Big Data Essay: Improving Sampling Bounds For Frequent Item Set Mining By PAC Learning?

Yuxuan Hou, Student Number: 4962273

1 Introduction

The world is now made up of data, and we live in this environment. People's daily lives generate a lot of data, and all of these data have some kind of meaning and are very valuable. So we are trying to use all the data - analyzing people's daily behavior (e.g. shopping preferences) - to change people's lives better. Currently, many scholars have proposed various ways to analyze this data.

One approach is to use a sample taken from a distribution in a dataset to represent the dataset itself, and to use this sample to obtain information that can be more broadly applied to the entire dataset. This essay focuses on the methods proposed by Matteo Riondato and Eli Upfal[1], as well as Hannu Toivonen[2] to draw a statistically representative sample and analyzes whether the use of PAC learning improves sampling performance.

Problem Statement. Due to the huge amount of data that is increasing day by day, it is important but difficult to make the obtained data split into their respective domains and make them really valuable to provide useful information for each domain. We hope to find algorithms that can help us efficiently obtain this potentially valuable information. In the previous paragraph we have briefly introduced a very clever approach to analyze the overall dataset using representative samples. This is a generalization process.

However, the disadvantage of this approach is that the final dataset will produce erroneous results due to incorrect sample sizes, such as miss the correct results. Thus, our problem becomes: how to minimize the error in our sample analysis. Or, how can we be sure that we find the right size of the sample that minimizes the data error?

2 Frequent Itemset Mining

From Association Rules to Frequent Itemsets Taking the supermarket as an example, we form a transaction-based database with the transaction information of these itemsets. And this database only includes information about all items in one transaction, and removes some irrelevant information such as quantity, so that we can easily know the specific itemset information included in each transaction. Here, the itemset refers to the set of purchased items. The association rule is an expression like $X \rightarrow Y$(e.g. beer $\rightarrow$ diapers), where X and Y are two disjoint item sets. This rule means that we can identify whether there is some association between these item sets. We call the set of items that occur together more frequently than a certain threshold θ a frequent itemset. Then it is logical that the more frequent itemsets are more closely related to each other.

Problem The total number of association rules between sets of items extracted in transactional databases is exponential, which makes association analysis difficult. We want to reduce the computational effort in rule extraction and ensure that the extracted rules are correct and trustworthy. We have used thresholding to improve this problem. For example, if a rule contains items with few occurrences, it may appear by chance and is not enough to be considered a hidden rule in the dataset, and restriction thresholds are used to eliminate this class of candidate rules. But it is not good to extract rules directly based on thresholds, because the level of thresholds can lead to significant changes in the number of rules, or in some cases can produce unreasonable rules.

Solution - Apriori Algorithm To solve this problem, we can extract frequent itemsets first, and then extract rules from frequent itemsets. When generating frequent itemsets, we can again reduce the candidate frequent itemsets first, thus reducing the itemset search time. Based on our prior knowledge, it is easy to think that if an itemset is not frequent, then the superset of this itemset must be not frequent; if an itemset is frequent, then the subset of this itemset must be frequent. With the a priori knowledge, we can prune the itemsets in advance. First, we set a threshold and scan through the dataset to find the first level of frequent itemsets that contain only one item in the itemset. Then, we combine the items from the first level frequent itemsets to generate the second level of itemsets (in the right order), and then scan the dataset again to find the second level of frequent itemset that meets the threshold. And so on, the candidate k-level frequent itemsets are generated based on the (k-1)-level frequent itemsets, and then the dataset is scanned to find the k-level frequent itemsets with

k=3,4,..., until no more frequent itemsets can be generated. This is a very smart process, where the results obtained at each step discard the itemsets that we already know are infrequent in the previous level, thus greatly reducing the overall search time. From the original exponential need of $O(2^{|\mathcal{I}|})$ (worst case, search for all itemsets $\mathcal{I}$) to $O(|\mathcal{I}|^m)$, where m represents the maximum expected size of frequent itemsets.

