UNIVERSITE D'ORLEANS
Faculté des Sciences
LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

Sampling the evaluation of relational patterns

Agnès Braud, Teddy Turmeaux
Université d'Orléans, LIFO

Rapport N^o **2004-08**

Sampling the evaluation of relational patterns

Agnès Braud Teddy Turmeaux

LIFO - Université d'Orléans,
B.P. 6759, 45067 Orléans cedex 2, France
{agnes.braud,teddy.turmeaux}@lifo.univ-orleans.fr

Résumé

Scalability is a critical problem in relational Data Mining. This is mainly due to the evaluation of many patterns. Tractability can be achieved by decreasing either the number of patterns evaluated, or the cost of their evaluation. We investigate the second case, and more precisely the use of data sampling.

Each pattern aims to express a piece of knowledge on some entities of interest, that have to be counted to evaluate the pattern. Learning in first order logic often implies a multi-instance setting that induces two levels in the data : entities and instances. We compare different strategies according to the level at which sampling is done. We show the importance of the strategy chosen in order to achieve a better estimate of the counting and thus to enable the use of sampling in relational Data Mining. We also propose a family of estimators for our framework together with some bounds on the accuracy obtained.

1 Introduction

Scalability is a critical problem in Data Mining, and especially in a relational framework. Data Mining algorithms search for patterns hidden in databases, that involve one or several objects. The aim is to find a piece of knowledge on a particular class of objects. This class appears one time in the pattern and we call it the entity of interest, that is the one on which we want to learn something. Other objects can appear in the pattern. The piece of knowledge is expressed as relations between the objects in the pattern and constraints on their features. The pattern is divided into two parts : a relational part which sets the links between the objects involved in the pattern and a selective part which sets the constraints on the object features.

Let us consider, for example, a database with two relations *molecule* and *atom* corresponding to two classes of objects and another relation *bond* expressing a link between two atoms. If we are interested in learning a piece of knowledge on molecules, then *molecule* is the entity of interest. A classification task may consist in learning a description of a certain concept concerning the molecules, let say organic, and a pattern $\mathcal{P}$ could be, expressed in natural language, "a molecule is organic if it has one atom of carbon and one atom of hydrogen". This would be represented in datalog (see for example [6]) by the clause
$organic(M) : -molecule(M), atom(A_1, M, c), atom(A_2, M, h)$

where variable symbols start with a capital, c is the constant representing carbon and h the one representing hydrogen.

The pattern scheme, that is the set of relations appearing in it, is $organic(M_1) : -molecule(M_2), atom(A_1, M_3, E_1), atom(A_2, M_4, E_2)$.

The relational part of $\mathcal{P}$ sets that : $M_1 = M_2 = M_3 = M_4$.

The selective part of $\mathcal{P}$ sets that : $E_1 = c$ and $E_2 = h$.

Most of the algorithms follow a process that alternates steps during which candidate patterns are generated with steps during which they are evaluated according to a given criterion. It is thus necessary to evaluate each of the patterns with respect to that criterion. The criterion is based on statistic measures, which require to evaluate how well the pattern fits the data and thus a counting among the entities of interest of the ones that satisfy that criterion. That counting is realized with conjunctive queries [7]. Those counting queries are generated from the pattern to evaluate. For our previous example, the evaluation of $\mathcal{P}$ requires to count the organic molecules appearing in the answer of the following conjunctive query

$? - molecule(M), atom(A_1, M, c), atom(A_2, M, h)$.

When other relations than the one of the entity of interest appear in the query (as in the one above), the relational part may have several instances for each entity of interest. The instances of each entity form a subdatabase. The problem when counting entities in the answer of the query is thus that of counting distinct entity identifiers among the instances satifying the selective part of the query. The cost of the evaluation increases with the amount of data and is even more complex when the pattern is relational because of the multi-instance setting.

Among the Data Mining tasks lying within this framework, let us cite in addition to the induction of Horn clauses [20] and the search for query extensions [12], the induction of regression trees or of first-order classification trees [18, 5].

Tractability of the mining process can be achieved by decreasing either the number of patterns evaluated, or the cost of the evaluation of a pattern. Both of those approaches may decrease the quality of the output of the algorithm so that the strategy used to carry them out has to be elaborated carefully.

