Big Data Essay

Athanasios Tsiamis SN:5223652

March 2022

1 Introduction

In the digital era we are living in, the amount of data people are using is increasing exponentially. According to Statista Research Department [1] the total amount of data created and replicated in 2020 reached a new high of just over 64 zettabytes. To put this into perspective, from the beginning of the human race to the year 2003, it is estimated [9] that 5 exabytes of data are created which is only 0.5% of a zettabyte! Data varies not only in size but also in forms and speed. These forms may include various means such as photos, videos or even random pieces of byte strings. Meanwhile, as the Internet of Things (IoT) is gaining a lot more traction, more and more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. However, as vast all of this data is, it provides no real value in its raw form. Even with cutting edge cloud computer farms, we are still unable to process these huge, unrefined amounts of data without some sort of preprocessing or analysis first.

Problem statement. Given this fairly large set of data, can it be analyzed in such a way that accurate results can be extracted from either the whole data set or an as good as possible sample? And if we take a sample, does the sample size matter?

2 Frequent item set mining

Before we delve deeper into addressing the problems posed before, it would be wise to set some preliminaries for easier understanding of the topics that follow. This will be done with the help of an example. Consider an online movie streaming platform that has various movies from different genres. The platform , as one might expect, wants to learn more about which movies are watched consecutively so it can provide better recommendations to its customers.

Given a set of items (i.e. an *itemset*) I=$i_1, ..., i_n$, and in our case movies, a transaction t is a subset of I and a database D is a set of those transactions. In our example, a transaction t would be a sequence of movies a viewer watched (e.g. "Godfather I" then "Godfather II" and then the "Goodfellas") while a database D would contain various of these transactions alongside a unique id to make them apart.

Although the streaming platform surely does contain a lot of different kinds of movies, it would make sense if a viewer has some preference to a certain genre. For example, there is one customer that likes superhero films and is clearly not into autobiography movies. In this case even though "Batman" and "Bohemian Rhapsody" may be the two most watched films in the platform it wouldn't make sense to recommend the latter film to this person, as there are not a lot people who actually follow through with that choice. Therefore, a framework is needed that can somehow mine the patterns that occur more frequently than a desired threshold. This occurrence in the database will be denoted from now on as *support*.

Formally, this type of problem is known as *Frequent Itemset Mining* (*FI Mining*); Given a transaction database D over a set of items $\mathcal{I}$, find all itemsets that are frequent in D given the minimal support threshold θ.

A first attempt at tackling this problem would be to simply check if every possible combination of itemsets is frequent or not. However, this brute force method is simply not feasible. For example, if we can check 1024 sets/sec then even for a very small streaming platform with just 100 movies we need more than 4×10^{18} years which exceeds the age of the universe!

Fortunately, this obstacle can be surpassed with the *Apriori Algorithm* that makes use of the *Apriori property*. Formally, the Apriori property is defined as:

$$\mathcal{I}_1 \subseteq \mathcal{I}_2 \Rightarrow \text{supp}(I_1) \geq \text{supp}(I_2)$$

Intuitively, reverting back to the movie platform example, let's consider the Rocky franchise which (as of today) consists of nine movies (denoted by R_i, $i = 1, ..., 9$). Obviously, watching Rocky I and then Rocky V is a subset of binge watching the entire collection of Rocky Movies ($R_1R_5 \subset R_1..R_5..R_9$). However, we expect that the amount of people who have watched just those two films (in that particular order) to be at least as many as those films connoisseurs that have watched the entire series ($supp(R_1R_5) \geq supp(R_1R_2...R_9)$). What is more, is that if $Y \subset X$ and $supp(X) < t_1$ then $supp(Y) < t_1$. This means that a set X is a candidate frequent set, if and only if, all its subsets are frequent. In that way, the Apriori Algorithm can search from smaller to bigger subsets for frequent itemsets (i.e. *levelwise*) and it can stop early if a subset is found not to be frequent.