Frequent Closed Itemsets There are more redundancies in frequent itemsets, so people have introduced the concepts of frequent closed itemsets. Frequent closed itemset means that, for all itemsets $\mathcal{I}$, and transactional database sets D, the following mappings are defined: 1) for x belonging to itemset $\mathcal{I}$, *f(x)* is the database set in which D contains x; 2) for y belonging to database sets, *g(y)* is the itemset in which all y are contained. Based on this, for general x, *g(f(x))* may be greater than x. Then the closed itemset is the set of items x that satisfy $g(f(x)) = x$. If this itemset is closed and its support is larger than or equal to threshold θ, then we call it frequent closed itemsets. Not abstractly, we can assume that people always buy {diapers, beer, milk}, then for the frequent itemset, any two and more combinations of these three goods are frequent itemsets, but only {diapers, beer, milk} is a frequent closed itemset, i.e., there will be no other items that always appear with the frequent closed itemset, otherwise *g(f(x))* would contain those other items.

Sampling Although we ultimately want to obtain the information exhibited by the entire database, it is not feasible to traverse the database, find candidate frequent sets, and subsequently traverse it again... We can use the extracted sample instead of this database. So, how to find the correct sample size so that this sample is exactly the best representation of this database? Here we assume that all our data sets D are sampled according to $\mathcal{D}$ with independent and identical distributions (i.e. *iid*).

The ideal and worst case is that our sample is the same size as the database (worst), so that the possible error is 0 (ideal). We are trying to find a maximum probability bound of the error making. We use Hoeffding bounds to relate sample size to the probability of error. As more samples are drawn, the more the final predicted true support p, will converge to the estimated support, $\hat{p}$. Using Hoeffding's inequality we can get: for the inequality $\mathbb{P}(|p - \hat{p}| > \epsilon) < \delta$, we can know that the estimate is probably (δ) approximately (ϵ) correct.

So for one itemset and a sample, when the sample size reaches a certain level, we can bring them in the inequality, and we know that samples and itemsets will gradually approach each other, and the error between these two will gradually decrease. The number of samples n must be greater than or equal to $n \geq \frac{1}{2\epsilon^2} ln \frac{2}{\delta}$, making the probability less than δ. In a practical sense, when the size of the sample is larger than this value, sample will better represent itemsets. But using sample to represent itemsets will also have the following two main problems: 1) it would frequent in sample but not on the real database; 2) it would not frequent on the sample but should actually frequent on the real database. To solve the first problem, we can re-run a scan of the database to check whether the frequent itemsets in the sample are equally frequent in the database, and then recalculate them and compare them with the threshold value. But for the second problem, we need to start by lowering the threshold. So when we restrict the new threshold to θ' and lower probability of missed itemsets on the sample to μ, then to reduce the occurrence of problem 2, we resubstitute Hoeffding's inequality to obtain the new threshold $\sqrt{\frac{1}{2n} ln \frac{1}{\mu}}$. Since the new lower threshold is used, to prevent the occurrence of problem 1, we need to draw a sufficiently large sample; use the reduced threshold to filter out the frequent itemsets on the sample by the Apriori algorithm; remove the itemsets below the original threshold. This has the advantage of enabling us to remove itemsets that are frequent itemsets in the sample but infrequent in the database by scanning the database only twice.

From Frequent Itemset Mining to Classification By observing the problem, we find out that this involves a classification problem. Frequent itemset mining relates to finding frequent itemsets in a database: We assume that for each transaction t in $\mathcal{D}$ denote by 1 or 0 whether the itemset appears in t. We find the indicator variable to be true and therefore the transaction contains this itemset exactly. Therefore, given an itemset, we map dataset D to either 0 or 1 (false or true) using the classifier function just described. This conclusion allows us to relate the problem of frequent itemset mining to the problem of classification and, in particular, to the theory of PAC learning.