In this paper, we investigate the second approach. More precisely, we are interested in decreasing the cost of counting distinct entities satisfying some given constraints (the constraints specified by the pattern).

Several approaches have been proposed in the Data Mining, Machine Learning, Database or Statistics fields. We can distinguish different kind of strategies (some are reported in [4]) :

- reduction of conjunctive queries, that is rewriting them so that the new ones are equivalent but simpler to evaluate. This includes minimization [11], semantic simplification [17] and query transformations [10],
- taking into account only a particular category of queries, namely acyclic conjunctive queries [16],
- reduction of the data [2], that is building and then working on a synopsis of the data like a sample, a histogram or wavelets,
- shared computations : some computations are often repeated, it can be interesting to save some results and to reuse them rather than having to compute them again. This includes materialization of features [9], treatment of queries sharing intermediate results by packs [3] or precomputation of statistics on data [13].

We are interested in an approach combining data synopses and precomputations. More precisely, we propose to adopt an approach in which data samples are computed and stored to be reused for the evaluation.

Sampling has the advantage of being simple, which makes it easy to adapt, robust and suitable for all kinds of data. Moreover, it has also the advantage of not requiring *a priori* assumptions on the data distribution. Let us notice that when using sampling, one does not obtain the exact result. Nevertheless, a certain margin of imprecision is acceptable at certain stages of the Data Mining process, especially at the beginning since it aims at exploring the space of patterns.

In this paper we focus on the problem of choosing a suitable strategy for sampling the data and then extrapolating the result using an estimator. Several sampling strategies can be adopted. We propose a classification according to the level at which sampling is performed (entities or instances). We compare the precision of the estimation obtained when making different parameters vary.

As far as we know, sampling has not been studied much in the relational learning framework. Srinivasan [22] considers two sampling techniques (subsampling (see section 3) and windowing that takes into account examples with erroneous predictions). Sebag and Rouveirol [19] have defined stochastic matching, but this work is not directly comparable to ours in that the technique is used to generate the pattern and not to evaluate it.

This paper is organized as follows. In Section 2, we describe an approach for the evaluation process. Then Section 3 presents different sampling strategies classified according to the level at which sampling is done. A selection of results obtained from experiments led on artificial data is reported in Section 4 and conclusions are given in Section 5.

2 Evaluation process

The evaluation of a query using our sampling approach is composed of three steps :

- extract a sample from the data corresponding to the relational part of the query,
- count in the sample distinct entities satisfying the selective part of the query,
- extrapolate (estimate) the result.

The two difficult steps are the first and the third ones. For the first one, one has to generate a sample that is as representative of the initial one as possible.

For the extrapolation, one uses functions called *estimators* (see [15] for a description of some classic estimators). Estimators are dependent on the sampling strategy adopted so they have to be designed according to it.

Generating a sample of the instances for each evaluation can be time-consuming. A solution is proposed in [1], which consists in precomputing and storing a sample of instances for the possible joins of $\mathcal{P}$. This approach is feasible because the number of joins is limited by the database schema (foreign keys). We extend this approach by exploiting specificities of Data Mining, for which constraints on the form of patterns (specified by biases) still reduce the number of joins.

Evaluation is then realized as follows.

2.1 Initialization

For each join potentially appearing in a query, we compute and calculate a sample of the instances.

Let us consider for example two molecules m_1 and m_2, three atoms $a1$, $a2$ and $a3$ of m_1 whose instances in *atom* are $atom(a_1, m_1, c)$, $atom(a_2, m_1, h)$ and $atom(a_3, m_1, c)$, and two atoms $a4$ and $a5$ of $m2$ with instances $atom(a_4, m_2, c)$ and $atom(a_5, m_2, c)$ in *atom*.