However, even with the Apriori algorithm being faster than the brute force method, in order to check how frequent an itemset set is, we would have to scan the whole database to find out. In our online platform example that could potentially mean checking millions of rows of data just to see if a particular movie is frequent or not. Unsurprisingly, this kind of process is time and space consuming and therefore, largely impractical. Instead, mining for frequent itemsets from a sample is far more efficient. However, since we are using a sample, we may end up eventually finding itemsets that are frequent in the sample but not in the whole database and vice versa. Evidently, the probability of such errors relies on the sample size. More about this sample size will be discussed in section 4.1 and 4.2.

So, let p be the probability that a random transaction among all the transactions in the database supports itemset Z. For example, if Z was in our movie example R_1R_3 then let p be the probability that a random subscriber among the movies he/she saw was Rocky I and then (possibly after many movies) Rocky III. What we're essentially doing is that we 're using Z as an indicator function; mathematically speaking[1]:

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

Therefore, given a transaction database D and an itemset Z we have managed to create a function Z: $D \longrightarrow \{0,1\}$ or, in other words, a *classifier* to dictate whether Z is supported or not.

This function indicates a shift to our original problem; from finding the ideal sample size for efficient frequent sample itemset mining to finding the ideal sample size for a classification problem. The latter process is called *PAC-Learning* and will be further discussed in section 3.

Finally, the notion of classifying a set X with the set of a classifiers $\mathcal{H}$ can be formalised as a *range space*. A range space (X, R) consists of a finite or infinite set of points X and a finite or infinite family R of subsets of X which is called *ranges*.

3 PAC-learning

Probably Approximately Correct (PAC) learning constitutes one of the cornerstones in the Machine Learning Field. In the following section a more thorough explanation of this framework will be given which was originally proposed by Vapnik & Chervonenkis ([10]) and Leslie Valiant ([8]).

In short, in this framework, the learning unit receives examples(e.g. data points) and must select a generalization function (called hypothesis) that accurately classifies them from a greater set of possible functions(i.e. a function family). The goal is that with high probability (the *"Probably"* part) the function picked by the framework will be able to accurately predict outcome values for the data (the *"Approximately Correct"* part).

In the final section the *No-Free-Lunch* Theorem is presented, which addresses the question of finding the best machine learning algorithm out there to solve any kind of problem.

3.1 Classification, Quality and convergence

Given a data set X of particular interest, we can safely assume that there is a distribution $\mathcal{D}$ that governs X. Ideally, we would like to somehow learn this distribution. However, as this may not be practical in a real world application, due to the sheer amount of data, we may have to resort to other methods. Therefore, a more feasible alternative would be to "learn" a (computable) function $h : X \to Y$ (called *hypothesis*) such that it classifies data points from X with labels of set Y. Such process, computing wise, would be far more efficient than finding the underlying distribution. However, this newly introduced definition on hypothesis function is too "liberal" and may as well be very much off of the true function f that classifies the data accurately. As a result, there is an obvious need to "judge" how well h performs.

A first (and hasty) attempt at this would be to compare the classification results of h to the results of the true function f that governs the data with the help of probability theory. Thus, the loss of using h rather than f can be formally defined as the probability $\mathbb{P}_{x \sim D}[h(x) \neq f(x)]$ or, informally, the probability that h makes a mistake while classifying. As said before, this kind of validation on h is considered too crude, as it depends on both the underlying distribution and the ideal true function f, that categorizes perfectly the data, both of which are obviously not known.

A more fine-grained attempt at evaluating the hypothesis function is measuring how well it performs on a given database sample D.

It's important here to notice that in the sampling process all the $x_i \in D$ have been sampled *independently* and *identically* distributed (IID) according to $\mathcal{D}$. This means that each random variable (in our case entries in the database) has the same probability distribution as the others and the occurrence of one does not affect the occurrence of others (i.e. mutually independent). For brevity, we state the *IID* assumption for a data set with m elements as : $D \sim \mathcal{D}^m$ in which $\mathcal{D}^m$ is the distribution over the m-tuples induced by $\mathcal{D}$.

