?

There is still a possibility that we miss frequent sets. Can we check whether we are missing results in the same database scan?
If $\{A\}$ and $\{B\}$ are frequent sets, we have to check the frequency of $\{A, B\}$ in the next level of level-wise search.
This gives rise to the idea of the *border* of a set of frequent sets:
Definition

Let $S \subseteq \mathcal{P}(R)$ be closed with respect to set inclusion. The border Bd(S) consists of the minimal itemsets $X \subseteq R$ which are not in S.

Example: Let $R = \{A, B, C\}$. Then

$$\mathrm{Bd}(\{\{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}\}) = \{\{B, C\}\}$$

The set of frequent itemsets is obviously closed with respect to set inclusion.

On the Border

Theorem Let FS be the set of all frequent sets on the sample (with or without the lowered threshold). If there are frequent sets on the database that are not in FS, then at least one of the sets in Bd(FS) is frequent.

Proof Every set not in FS is a superset of one of the border elements of FS. So if some set not in FS is frequent, then by the A Priori property, one of the border elements must be frequent as well.

So, if we check not only FS for frequency, but FS $\cup$ Bd(FS) and warn when an element of Bd(FS) turns out to be frequent, we know that we might have missed frequent sets.

Finding Frequent Itemsets

Algorithm 1 Sampling-Border Algorithm

1: FS $\leftarrow$ set of frequent itemsets on the sample
2: PF $\leftarrow$ FS $\cup$ Bd(FS) {Perform first scan of database}
3: $F^{(0)} \leftarrow \{I : I \in$ PF and I frequent on $D\}$
4: NF $\leftarrow$ PF $\setminus F^{(0)}$ {Create candidates for second scan}
5: **if** $F^{(0)} \cap$ Bd(FS) $\neq \emptyset$ **then**
6: **repeat**
7: $F^{(i)} \leftarrow F^{(i-1)} \cup$ (Bd($F^{(i-1)}$) $\setminus$ NF)
8: **until** no change to $F^{(i)}$
9: **end if** {Perform second scan}
10: F $\leftarrow F^{(0)} \cup \{I : I \in F^{(i)} \setminus F^{(0)}$ and I frequent on $D\}$
11: **return** F

Discussion

As we already noted

- the sample size grows linearly with $|\mathcal{I}|$ and, thus, can become rather large

moreover, step 1 of the algorithm is obviously efficient

- but from then on we can be out of luck
- $F^{(i)}$ could grow into the rest of the lattice
- which means we run the naive algorithm!

So, the question is

- could we derive tighter bounds on the sample size,
- and, at the same time, can we have direct control on the probability that we miss frequent itemsets?

Lowering the threshold gives us indirect control

- why?

A Crucial Observation

In computing the sample size

- p was the probability that a *random* transaction supports itemset Z

That is, we were using an itemset as an

indicator function

For $t \in \mathcal{D}$:

$$1_Z(t) = \begin{cases} 1 & \text{if } Z \subseteq t \\ 0 & \text{otherwise} \end{cases}$$

Slightly abusing notation, we will simply write Z rather than 1_Z

- that is we will use Z both as an itemset and as its own indicator function

Indicators are Classifiers

So, given a transaction database D and an itemset Z, we have

$$Z : D \to \{0, 1\}$$

For those of you who already followed a course on

- Data Mining, Machine Learning, Statistical Learning, Analytics, ...

or simply keep up with the news. This must look eerily familiar:

the observation tells us that Z is a classifier

This means that if there would be a theory about sample sizes for classification problems

- we might be able to use that theory to estimate sample sizes for frequent itemset mining

And it happens that there is such a theory:

Probably Approximately Correct learning

Classification

Learning Revisited

We already discussed that the ultimate goal is to

learn $\mathcal{D}$ from D

Moreover, we noted that for this course we are mostly interested in learning

a marginal distribution of $\mathcal{D}$ from D

More in particular, let

$$D = X \times Y \sim \mathcal{D} = \mathcal{D}|_X \times \mathcal{D}|_{Y|X} = \mathcal{X} \times \mathcal{Y}$$

Then the marginal distribution we are mostly interested in is:

$$\mathbb{P}(Y = y \mid X = x)$$

where $\mathcal{Y} = \mathcal{D}|_{Y|X}$ (and thus Y) is a finite set

Classification

The rewrite of $\mathcal{D}$ to $\mathcal{X} \times \mathcal{Y}$ was on purpose

- $\mathcal{X}$ are variables that are easy to observe or known beforehand
- $\mathcal{Y}$ are variables that are hard(er) to observe or only known later

In such a case, one would like to

- predict $\mathcal{Y}$ from $\mathcal{X}$
- that is, given an $X \sim \mathcal{X}$ with an observed value of x
  1. give the marginal distribution $\mathbb{P}(Y = y \mid X = x)$
  2. or give the most likely y given that $X = x$
  3. or any other prediction of the Y value given that $X = x$

Given that (Y) is finite, this type of prediction is commonly known as classification. Bayesians prefer (1), while most others prefer (2).