The result of the query

$$? - molecule(M), atom(A_1, M, c), atom(A_2, M, h)$$

is contained in the result of the join query

$$? - molecule(M), atom(A_1, M, E_1), atom(A_2, M, E_2)$$

namely

M	A_1	E_1	A_2	E_2
m_1	a_1	c	a_1	c
m_1	a_1	c	a_2	h
m_1	a_1	c	a_3	c
m_1	a_2	h	a_1	c
m_1	a_2	h	a_2	h
m_1	a_2	h	a_3	c
m_1	a_3	c	a_1	c
m_1	a_3	c	a_2	h
m_1	a_3	c	a_3	c
m_2	a_4	c	a_4	c
m_2	a_4	c	a_5	c
m_2	a_5	c	a_4	c
m_2	a_5	c	a_5	c

In that result, only the four instances for which E_1 is set to c and E_2 to h satisfy the selective part of the query, and the result of the counting is 1. Indeed, for an entity to be counted, at least one of its instances has to be in the answer of the query.

We propose to collect a sample of the result of the query without selection in order to estimate the result of the counting. The initialization step thus consists in building those samples.

Several strategies may be adopted. We propose a classification of those strategies with respect to the level at which sampling is done (entities or instances).

2.2 Query evaluation

For each query to evaluate :
– choose the sample corresponding to the query joins,
– count the entities having at least one instance satisfying the query conditions,
– estimate the result of the query on the whole data set.

Let us notice that several queries can be evaluated using the same sample, namely those sharing the same joins.

In the following, we call *true instance* for a query an instance that belongs to the result of that query. For example, the instance

M	A_1	E_1	A_2	E_2
$m1$	$a3$	c	$a2$	h

is true for the query
$? - molecule(M), atom(A_1, M, c), atom(A_2, M, h)$

We also say that an entity is true if it has at least one true instance. For example m_1 is a true entity in the example.

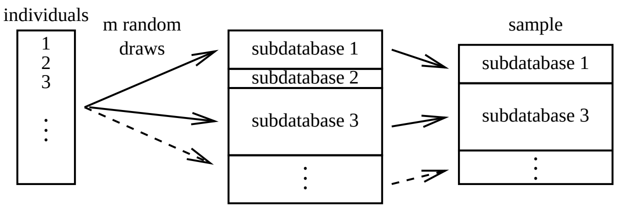

FIG. 1 – Sampling at the entity level

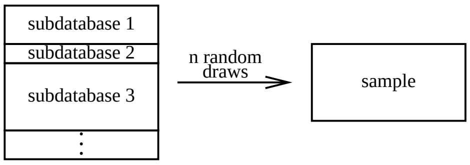

FIG. 2 – Sampling at the instance level

3 Sampling strategies, estimation and error guarantees

We propose a classification of the sampling strategies according to the level at which sampling is done (entities or instances). This leads to three classes of sampling strategies that we describe in this section.

Let us first recall that in the case of random sampling, also called probabilistic sampling, all the elements of the population have an equal probability of being in the sample. This can be done with or without replacement. For the estimation, better results are obtained with a sampling without replacement. For further details on random sampling and especially on how to realize it in a relational database, see [21].

3.1 Sampling at the entity level

A random sampling is performed among entities, then the sample is built by including all the instances of the selected entities (see figure 1). It is called subsampling in [22].

Let N be the number of entities in the initial set, n the number of entities in the sample (that is having at least one instance in the sample) and d the number of instances belonging to the resultset of the query.

An unbiased estimator of D, number of entities from the initial data set matching the query, is given by

$$\hat{D} = d\frac{N}{n}.$$

We can be at least $(1-\alpha)100\%$ confident that the error in estimating D will be less than a specified amount ϵ when $n \geq \frac{z^2_{\alpha/2}}{4\epsilon^2}$, where $z_{\alpha/2}$ is the value of the standard normal distribution leaving an area of $\alpha/2$ to the right [23].

This sampling strategy has the drawback of not allowing the control of the sample size. Indeed, all the instances kept are included in the sample, even if they are numerous.

3.2 Sampling at the instance level

The sample is obtained by performing a random sampling among all the instances of all the entities (see figure 2).

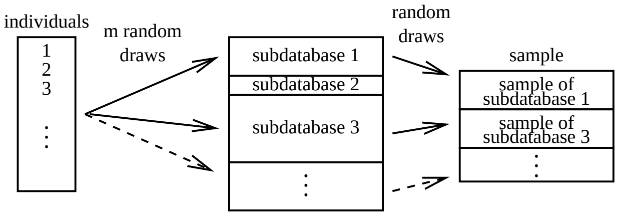

FIG. 3 – Two-level sampling

Let us notice that by applying the selection of a query Q on a random sample of the relational part of Q, we obtain a random sample of the instances that satisfy Q [21].