So, the empirical error $L_D(h)$ based on a sample D

[1]For brevity the $\mathbb{1}_Z$ was omitted and Z was used instead.

is defined as:

$$L_D(h) = \frac{|\{(x_i, y_i) \in D \mid h(x_i) \neq y_i\}|}{|D|} \quad (1)$$

The second approach, while it most certainly solves the problems mentioned before, it heavily depends on a good sample D. To elaborate further on this, let's define a hypothesis class $\mathcal{H}$ as a set of hypotheses. While it surely is possible to include infinitely many hypotheses in the class $\mathcal{H}$ it would be wise to restrict to a finite hypothesis class and look for the hypothesis h that has minimal loss over it. Mathematically speaking, we choose h_D such that $h_D \in \arg\min_{h \in \mathcal{H}}$.

This is known as *Empirical Risk Minimization* (ERM). A simple way to implement ERM would be, for example, to use a *halving algorithm.* Such an algorithm, using a given sample D, iteratively drops hypotheses which make mistakes while classifying, until it finds the one with minimal loss.

However, as said above, the formula (1) is dependent on the sample D and the choice of a good h_D. Therefore, the loss can be considered a random variable and the problem derived by formula (1) can be now thought of as: the probability to pick a data set out of the database, for which the loss is small enough. By convention, the probability of getting a non-representative sample (i.e. a sample with high loss) is denoted by the letter δ and the confidence of our prediction with $1 - \delta$.

Naturally, a misleading, non-representative sample may impact negatively our results. Since the term *"misleading sample"* can be somewhat abstract in its current form, it is properly formalised with the help of the accuracy parameter ϵ. Therefore, a sample D is "good" if the loss of an hypothesis on that sample is ϵ-close to its true loss under the distribution $\mathcal{D}$ ($|L_D(h) - L_{\mathcal{D}}(h)| \leq \epsilon$) and it is "bad" if the opposite inequality applies ($|L_D(h) - L_{\mathcal{D}}(h)| > \epsilon$).

To further elaborate how a misleading sample may have adverse effects, let $\mathcal{H}_{\mathcal{B}}$ be the set of bad hypotheses, $\mathcal{H}_{\mathcal{B}} = \{h \in \mathcal{H} \mid L_{\mathcal{D},f}(h) > \epsilon\}$. Additionally, let M be the set of a misleading sample that teaches us a bad hypothesis, $M = \{D \mid \exists\, h \in \mathcal{H}_{\mathcal{B}} : L_D(h) = 0\}$. Assume that there is a hypothesis h which makes no mistakes while classifying on the sample, that is $L_D(h_D) = 0$. So, $L_{\mathcal{D},f}(h) > \epsilon$ would mean that the hypothesis h_D makes way too many mistakes while trying to classify the general data set $\mathcal{D}$ or, equivalently, the sample is misleading. Using mathematical notation, this means that $\{D \mid L_{\mathcal{D},f}(h) > \epsilon\} \subset M$.

Rewriting M as a union of sets in $\mathcal{H}_{\mathcal{B}}$ and using the union bound, the Hoeffding inequality and some basic mathematical analysis, one can prove that:

$$\mathcal{D}^m(\{D|L_{\mathcal{D},f}(h) > \epsilon\}) \leq |\mathcal{H}_{\mathcal{B}}|e^{-\epsilon m} \leq |\mathcal{H}|e^{-\epsilon m} \quad (2)$$

Setting the right side in (2) less or equal than δ and with some basic algebra we can prove that:

$$m \geq \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \quad (3)$$

Thus, by choosing a sample m bigger than this fraction described above, we can force the probability of a bad sample to be small. Evidently, this quantity which depends on $\mathcal{H}$ dictates the size of iid samples needed. Since there are infinitely many function families $\mathcal{H}$, it would be smart though, to look for the one that uses the minimal amount of those samples. This function, which is denoted by $m_{\mathcal{H}}$, is known as the *sample complexity.*