Here, we introduce one term proposed by Riondato and Upfal: *range space*, and we will use this term in section 4.2. We use hypothesis class to represent a set of classifiers (we will talk about it in section 3). Each classifier will correspond labels and elements one by one. This means that a set of hypotheses is also a subset of our domain X. So we have the range space *(X, R)*. This includes a finite or infinite set of points X, and a finite of infinite family R of subsets of X, called ranges, which are simple classifiers. The projection of a range on a subset A is the intersection of the elements of this range set with the domain.

3 PAC-learning

In PAC, we can split it into two parts: *Probably* and *Approximately Correct*. Two of them correspond to the confidence parameter (1-δ), and the other one corresponds to the accuracy parameter ϵ.

3.1 Classification, Quality and Convergence

Classification and Hypothesis Class In the classification problem, we want to predict Y from X. Then suppose there exists such a function h (which is learned by the algorithm from the dataset D) that can help us predict Y from X. Then we will be particularly interested in how well h predicts the outcome. The function that can predict the true outcome we consider as f. Then we want to minimize the difference between the true function f and our learned function h (the loss function). That is, we need to find a loss function such that the loss function is small enough that we can obtain the most similar results available. To reduce the difficulty of computing the loss, we use the probability of possibly making a mistake as the loss from using function h instead of the true function f. But since we now only know our finite data set D, we want to train a model such that this model minimizes the training loss (empirical risk). This process we called *ERM (empirical risk minimization)*.

We have many functions that can be used for prediction (e.g. classification tree), and the set of these possible functions is called hypothesis class $\mathcal{H}$. We want to find a hypothesis that minimizes the risk in a finite number of hypothesis classes. We assume that there exists a hypothesis h^* in our hypothesis class that is learned from D such that the loss obtained by prediction on the sample is 0 (*The Realizability Assumption*). That is, by performing an ERM on hypothesis class $\mathcal{H}$, for any sample set D, we obtain a hypothesis h_D that results in a loss value of 0.

To find this ideal hypothesis, we utilize The Halving Learner algorithm. That is, we let all hypothesis classes perform the prediction, and finally only leave the hypothesis that gives the correct label; subsequently, we repeat the filtering of these hypothesis classes several times until finally we are left with the best hypothesis that can correctly predict all the data. This linear algorithm's complexity is $O(|D|)$. But back to the point of our discussion: we still want to reduce the complexity by sampling. We want to make the sample give a small error rate even in the worst poor conditions, so that other less poor samples can naturally show better prediction results.

Bad Sample and Sample Size Bound We already know that the final loss value is related to the sampling D and the chosen function h_D, and the true loss $L_{\mathcal{D},f}(h_D)$ becomes a random variable. The generation of the training set is a random event, so our samples may not be representative. We use δ to denote the probability of drawing these unrepresentative samples. When the hypothetical true loss learned from the sample is greater than the accurate parameter ϵ, we consider this sample to be bad, i.e., it is a failed learning. We want to find an upper bound on this failure probability, a probability that the true loss of the hypothesis learned on this dataset is greater than ϵ. Since we have realizable hypotheses, the hypotheses selected by ERM must satisfy the loss of 0. Then we fail only due to some bad hypotheses that perturb the final result and deceive ERM. we use $\mathcal{H}_B$ to represent the set of these bad hypotheses. By integrating multiple inequalities, the probability that we get misleading sample is less than the number of bad hypotheses, which in turn has a value less than or equal to the number of total hypotheses $|\mathcal{H}|e^{-\epsilon m}$. That is, if we have a large enough training set m, not too large, but at least $\frac{log(|\mathcal{H}|/\delta)}{\epsilon}$, we can ensure that for any labeling function and distribution, we have at least 1-δ confidence that the true loss for any ERM hypothesis will be less than or equal to ϵ. Thus, we know that every finite hypothesis class $\mathcal{H}$ is PAC learnable with sample complexity $m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \frac{log(|\mathcal{H}|/\delta)}{\epsilon} \rceil$.