We can thus estimate the number of entities having a true instance from that sample.

Among the estimators possible to estimate the number of distinct values from a random sample of the data, let us cite the Chao estimator [15].

Let d be the number of true entities in the sample, d_i be the number of true entities having i instances in the resultset of the query executed on the sample. The Chao estimator is given by

$$\hat{D}_{Chao} = d + \frac{d_1^2}{2(d_2+1)} - \frac{d_1 d_2}{2(d_2+1)^2}.$$

The estimator tries to correct the result obtained on the sample by applying correcting factors on the number of entities having few entities in the sample.

It has been shown theoretically that this type of sampling is not suitable for our problem. Indeed, Charikar et al. [8] have proved that for a set of M instances, the error (see sect. 4.2 for the definition of the error) on the estimation of the number of entities D obtained from a random sample of size m is at least $error(\hat{D}) \geq \sqrt{\frac{M-m}{2m}} \ln(\frac{1}{p})$ for any estimator for a given confidence rate p. This shows that it is necessary to consider almost the whole set of instances to guarantee a low error, even with a confidence rate of 0.5.

3.3 Two-level sampling

Two-level sampling refers to strategies for which entities are sampled and then a sample of the selected entities is collected (see figure3). It is a solution for the problem inherent to sampling at the entity level since it restricts the size of the subsample linked to each entity.

We can use different strategies to build the samples of the second level (samples of the instances of an entity), such as taking a random sample or a sample reproducing the initial distribution of the instances among the entities kept.

There is a new difficulty for the estimation compared to the case of subsampling : since we do not have all the instances of an entity, we know for sure that an entity is true if it has a true instance, but if it has none, it does not mean that it is not a true instance. We thus define a family of estimators for the two-level sampling as follows.

The truth value X_i of an entity i is now a value between 0 and 1, with $X_i = 1(0)$ meaning that it is a true (false) entity, and is set to :

– 1 if the entity has at least one true instance in the sample,

– else $\begin{cases} \text{- 0 if the sample contains all the instances of} \\ \text{the entity (thus we are sure of the value since} \\ \text{ we have seen all the instances)} \\ \text{- x} \in [0,1] \text{ otherwise} \end{cases}$

x is an estimation of the value that can be computed based on different parameters : the proportion of instances of i in the sample (it is feasible to consider that this quantity is known : we do not need to compute the whole join to get it), known frequency of true entities in the sample, ...

We propose to use an estimator of this family that takes into account the two parameters above, considering that the less instances we consider, the more the estimation has to be corrected, and that this correction can be applied to the observed frequency of the property considered. This has led us to : $x = \frac{d}{n}.(1-\frac{k}{K_i})^2$, where k is the number of instances kept for an entity and K_i is the number of instances of i in the initial set. We call this estimator *EstimCorrected* in the following.

Let us notice that setting x to 0 in the third case amounts to apply the estimator designed for the case where all the instances of the entity are kept (subsampling), so that it underestimates the real value. We call this estimator *Estim*0 in the following.

An estimator of the number of true entities in the initial data set is then given by $$\hat{D} = (\Sigma_{i=1}^{n} X_i).\frac{N}{n}.$$

We obtain an immediate confidence interval for this family of estimators as follows.

For an entity, we define two values X_{min} and X_{max} which represent respectively a lower and an upper bound on the truth value of the entity.

X_{min} equals :
– 1 if the entity has at least one true instance in the sample,
– 0 otherwise.

X_{max} equals :
– 1 if the entity has at least one true instance in the sample,
– else $\begin{cases} \text{- 0 if the sample contains all the instances of} \\ \text{ the entity} \\ \text{- 1 otherwise} \end{cases}$

Let P be the proportion of true entities in a sample containing n entities, we have $P_{min} \leq P \leq P_{max}$
(with $P = \frac{\Sigma_n(X_i)}{n}, P_{min} = \frac{\Sigma_n(X_{i\,min})}{n}, P_{max} = \frac{\Sigma_n(X_{i\,max})}{n}$).