What's more, a data set D is called ϵ-representative if the sample is "good" for every hypothesis in the hypothesis class $\mathcal{H}$ ($\forall\, h \in \mathcal{H} : |L_D(h) - L_{\mathcal{D}}(h)| \leq \epsilon$). Epsilon representative samples are especially important in the sense that they can establish PAC-Learning if somehow the presence of them is always guaranteed.

Luckily, this guarantee is given by the uniform convergence property! A hypothesis class $\mathcal{H}$ has the uniform convergence property (with respect to domain Z) if there exists a sample complexity function $m_{\mathcal{H}}^{UC}$ such that $\forall\, (\epsilon, \delta) \in (0,1)^2$ and for any distribution $\mathcal{D}$ on Z, if D is an i.i.d. sample according to $\mathcal{D}$ over Z of size m bigger than $m_{\mathcal{H}}^{UC}$, then D is ϵ-representative with probability at least $1 - \delta$.

3.2 Realizable vs agnostic PAC-learning

As mentioned before, PAC learning is an extremely important notion which characterizes whether a hypothesis class is learnable or not. PAC-learning comes in two flavours; *Realizable* and *Agnostic.*

Realizable PAC-Learning makes use of the *realizability assumption* which states that the true hypothesis is in $\mathcal{H}$. In other words, there exists one hypothesis h^* that classifies perfectly the data and makes no mistakes ($L_{\mathcal{D},f}(h^*) = 0$).

Therefore, using the definitions from before we can now state the definition of *realizable PAC Learning*: A hypothesis class $\mathcal{H}$ is Realizable PAC Learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm A such that $\forall\, \epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X}$ and for every $f : \mathcal{X} \to \{0,1\}$, if the realizability assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then, when running A on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d samples generated by $\mathcal{D}$ and labelled by f, A returns a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta : L_{\mathcal{D},f}(h) \leq \epsilon$.

Informally, Realizable PAC Learning sets some rules on the classification problems. Namely, if a sample is of sufficient size and one of the hypotheses classifies perfectly the data then reasonable results are expected by the algorithm.

In section 3.1 we established the minimum sample size required so the probability of a bad sample is small. While trying to reach the desired outcome, we essentially used the realizability assumption ($L_D(h_D) = 0$) to come up to the final result described in equation (3). Hence, now that we have the knowledge of realizable PAC-Learning, we can see that every finite hypothesis $\mathcal{H}$ is PAC learnable with sample complexity:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon} \rceil \tag{4}$$

However, assuming that we have found an assumption that has zero loss is rather unrealistic. The realizabilty assumption , even though it greatly simplifies the calculations , it leaves no space for outliers, experimental errors or other noise that may impact negatively results . Therefore, in the more general case, we have but to drop this assumption and instead resort to probability theory to accurately classify the data points. This means reverting back to a probability distribution $\mathcal{D}$ on $X \times Y$ i.e. $\mathcal{D} = \mathcal{D}|_X \times \mathcal{D}|_{Y|X}$. Then, for a binary classification problem $X \times \{0, 1\} \sim \mathcal{D}$, the labelling classifier function is defined in a probabilistic way:

$$f_{\mathcal{D}} = \begin{cases} 1 & \text{, if } \mathbb{P}(y = 1|x) \geq \frac{1}{2} \\ 0 & , otherwise \end{cases}$$

Additionally, up until now, the main goal was the minimization of the number of errors no matter their kind. Even though this is a nice thing to achieve, it shouldn't be regarded as an end in itself. For example, imagine a problem where a machine learning task is classifying tumors as malignant or not. Needless to say, finding a classification function that reduces the number of both false negatives (i.e. a malignant tumor is classified as benign) and false positives (vice versa) would be ideal. However, classifying some tumors as malignant when they are actually benign is not that big of a deal as the patients can just repeat the examination and find out.