3.2 Realizable vs Agnostic PAC-learning

In general, because of *no free lunch theory*, we cannot find a general algorithm that holds for any distribution. We can find a good algorithm only if we have some prior knowledge(e.g., we know in advance that an image has a certain bias, or that the data itself has a certain structure), so that we can narrow the scope of the algorithm and choose the algorithm with a better performance. PAC learning introduces the prior knowledge via "realizable" over a hypothesis class, but we can also use agnostic PAC learning to relax the hypothesis by adaptive loss and then introduce prior knowledge.

Realizable PAC-learning We can see realizable assumption as the prior knowledge, that is, when the hypothesis class is given, the distribution must be reached with loss equal to 0. This also restricts the distribution cannot be arbitrary, it must be such that one of the hypothesis in its class $\mathcal{H}$ has to satisfied loss equal to 0. (Notice that, PAC learning restricts the distribution, while no free lunch is for arbitrary distributions, and thus we say that with PAC learning, we can obtain a learning algorithm that, with a high probability when the number of samples is large enough, makes the true loss on the distribution relatively low).

But $\mathcal{H}$ always contains a true hypothesis is not very realistic, then we can reasonably assume that the true loss of the hypothesis is always greater than 0, but

not equal to 0. Then we want to know, how big is the difference between the true loss and the loss of the sample ($|L_{\mathcal{D}}(h) - L_D(h)|$)? That is, how much worse our estimate of the loss will be after using the sample. By using Hoeffding's inequality, we already know that the probability that this difference is greater or equal to ϵ is less than or equal to $2e^{-2m\epsilon^2}$. We bring this inequality to all hypotheses. For multiple probabilities that the difference is greater than ϵ, we want at least one of them to be greater than ϵ. Subsequently, our union bound helps us to get δ as $2|\mathcal{H}|e^{-2m\epsilon^2}$. By comparing the bounds in consistent and inconsistent case, the number of examples we need to look at is the square of the number of examples we need to use. We have to look at the true hypothesis, so if we know that the true hypothesis exists with $\mathcal{H}$, we can finish faster; but if we don't, then we have to get a larger sample, which results in a slower algorithm.

Agnostic PAC-learning Because we realized that the realizability assumption is too strict and sometimes does not meet our requirement for reality. For example, it is quite possible to experience cookies with the same shape and color, but one tastes good and one does not. However, the realizable assumption excludes this possibility, and this is obviously not realistic. So we made an extension and relaxation of it using *Agnostic PAC learning*. We subtracted the requirement for realizable, in this case, the labeling function will be a probability function that assigns the label to 1 if the probability of the result being 1 is greater than or equal to 1/2, as this minimizes the error. Then the final conclusion is that the loss $L_{\mathcal{D}}(h) \leq \epsilon + term$, and this term is the loss generated by a hypothesis h' that minimizes the loss in all hypothesis classes $\mathcal{H}$: $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$. This added item in agnostic PAC learning relaxes our scope. If we are not sure whether a true hypothesis exists, we can still have an algorithm that returns a result that is likely to be close to the best possible outcome. Otherwise, when the hypothesis class itself is realizable, this term equals to 0, then it returns to the previous state of PAC learning.