Let π be the proportion of true entities in the initial set and S^2 the variance ($S^2 = P(1-P)$).

When the sample is large enough, we know that with a probability $1-\alpha$, π is such that :

$P - z_{\alpha/2}\frac{S}{\sqrt{n}} \leq \pi \leq P + z_{\alpha/2}\frac{S}{\sqrt{n}}$.

We thus have

$P_{min} - z_{\alpha/2}\frac{S}{\sqrt{n}} \leq \pi \leq P_{max} + z_{\alpha/2}\frac{S}{\sqrt{n}}$

S is maximum for $P = 0.5$ and then

$P_{min} - z_{\alpha/2}\frac{\sqrt{0.25}}{\sqrt{n}} \leq \pi \leq P_{max} + z_{\alpha/2}\frac{\sqrt{0.25}}{\sqrt{n}}$

This gives us a first confidence interval for our estimators family and more generally for two-level sampling strategies. We give in appendix a way of determining the suitable values for the parameters of a sampling that keeps at most k instances, in order to reach an estimation error smaller than any ϵ with a probability larger than a given δ.

Let us also notice that when all the instances of the entities are considered, we have $P_{min} = P = P_{max}$. In this case, the confidence interval is that of the sampling at the entity level.

4 Experiments

We have performed tests on artificial data that we have generated so as to be able to represent the different aspects of the problem.

4.1 Data

For our study purposes, it is enough to generate data sets whose tuples are composed of an index representing the entity and a boolean which indicates if the tuple is a true or false instance. The following table presents an example with two entities, one having three instances (two true instances and one false), and the other one having one false instance.

Entity	Value
1	true
1	true
1	false
2	false

Two parameters are known to be of great impact on the efficiency of sampling : the data distribution and the selectivity of the query. In the case of data distribution, data with high skew (large variations in the value frequencies) are difficult to treat because one runs the risk of ignoring values which are less represented.

In order to test the behaviour of sampling strategies on different data distributions, we have generated the data using Zipfian distributions. The Zipf law enables to allocate a number of occurrences to each value of a set. Different distributions are obtained by modifying a parameter whose value varies between 0 (the number of occurrences has large variations between values) and 1 (uniform distribution). This law thus enables us to generate very different distributions for our data, including cases which are difficult for sampling.

Another factor of error is selectivity. Two selectivities can be defined :

- selectivity in terms of instances, defined as the ratio of the number of instances fitting the selective part of the query to the number of instances in the initial set. A low selectivity creates problems because one has less chances to have instances in the answer of the query which makes the extrapolation difficult,
- selectivity in terms of entities, which is the ratio of the number of entities having at least one true instance to the total number of entities.

To simulate the effect of selectivity, we generate data sets with various numbers of true instances per entity.

4.2 Parameters studied

We study the ratio error obtained on the value to estimate when making the following parameters varying :

- size m of the sample, that is the number of instances kept,
- data distribution,
- selectivity.

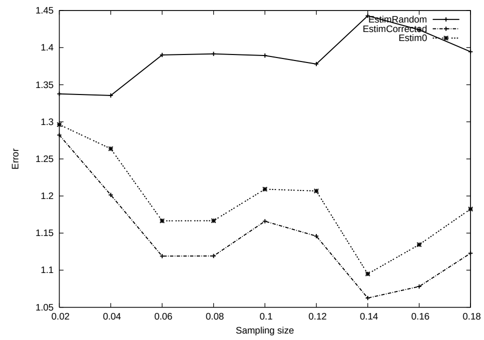

FIG. 4 – Influence of the sample size

The error on the estimation of D is defined as $error(\hat{D}) = (\frac{D}{\hat{D}}, \frac{\hat{D}}{D})$. It is obvious that the closer to 1 this ratio is, the better the estimation is.

4.3 Results

In this section we report some of the results we have obtained on two-level sampling.