Thus, in order to address the importance of error types, let Z be stochastic variable with distribution D. A loss function l makes use of this stochastic variable $l : Z \times \mathcal{H} \to \mathbb{R}_+$ so as to give "significance" to some specific outcomes. The loss of hypothesis $h \in \mathcal{H}$ is now defined as the expected value of the loss $L_{\mathcal{D}}(h) = E_{z \sim \mathcal{D}}\ l(z, h)$.

These two points pave the way for *agnostic PAC learning.*

A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $l : Z \times \mathcal{H} \to \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm A with the property that: for every $\epsilon, \delta \in (0, 1)$ and for every distribution $\mathcal{D}$ over Z, when running A on $m_{\mathcal{H}}(\epsilon, \delta)$ iid samples generated by $\mathcal{D}$, A returns a result which is close to the best possible result. More specifically, A returns a hypothesis h $\in \mathcal{H}$ such that with probability at least $1 - \delta$, $L_{\mathcal{D}} \leq \min_{h^* \in \mathcal{H}} L_{\mathcal{D}}(h^*) + \epsilon$.

As for the sample complexity in the agnostic case, it can be proven, through a similar process as in section 3.1, that for a general loss function $l : Z \to [a, b]$ we have:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \lceil \frac{2(b-a)^2 ln(2|\mathcal{H}|/\delta)}{\epsilon^2} \rceil \tag{5}$$

Comparing the inequalities 4 and 5 we can see that the biggest difference is the ϵ. The denominator goes from ϵ to ϵ^2 when going from the realizable to the agnostic case, which means that the minimal sample size grows by a factor of $1/\epsilon$. For example if ϵ is 0.01 then the minimal sample size has to be 100 times bigger! Thus, it is self evident that imposing the realizabilty assumption reduces significantly the minimum sample size required.

3.3 VC-dimension and the fundamental theorem

While in section 3.1 no restrictions were imposed on Y, which is the codomain of function h, it would be wise if we started simply and restrict ourselves to binary classifications {0,1}. For example, such binary classification would be the tumor classification task described before. Let C be a (finite) subset of h's domain set X ($C \subset X$). The restriction of $\mathcal{H}$ to C (denoted as $\mathcal{H}_{\mathcal{C}}$) is the set of all functions that can be derived from $\mathcal{H}$. C is *shattered* by $\mathcal{H}$ if $\mathcal{H}_{\mathcal{C}}$ is the set of all functions from C to {0,1}.

Having established those, we can now come up to the notion of *Vapnik-Chervonekis* (VC) *dimension.* Formally, let $S = (X, R)$ be a range space, then the VC-dimension of S is the maximum cardinality of a shattered subset of X. If there are arbitrarily large subsets then $VC(S) = \infty$.

On a more informal note, imagine a binary classification task as the one given above classifying tumors as malignant or not and n data points that represent those tumors in some way (e.g. by length and width). Since there are 2 possible labellings for each data point, there are 2^n possible labellings in total. For each of these labellings, if we can take a function from our function family that fully separates the malignant classified tumors from the benign ones, then the set of n points is *shattered* by our family of functions. The maximum n, which this can happen, is the VC-dimension of our function family($\mathcal{H}_{\mathcal{C}}$).

Hence, we can now conclude this section with a theorem that ties all these results together; *the Fun-*

damental Theorem of PAC-Learning. Let $\mathcal{H}$ be a hypothesis class of functions from X to $\{0,1\}$ and let the loss function be the 0-1 (binary) loss. Then the following statements are equivalent:
1. $\mathcal{H}$ has the uniform convergence property.
2. ERM is a successful PAC learner of $\mathcal{H}$.
3. $\mathcal{H}$ is PAC learnable.
4. The VC-dimension of $\mathcal{H}$ is finite.