General Sample Size We want to know which hypothesis class can do PAC learning. Since the performance of hypothesis depends on the quality of the sample, and we consider a hypothesis on this sample to be good when the loss produced by the hypothesis is approximately true loss, that is, their difference is less than or equal to ϵ, this sample is good and we call it ϵ-*representative*. ϵ-representative is good, because if we know if we learn from ϵ-representative sample, we can achieve PAC learning. Rather than directly proving that we can do PAC learning, we prove that a hypothesis class has the *Uniform Convergence Property*. It means that, If our sample is big enough, that its dataset will be ϵ-representative with probability of at least 1-δ. That is, we can again prove that a hypothesis class is PAC learnable by proving this property. By a similar procedure as in the previous proof of PAC learnable, and reusing Hoeffding's inequality, we can obtain that if the loss function is bounded in the interval [a, b], when the sample complexity $m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \lceil \frac{(b-a)^2 log(2|\mathcal{H}|/\delta)}{2\epsilon^2} \rceil$, then this finite hypothesis class is PAC learnable; and when the sample complexity is $m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \lceil \frac{2(b-a)^2 log(2|\mathcal{H}|/\delta)}{\epsilon^2} \rceil$, then this $\mathcal{H}$ is agnostically PAC learnable. By comparing the new derived sample $m_{\mathcal{H}}(\epsilon, \delta)$ with the previous one (Section 3.1), we can obtain that since we are not assuming the existence of the true hypothesis, our minimum sample size grows by $\frac{1}{\epsilon}$; and we also have $(b-a)^2$ now (i.e. how big the loss can be, which we can usually simply normalize to [0,1]). In comparison, the only significant change we make is the change from $\epsilon \to \epsilon^2$, which is a very cost-effective change, as we get more general results.

3.3 VC-Dimension and the Fundamental Theorem

Our previous theories were implemented based on the finite hypothesis class. But imagine facing an infinite hypothesis class, it may become not so simple. For example, even the simplest linear classifier is not a finite set. The VC dimension is a good solution to this problem. [1]

Shatter and VC-Dimension Suppose we need to distinguish whether a cookie is sweet or salty, we can use size and softness to make predictions about a cookie. This is a simple binary classification problem. Suppose our classifier is linear, i.e., a straight line can classify sweet or salty. Imagine we have two sets of data, and these two sets are randomly combined, and we get three results, i.e., all sweet, all salty, and one sweet and one salty. In either case, we can use a straight line to separate the two results. This shows that the linear model is able to shatter two sets of data. But if we have four sets of data, we can't guarantee that a line will successfully divide the two categories. For example, the top left is sweet, the bottom left is salty, the top right is salty, and the bottom right is sweet. This shows that the linear model cannot divide four sets of data. But if it is three groups of data, we can successfully separate them again. This means that the VC-Dimension of the linear model in this case of two-dimensional data is 3. (Because the linear model can shatter at most 3 groups of data). Now, if we change the model to non-linear, then we have many

[1] In my understanding, although the hypothesis class is infinite, the number of valid hypotheses is finite when applying to a certain sample size. So we can replace the infinte hypothesis class with a finite number (VC-Dimension) and finally achieve the purpose of reducing the loss.

possible results, which means that the VC-Dimension becomes more. So in layman's terms, VC-Dimension is how inclusive a certain type of model is of the number of data. The higher the VC-Dimension, the more inclusive it is. Or VC-Dimension is the complexity of the model, the larger the model hypothesis class, the higher the VC-Dimension. The role of VC-Dimension is that it can help us to choose a better model, in this case, a model with lower risk.

Therefore, in general terms, when there is a set of size C, we say that the set C is *shattered* by $\mathcal{H}$ if the hypotheses in the hypothesis set $\mathcal{H}$ can be divided into $2^{|C|}$ different combinations of functions. And the *VC-Dimension* of a hypothesis set $\mathcal{H}$ is the size of the largest data set that can be shattered by $\mathcal{H}$. If the VC-Dimension of $\mathcal{H}$ is infinite, this means that $\mathcal{H}$ is also infinite, which means that it is not PAC learnable.

The Fundamental Theorem We have introduced in the previous paragraph the relation between VC-Dimension, $\mathcal{H}$, and PAC learnable, i.e., $\mathcal{H}$ is PAC learnable if and only if VC($\mathcal{H}$) is finite. This relation we call *The fundamental theorem*. In this part we link finite VC complexity and agnostic PAC learnable by proving Uniform Convergence (UC).