For the sampling at the second level, we have chosen to use a strategy which consists in keeping at most k instances for an entity, where k is an additional parameter of the sample. This induces a fair allocation of instances among entities and prevents any from not being enough represented compared to the others. The same choice has been made by [14], but in contrary to that latter approach, we use at the most the space available for the sample. This is due to the fact that [14] impose a very strong constraint of building the sample in only one scan of the data.

The number of entities represented in the sample is thus at least $\frac{m}{k}$.

Each error reported is the average of 10 trials. The initial set contains 256000 instances corresponding to 1024 entities. k, maximal number of instances considered for each entity is set to 200. We are in the worst case where each true entity has only one true instance.

Figure 4 shows the results obtained when the sample size varies from 2% of the number of instances of the initial set to 18%. Results are presented for $Estim0$, $EstimCorrected$ and $EstimRandom$ which is the estimator obtained by setting x to a random value.

The better results are obtained by $EstimCorrected$. Let us notice that the curves are not monotonous because only k instances are kept per entity.

Figure 5 then presents the results obtained when the selectivity in terms of entity varies from 10% to 90%.

Both $Estim0$ and $EstimCorrected$ are stable and gives a good estimation, but $EstimCorrected$ still acts better.

On figures 6, 7 and 8, we make the percentage of true instances per true entity vary. The most difficult cases are when this percentage is low since it becomes more difficult to have a true instance in the sample. Figure 6 shows a comparison between $Estim0$ and $EstimCorrected$. We can see that $EstimCorrected$ is always better than $Estim0$.

Figures 7 and 8 show the minimum and maximum values found by the two estimators among the ten trials. They show that $EstimCorrected$ tends to be a bit more stable than $Estim0$.