Equivalence of 2. and 4. are especially important for the section that follows (3.4).

3.4 Bias and no free lunch

It seems as if this ERM rule of finding the hypothesis that has the minimum loss of all hypotheses, is the panacea for all our classification problems! No matter what the problem is, we could allegedly use an (absurdly) large, but finite, hypothesis class $\mathcal{H}$ and solve the problem at hand.

But before we confirm (or debunk) such statement, let's think of the loss as a composite of 2 factors; approximation and estimation error ($L_{\mathcal{D}}(h_D) = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon_{est}$). Since we are dealing with big data, and thus working with samples is inevitable as explained in section 2, estimation error can't be avoided. What could be avoided, though, is the approximation error which measures how well the hypothesis class $\mathcal{H}$ fits the distribution. Therefore, if for any problem, no matter what it had to with, we chose a big enough $\mathcal{H}$ then we could allegedly enforce the realizability assumption and apply the ERM rule!

Unfortunately (or fortunately!), all of these would be true if it wasn't for the *No Free Lunch Theorem* (NFL). The NFL Theorem (loosely speaking) states that for every learning algorithm there are some cases in which the algorithm will behave poorly. Choosing an appropriate algorithm, that performs well consistently, requires making assumptions about the data generating process. Without assumptions, no "superior" algorithm, performs better than any other one.

The NFL has proven to be a controversial topic for the scientific community. While some academics claim that the NFL makes some very interesting points, others, such as Giraud-Carrier and Provost, argue that "[..] the NFL theorem is of little relevance to research in the machine learning" ([11],[2]).

4 Frequent item set mining on big data

The task of mining frequent items sets on large databases are fundamental techniques in data mining and database applications. There are various algorithms for this problem which vary in their approaches and their results([3], [6]). In this essay, however, the focus will be on 2 important papers that greatly influenced the academic community: Toivonen's [7] and Riondato and Upfal's [5]. In the end a comparison of their results will be given.

4.1 Toivonen and frequent item sets

Hannu Toivonen ([7]) was one of the pioneers in finding frequent item sets efficiently in very large databases. He presented an algorithm that, by mining a random sample on the data set, creates a candidate set of FIs that contains all the FIs in the database with a probability relative to the sample size. It is by no means guaranteed that that all the items in the candidate set are frequent and also that all of the frequent itemsets are also in the candidate set. However, in those rare cases the latter occurs, the missing items can be found in a second pass of the database. Toivonen also proposed a bound on the sample size to ensure that the frequencies of the itemsets in the sample are not far off of their real values.

Thus, he proved that for $\epsilon, \delta > 0$ a sample of size bigger than $\frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$ is needed for an epsilon close approximation of the frequent itemsets.

For example, for a δ and an ϵ value of 0.001 and 0.01 respectively, a sample size of at least 50000 is required, which is pretty reasonable given our thresholds. If the error requirements were stricter, then the sample size would be quite substantial.

However, removing the assumption that the database can be infinitely large then a sample size bigger than $\frac{1}{2\epsilon^2} \ln \frac{2|I|}{\delta}$ is needed. The proof of this bound uses the Hoeffding and the Union bound. This sample size can be potentially large, given that the amount of itemsets could be many.

4.2 PAC-learning and frequent item sets

Riondato and Upfal used the aforementioned VC-dimension in section 3.3 to develop a technique that provided high quality approximations by mining a small random sample of the data set [5]. They also constructed bounds for the sample size needed to obtain ϵ close approximations to the collection of frequent itemsets. This was done through a concept called the *d-index.* To further elaborate on that, let D be a data set, then the *d-index* is defined as the maximum integer that D contains at least d different transactions of length d such that no one of them is a subset of another. The last subordinate clause is also known as an *antichain.*

For example, consider the data set D=$\{\{a,b,c,d\},\{a,b,d\},\{a,c\},\{d\}\}$ consisting of 4 transactions built upon the set of items $\mathcal{I} = \{a,b,c,d\}$. The d-index of D is 2 as $\{a,b,d\}$ and $\{a,c\}$ form an antichain. It can be easily checked that a d-index of 3 or 4 is not possible in this particular example.