While I'm a Bayesian as far as my statistical beliefs are concerned

- it is the only coherent, thus rational, approach to statistical inference

we will focus, almost always, on (2)

Classification, continued

Answering the question which y is the most probable is easy if you know the marginal distribution $\mathbb{P}(Y = y \mid X = x)$

- simply select that y that has maximal probability
- this is the Bayes optimal solution

If that is the only thing we want,

- learning the complete (marginal) distribution seems overkill

After all,

- the exact probability values are unimportant
- only the ranking the highest one right matters

For that reason, classification is often studied as the problem of

- learning a (computable) function $h : \mathcal{X} \to \mathcal{Y}$
- such that $h(x) = \operatorname{argmax}_y \mathbb{P}(Y = y \mid X = x)$
- from $D = X \times Y$

Why? Simpler Means Simpler, probably

Learning a computable function $h : \mathcal{X} \to \mathcal{Y}$ should be simpler than learning the marginal probability distribution. For,

- $\mathbb{P}(Y = y \mid X = x)$ contains more information than
- $\operatorname{argmax}_y \mathbb{P}(Y = y \mid X = x)$
- in the sense that you can use the former to compute the latter, but not vice versa

That is, an algorithm that computes the marginal distribution is easily extended in an algorithm that computes the function h.

- Hence, it is reasonable that expect that computing the classification function has lower complexity than computing the marginal distribution
  - in terms of the amount of data needed
  - in terms of computational resources

Note that a reasonable expectation is not necessarily always true

Loss Functions

Say our algorithm learns function h from D, the obvious question is:

- how good is h?

Intuitively this is easy

- the assumption is that there is a *true* function $f : \mathcal{X} \to \mathcal{Y}$
- so we simply compare h with f
- the more often they agree, the better h is.

This comparison is known as a loss function

So, intuitively, the loss is

$$L_f^i(h) = |\{x \mid h(x) \neq f(x)\}|$$

or, if you want, the average of this (over all possible x values). The intuition is good, mathematically, it stinks however.

Cleaning Up Mathematically

The reason this intuitive definition fails is

- the domain we are dealing with may very well be infinite
  - i.e., we need measure theory to make clear what the size of the set is
- the intuitive definition counts
  - a failure for an x that appears once every eon
  - as bad as one that occurs every second

  clearly, that doesn't make sense

Fortunately, both problems disappear if we turn to probabilities:

$$L_{\mathcal{D},f}(h) = \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)]$$

That is, the loss of using h rather than f is

the probability that we make a mistake

Cleaning Up, Realistically

While this is a nice loss function, probably the best one possible, there is a small problem

- it depends on both $\mathcal{D}$ and f, and we know neither!
- in fact, that is what we want to learn
  - as holy grail and as simpler goal respectively

All we have is D, our (finite!) sample

Hence, we have to make do with the training error, aka empirical error, aka empirical risk:

$$L_D(h) = \frac{|\{(x_i, y_i) \in D \mid h(x_i) \neq y_i\}|}{|D|}$$

Finding a function that minimizes this is loss is known as Empirical Risk Minimization (ERM).

Learning Classifiers Isn't Easy

ERM may seem to make learning an easy problem

- simply search for a hypothesis that minimizes the risk

Unfortunately it isn't that simple, we briefly discuss two ways we can go wrong

- overfitting
- a too rich hypothesis class

Overfitting

Let $D \subseteq \mathcal{D}$ (i.e., the values we find in our sample) be such that:

- $\forall (x, y) \in D : y = 1$ while
- $\forall (x, y) \in \mathcal{D} \setminus D : y = 0$

A function that minimizes the empirical risk is given by

$$h(x) = 1$$

Depending on the respective sizes of $\mathcal{D} \setminus D$ and D, the true loss can be arbitrarily big

- we will miss-classify every new example!

This is what is known as overfitting.

- the example may look a bit contrived, but the problem is real
- an aspect of the problem of induction

The solution we will mostly take is: restricting the hypothesis class

The Online Game

The goal is to learn a simple threshold, i.e., our set of hypotheses is given by

$$\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}\}, \text{ where } h_\theta(x) = \text{sign}(x - \theta)$$

for $t = 1, 2, \ldots$

- our learner is presented with example $x_t \in \mathcal{X}$
- the learner predicts $\hat{y}_t$
- he is shown the true y_t
- if $\hat{y}_t \neq y_t$ the cost is 1

The goal is to make as few mistakes as possible.

(we are now following some slides from Shai Shalev-Shwartz)

Learning Thresholds

The goal is to learn a simple threshold, i.e., our set of hypotheses is given by

$$\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}\}, \text{ where } h_\theta(x) = \text{sign}(x - \theta)$$

The three rational learners are

$$\hat{\theta}_t^r = \min\{x_{t'} \mid t' \leq t \wedge f(t') = 1\}$$
$$\hat{\theta}_t^l = \max\{x_{t'} \mid t' \leq t \wedge f(t') = -1\}$$
$$\hat{\theta}_t^m = \frac{\hat{\theta}_t^r + \hat{\theta}_t^l}{2}$$

but you can pick any learner you want.

Your adversarial teacher knows θ and he knows your current estimate $\hat{\theta}_t$, so he will choose

$$x_{t+1} = \frac{\theta + \hat{\theta}_t}{2}$$

and your learner will be wrong *every time*

Too Rich

You may be shocked that we cannot even learn such a simple example

- the reason is that it isn't simple at all
- the hypothesis class is far too rich (expressive)

Recall that

- there are only countably infinite computable numbers
- while there are uncountable many real numbers
  - strictly and far more

You adversarial teacher can force you to try to learn an uncomputable number

- which is obviously impossible
- for, how would you be able to learn a number that has not even a finite representation?

Our Approach

Note that if we restrict ourselves to integer thresholds

- for the moment both for the hypotheses and the true classification function

it is suddenly an easy to learn task.

- can you think of an algorithm?

The approach we take is that we

- first discuss the simple finite case
- to understand why learning always works in finite cases
- and then generalize to infinite cases having certain desirable properties
- ending up by showing that these properties are not only sufficient but also to a large extent necessary

And somewhere along this route, we'll apply our newly found knowledge to frequent itemset mining