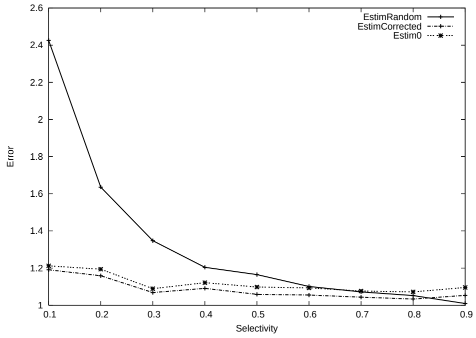

FIG. 5 – Influence of the selectivity

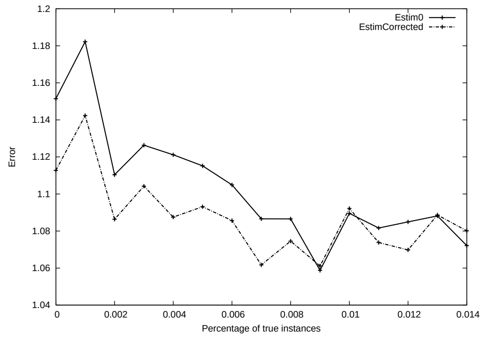

FIG. 6 – Effect of the percentage of true instances per true entity

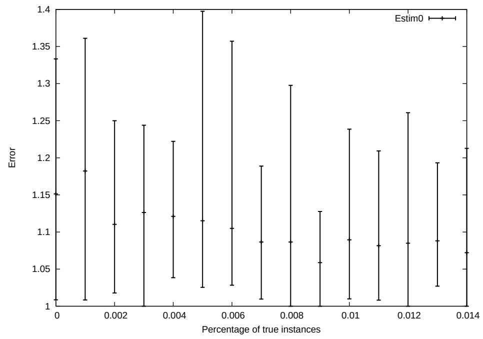

FIG. 7 – Error interval for *Estim*0

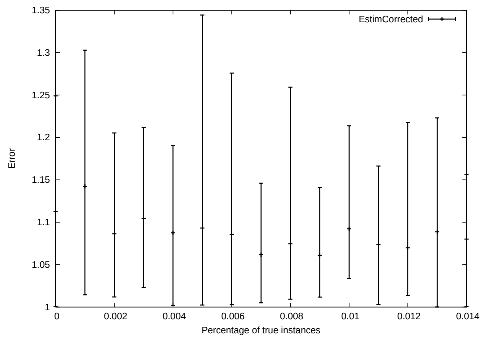

FIG. 8 – Error interval for *EstimCorrected*

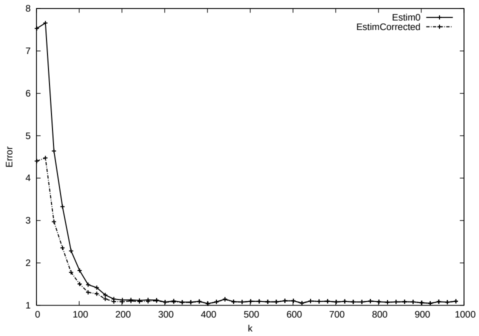

Fig. 9 – Influence of the number of instances kept

The estimator $EstimCorrected$ acts more on the negative effects coming from a low percentage of instances, which is normal considering the parameters we have used to define the correcting factor.

The estimator we propose thus seem to be quite efficient to correct the estimation error made when few instances of an entity are considered.

On figure 9, we make the number k of instances kept per entity vary.

We can see that under a certain threshold the estimation is unusable. We thus have to set carefully the value for k (see appendix).

5 Conclusions

We have described different sampling methods that can be used when evaluating relational patterns generated during a Data Mining process. The aim is to decrease the cost of this operation, by estimating the number of entities satisfying the conditions of the pattern. Those countings are parameters of the statistic measures used to evaluate the quality of the patterns. We have classified those sampling approaches according to the level at which sampling is performed (entities or instances).

It has been proved theoretically that sampling at the instance level does not lead to good results, what we have checked empirically. On the other hand, sampling at the entity level does not allow the control of the sample size (the number of instances). An alternative is two-level sampling, in which sampling acts at the entity level and then at the level of the instances of the selected entities. The aim is to obtain a more fair allocation of space among the selected entities in the sample. This approach enables to obtain very good estimations even with a very small sample. We have defined a family of estimators for two-level sampling, based on a schema that we can instantiate to take into account different parameters. Experiments have shown that it allows to adjust the estimation. Moreover, we have presented some theoretical results on error guarantees for two-level sampling.

We are now planning to study in more details the family of estimators, notably by studying the elaboration of the correcting factor, and to test our estimators on real data.

Finally two-level sampling combined with precomputation of samples appears as a good way to deal with the problem of scalability in relational Data Mining.

Références

[1] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2) :275–? ?, 1999.

[2] Daniel Barbará, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Theodore Johnson, Raymond T. Ng, Viswanath Poosala, Kenneth A. Ross, and Kenneth C. Sevcik. The New Jersey data reduction report. *IEEE Data Engineering Bulletin*, 20(4) :3–45, December 1997.

[3] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. mon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16 :135–166, 2002.

[4] H. Blockeel and M. Sebag. Scalability and efficiency in multi-relational data mining. *SIGKDD Explor. Newsl.*, 5(1) :17–30, 2003.

[5] Hendrik Blockeel and Luc De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2) :285–297, June 1998.

[6] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1(1) :146–166, 1989.

[7] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[8] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. Towards estimation error guarantees for distinct values. In *Proc. of PODS-00*, pages 268–279, 2000.

[9] Weidong Chen and David S. Warren. Tabled evaluation with delaying for general logic programs. *Journal of the ACM*, 43(1) :20–74, 1996.

[10] V. Costa, A. Srinivasan, R. Camacho, H. Blockeel, B. Demoen, G. Janssens, J. Struyf, H. Vandecasteele, and W. Van Laer. Query transformations for improving the efficiency of ilp systems, 2002.

[11] Vítor Santos Costa, Ashwin Srinivasan, and Rui Camacho. A note on two simple transformations for improving the efficiency of an ILP system. *LNCS*, 1866 :225–242, 2000.

[12] Luc Dehaspe. *Frequent Pattern Discovery in First-Order Logic*. PhD thesis, 1998.

[13] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proc. of the 16th IJCAI*, pages 1300–1309. Morgan Kaufmann Publishers Inc., 1999.

[14] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB 2001, Italy*, pages 541–550, 2001.

[15] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Lynne Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. of VLDB'95*, pages 311–322, 11–15 September 1995.

[16] T. Horvàth and S. Wrobel. Toward discovery of deep and wide first-order structures a case study in the domain of mutagenicity. pages 51–64, September 2001.