So using those definitions and some set theory, they proved the following theorem: There exists a data set

D with d-index d, and a corresponding range space that has VC-dimension exactly d.

However, due to the antichain requirement, computing the d-index can be quite slow as it requires multiples passes of the database. Hence, a new algorithm was proposed to efficiently compute an upper bound q to the d-index; namely the *d-bound*. By definition, a subset satisfying the theorem mentioned previously should also satisfy the same theorem where the d-bound is substituted for the d-index since d-index $\leq$ d-bound. The advantage is that the computation of d-bound can be performed with a single linear scan which makes it really easy to update it when new transactions are added in the database.

As for the sample size, the authors showed that given a data set D, d being the d-bound, $\epsilon, \delta \in (0, 1)$ and c a constant then for an absolute epsilon-close approximation of the frequent itemset of the database, a sample of at least $\min\{|D|, \frac{c}{\epsilon^2}(d + \log \frac{1}{\delta}\}$ is required. Note that c is an absolute positive constant that does not depend on the range space or any other parameter. Löffler and Phillips showed experimentally that this constant is at most 0.5 [4].

4.3 Toivonen vs PAC-learning

Toivonen(section 4.1) and Riondato & Upfal (section 4.2) are 2 different approaches for frequent item set mining in databases. In a way, Riondato & Upfal constitute an improvement in Toivonen's works by circumventing major drawbacks in his algorithm. More specifically, the major drawback in Toivonen's bound was that the sample bound depends linearly on the number of individual items appearing in the data set (i.e. $|I|$). Riondato & Upfal showed that the sample size needed to obtain an epsilon-close approximation is independent from the number of transactions in the database. Not only that but also, it is always at most as large as the d-bound (or d-index), which is always less or equal to the longest transaction in the data set by definition. This in turn is less or equal to the number of individual items $|\mathcal{I}|$ which shows the clear advantage of their work over Toivonen's.

References

[1] Statista Research Department. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. `https://www.statista.com/statistics/871513/worldwide-data-created/`, 2022. [Online; accessed 05-April-2022].

[2] Christophe Giraud-Carrier and Foster Provost. Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper? *Proceedings of the ICML-2005 Workshop on Meta-learning*, 01 2005.

[3] Yanrong Li and Raj P. Gopalan. Effective sampling for mining association rules. In Geoffrey I. Webb and Xinghuo Yu, editors, *AI 2004: Advances in Artificial Intelligence, 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, December 4-6, 2004, Proceedings*, volume 3339 of *Lecture Notes in Computer Science*, pages 391–401. Springer, 2004.

[4] Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions. *CoRR*, abs/0812.2967, 2008.

[5] Eli Riondato, Matteo; Upfal. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. *ACM Transactions on Knowledge Discovery from Data vol. 8 iss. 4*, 8, aug 2014.

[6] Tobias Scheffer and Stefan Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. 3(null):833–862, mar 2003.

[7] Hannu Toivonen. Sampling large databases for association rules. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, page 134–145, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[8] L. G. Valiant. A theory of the learnable. *Communications of the ACM vol. 27 iss. 11*, 27, nov 1984.

[9] Jeff Vance. Big data analytics overview. `https://www.datamation.com/applications/big-data-analytics-overview/`, 2013. [Online; accessed 05-April-2022].

[10] A. Ya. Vapnik, V. N.; Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability Its Applications vol. 16 iss. 2*, 16, jan 1971.

[11] Darrell Whitley and Jean Watson. *Complexity Theory and the No Free Lunch Theorem*, pages 317–339. 01 1970.