Since we have already shown that the hypothesis class with Uniform Convergence property is agnostic PAC learnable. Then, to prove The Fundamental Theorem, we can show that small effective size has the property of uniform convergence. We want to obtain bounds on the expectation of the difference between the supremum of each hypothesis's loss and the true loss, and we want this difference small for all $\mathcal{H}$. We use the property of the convex function, that is, the value of this function at the average is always less than or equal to the average of the function values. We arrive at the inequality: $\mathbb{E}_{D\sim\mathcal{D}^m}\big(sup_{h\in\mathcal{H}}|L_D(h) - L_{\mathcal{D}}(h)|\big) \leq \mathbb{E}_{D,D'\sim\mathcal{D}^m}\big(sup_{h\in\mathcal{H}}|L_D(h) - L_{D'}(h)|\big)$. This is not good enough as we want to get rid of supremum's effect. The two datasets in this inequality are exactly same size (the expectation or average stays the same whether we move a point from the dataset to the test-set). This means that we can treat this difference sign as positive or negative. Thus, the equation becomes the one with an external expectation loop and an internal expectation loop, and the inner loop we treat as a finite loop, which, helps us to link supremum and maximum. We use Hoffeding's inequality to get a bound on the probability, and then we go back to the expectation and get that our bound of the expectation for the difference now becomes $\frac{4+\sqrt{log(\mathcal{T}_{\mathcal{H}}(2m))}}{\sqrt{2m}}$. Finally, we use Markov's inequality to show that the bound we have now can be as samll as we want.

By narrowing the bound, we prove that these six statements are equivalent: *1) $\mathcal{H}$ has the uniform convergence property; 2) Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$; 3) $\mathcal{H}$ is agnostic PAC learnable; 4) $\mathcal{H}$ is PAC learnable;*[2] *5) Any ERM rule is a successful PAC learner for $\mathcal{H}$; 6) $\mathcal{H}$ has a finite VC dimension.* So being learnable is equivalent to having a finite VC-Dimension, then it is PAC learnable. So up to here, we have linked all the concepts that have been mainly described so far. But we still need to be aware that just because a hypothesis can be learned does not mean that the results we can find are good or that we can learn it in an efficient way.

4 Frequent Itemset Mining on Big Data

Back to the original question, now what we trying to do is to find the sample bound for itemset mining. So how can we use PAC learning to give a sample bound for frequent itemset mining and what guarantees will we get after we apply on this bound.

4.1 Toivonen and Frequent Itemsets

We already know that our goal is to find a suitable sample size that minimizes the error in the final result. Toivonen tells us that we can perform sample extraction for frequent itemset mining and gives the bound of the sample size.

In the previous section on sampling we have shown how to find a suitable sample size. by Hoeffding's inequality, so that for a certain sample size, the true support p we predict will gradually converge to the estimated support $\hat{p}$, and we want the probability that the error between these two is greater than ϵ to be less than or equal to δ. By computing, we obtained a bound on this sample size n as follows: $n \geq \frac{1}{2\epsilon^2} ln\frac{2}{\delta}$. But we are not satisfied with this result; our goal is to explore the sample bounds on all itemsets $\mathcal{I}$. Using union bound, Toivonen gives the result, our sample S should:

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

. However, here, as $\mathcal{I}$ could be linear and become large, we still need to tight the bound.

4.2 PAC-learning and Frequent Itemsets

One critual notice is that the itemsets are classifiers. It gives 1 if we contain it and results 0 if we don't have it. Thus, itemset mining is from an unsupervised machine learning problem change to supervised learning as it is a classification problem. We also

[2]Here, we say statement 3 and 4 are equivalent simply means: if we have a hypothesis with a finite VC-Dimension and we know that the realizable assumption holds, then we can do PAC learning, even if we don't know whether it holds or not, we can still do agnostic PAC learning. This just means that the sample size in these two cases is different.

call the set of classifiers as hypothesis class, and combine with the fundamental theorm, we roughly have the following logical chain: *frequent itemset ming* $\rightarrow$ $Classification \rightarrow$ *PAC learning*.