[17] Jonathan J. King. *Query Optimization by Semantic Reasoning*. Un.Michigan press, ; ACM CR 8609- ? ?, 1984.

[18] S. Kramer. Structural regression trees. In *Proc. of AAAI-96*, pages 812–819. AAAI Press/MIT Press, 1996.

[19] M. Sebag and C. Rouveirol. Any-time relational reasoning : Resource-bounded induction and deduction through stochastic matching. *Machine Learning*, 38(1/2) :41–62, 2000.

[20] Stephen Muggleton and Luc De Raedt. Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19/20 :629–679, 1994.

[21] Frank Olken. *Random Sampling from Databases.* PhD thesis, University of California, April 1993.

[22] A. Srinivasan. A study of two sampling methods for analysing large datasets with ILP. *Data Mining and Knowledge Discovery*, 3(1) :95–123, 1999.

[23] R.E. Walpole and R.H. Myers. *Probability and Statistics for Engineers and Scientists*. Collier Macmillan, 2 edition, 1978.

A Bounds

Let π be the proportion of true entities in the whole population of m entities, let π_0 be the proportion of true entities in the same population when only a random sample of at most k instances are considered and let P_0 be the estimation of π_0 from a random sample of size n of the m entities.

P_0 versus π_0.
Chernoff asserts that
$P(|P_0 - \pi_0| \leq \epsilon_2) \geq 1 - 2e^{-2n\epsilon_2^2} (= \delta_2)$
We can compute n, the number of entities considered, to achieve any ϵ_2 with probability δ_2 : $n = \frac{1}{2\epsilon_2^2} ln(\frac{2}{\delta_2})$

π_0 versus π.
Now to find some bounds, we suppose that all the true entities have at least a proportion f of true instances. So, when considering a sample of size k of the instances, the probability that a sample of a true entity does not contain a true instance is lower that $(1 - f)^k$. We call p_k this probability.

To compute the difference between π and π_0, we use the following result about Bernouilli trials : let S_n be the number of successes on n trials, that is : $P(S_n = k) = b(n, k, p)$

then we have for example the following bound : $P(S_n \geq r) \leq \frac{r(1-p)}{(r-np)^2}$ if $r > np$.

Now remembering that p_k is the probability of failing to detect a true entity from a sample of k of its instances, then the number e of errors for the m entities is such that (with $r = \epsilon_1 m$)

$$\begin{aligned} P(e \geq \epsilon_1.m) &\leq \frac{\epsilon_1 m(1 - p_k)}{(\epsilon_1 m - mp_k)^2} \\ &\leq \frac{\epsilon_1(1 - p_k}{m(\epsilon_1 - p_k)^2} \end{aligned}$$

That is $P(e \leq \epsilon_1.m) \geq \frac{\epsilon_1(1-p_k)}{m(\epsilon_1-p_k)^2}$.

Thus finding p_k such that $\delta_1 = \frac{\epsilon_1(1-p_k)}{m(\epsilon_1-p_k)^2}$ ensures that $P(|\pi_0 - \pi| \leq \epsilon_1) \geq \delta_1$ and ϵ_1 can be taken as low as desired by raising k

Similar work can be done when $(r < np)$.

Mixing the bound.

Finally we have

$P(|P_0 - \pi| \leq \epsilon_1 + \epsilon_2 \cap |\pi_0 - P_0| \leq \epsilon_2)$
$= P(|\pi_0 - P_0| \leq \epsilon_2).P(|P_0 - \pi| \leq \epsilon_1 + \epsilon_2| \, (|\pi_0 - P_0| \leq \epsilon_2))$
$\geq P(|\pi_0 - P_0| \leq \epsilon_2).P(|\pi_0 - \pi| \leq \epsilon_1)$
$\geq \delta_2.\delta_1$

Remembering that $P(A) \geq P(A \cap B)$, we have
$P(|P_0 - \pi| \leq \epsilon_1 + \epsilon_2) \geq \delta_2.\delta_1$

So for each $\epsilon (= \epsilon_1 + \epsilon_2)$ with probability $\delta (= \delta_1.\delta_2)$, we can determine parameters such that : $P(|\pi - P_0| \leq \epsilon) \geq \delta$.