We introduced range space at the end of section 2. The final projection takes the form: $\sqcap_R(A) = \{r \cap A | r \in R\}$. This means that a subset $A \subset X$ is shattered by R if $\sqcap_R(A) = P(A)$. Remember that we used ϵ-representative to represent the goodness of a sample in Section 3.2. We now translate this to ϵ-approximations. we describe it using range space rather than the set of classifiers. Since ranges are simply classifiers, we utilize here the difference between the loss on range A compared to the loss on range B. That is, the ϵ-approximation means that our estimated loss is always close to the true loss.

We translate all our knowledge into frequent itemset mining. We first need a range space, which refers to all transactions, and the ranges are the support sets of the closed itemsets. Now we need to determine the size of the VC-Dimension and find a tight upper bound. We assume that there exists a subset A of size υ. When each subset of A has an itemset I_B coinciding with it, then the VC-Dimension of this range space is greater than or equal to υ. That is, if we look for transactions in A that support items related to B, what we find is B itself (i.e., $T_A(I_B) = B$). Here we restrict the classifier to A, but we actually want to do this for all subsets, which is a process of shattering. This is an awesome idea, if we can shatter sets of size υ, then we have a VC-Dimension of at least υ. But it' s complexity is still large. Fortunately, we have *d-bound*.

We need to have at least υ transactions that all of them at least υ long. This is to ensure that we can achieve $\forall B \subset A : \exists I_B \subset \mathcal{I} : T_A(I_B) = B$. Let D be a data set. The d-bound of D is the largest integer d such that D contains at least d different transactions of length at least d. Then this d becomes tight upper bound of VC-Dimension. Thus, our VC-Dimension would be at most d and thus becomes finite, which means we can do PAC learnable on it!

Finally, let S be a random sample of dataset D, d be the d-bound of D and c is constant (≤ 0.5 experimentally). We have:

$$|S| \geq min\{|D|, \frac{4c}{\epsilon^2}(d + log\frac{1}{\delta})\}$$

Thus, we get an approximation of all frequent itemsets with simply 2 scans of our dataset.

4.3 Toivonen vs PAC-learning

So so far, we have two different sample size bound:

$$Toivonen's : |S| \geq \frac{1}{\epsilon^2}(|\mathcal{I}| + ln(\frac{2}{\delta}))$$

Riondato and Upfal's:

$$|S| \geq min\{|D|, \frac{4c}{\epsilon^2}(d + log\frac{1}{\delta})\}$$

The first one is just a simple sampling bound while the other one is the bound using PAC learning. We want to have a tighter bound, i.e. comparing these two different bounds, we want to choose the one with a smaller sample size, under the same conditions. Comparing the two equations, we find that the size of these two equations is mainly influenced by $|\mathcal{I}|$ and d. Since $|\mathcal{I}|$ is affected by the size of itemsets, Toivonen's results depends linearly on the number of individual items appearing in the dataset; while d is affected by the range space. Although Riondato and Upfal illustrate that our sample size is also linear, it is more dependent on the VC-Dimension of the range space associated with the dataset. In a very large dataset, a constraint such that there are at least d transactions in the dataset and each transaction is at least d in length is very strict, and thus draw a smaller and more rigorous sample while doing the sampling. Unfortunately, in our real life, our database of transactions is extremely large. So when facing a very large database, we would get a smaller sample size using Riondato and Upfal's bounds, which means they have better performance. So we say that using PAC learning actually improves our initial (Toivonen's) bounds.

References

[1] Matteo Riondato and Eli Upfal. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. 8(4), 2014.

[2] Hannu Toivonen. Sampling large databases for association rules. page 134–145, 1996.