

# Out-Of-Bag Discriminative Graph Mining

Andreas Maunz  
Institute for Physics  
Hermann-Herder-Str. 3  
79194 Freiburg, Germany  
andreas@maunz.de

David Vorgrimmler  
in-silico Toxicology  
Altkircherstr. 4  
4054 Basel, Switzerland  
vorgrimmler@in-silico.ch

Christoph Helma  
in-silico Toxicology  
Altkircherstr. 4  
4054 Basel, Switzerland  
helma@in-silico.ch

## ABSTRACT

In class-labeled graph databases, each graph is associated with one from a finite set of target classes. This induces associations between subgraphs occurring in these graphs, and the target classes. The subgraphs with strong target class associations are called discriminative subgraphs. In this work, discriminative subgraphs are repeatedly mined on bootstrap samples of a graph database in order to estimate the subgraph associations precisely. This is done by recording the subgraph frequencies (support values) per class over the out-of-bag instances of the bootstrap process. We investigate two different methods for the approximation of the true underlying support values from these empirical values, involving sample mean and maximum likelihood estimation. We show that both methods significantly improve the process, compared to single runs of discriminative graph mining, by applying the different methods to publicly available, chemical datasets. In computational models of toxicology, subgraphs (fragments of chemical structure) are routinely used to describe toxicological properties of molecules. Apart from the subgraph associations being statistically validated, the subgraph sets created by the proposed methods are also small, compared to ordinary graph mining, and may thus be beneficial for statistical models, as well as for inspection by toxicological experts.

## Categories and Subject Descriptors

H.2.8 [Database Applications]: Data Mining, Statistical Databases

## General Terms

Theory, Experimentation, Performance

## 1. INTRODUCTION

Given a class-labeled graph database, discriminative graph mining is a supervised learning task with the goal to extract graph fragments (subgraphs) with strong associations to the

classes, according to statistical constraints set by the user. For example, the subgraphs may be molecular fragments that induce toxicity. Indeed, finding such subgraphs is a major goal in toxicology [8]. However, discriminative graph mining yields large result sets, even for very tight constraints on subgraph significance values ( $p$ -values) [7]. Moreover, it is an unstable process, i.e. slight changes in the sample may lead to substantially different subgraph sets.

A bagged predictor consists of several aggregate predictors, each trained on a dedicated bootstrap sample of the training data. Bagged predictors have the potential for considerably higher predictive accuracy, compared to single predictors, especially in the context of unstable prediction methods [2]. Bagged predictor estimates may be obtained from the out-of-bag instances, which saves computational effort compared to crossvalidation on the bootstrap samples. For example, out-of-bag estimation of node errors in decision trees could improve on estimates obtained from the training data as a whole [2].

This work extends discriminative subgraph mining by out-of-bag estimation for subgraph frequencies (support values) on the target classes that govern the training data, and by aggregating subgraphs in the output that pass the statistical constraints. Infrequently occurring subgraphs are filtered out, which removes the instability to a large extent. The result is a set of statistically validated, highly significant, discriminative subgraphs, which may be useful for diagnostic or predictive modelling, or for expert inspection.

The remainder of this work is structured as follows: We present related work with a focus on discriminative graph mining (section 2), our proposed methods for out-of-bag estimation of support values and  $p$ -values, as well as algorithmic implementation (section 3). Experiments include validation of support values and  $p$ -values on publicly available databases of various sizes and numbers of target classes (section 4). We draw conclusions in section 5. The contributions of this work are summarized as follows:

- Repeated discriminative graph mining on bootstrap samples of a class-labeled graph database is proposed as a means to stabilize estimates for subgraph properties, such as support values per class, compared to single runs of discriminative graph mining. The estimation is performed using the out-of-bag instances of the individual bootstrap samples.
- Two methods for the estimation of support values are described, where both handle multiple classes. Significance tests are applied to these estimated values and insignificant subgraphs are removed from the results.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'13 March 18-22, 2013, Coimbra, Portugal.

Copyright 2013 ACM 978-1-4503-1656-9/13/03 ...\$10.00.

- The estimated support values, predicted  $p$ -values, and predicted class associations are empirically validated on molecular databases of various sizes. The results indicate significant improvements over ordinary discriminative graph mining, where the effect is generally larger the smaller the datasets are. We conclude that out-of-bag estimation has the potential to generate concise, highly discriminative subgraph sets for use in statistical models and/or expert inspection.

## 2. RELATED WORK

Out-of-bag methods have been used to robustly estimate node probabilities and node error rates in decision trees [2] as well as the generalization error and accuracy of bagged predictors. In the work by Bylander [4], generalization error was well modeled by out-of-bag estimation, and its already small bias could be further reduced by a correction method, where, in order to correct the prediction of a given instance, similar out-of-bag instances with the same target class were employed. However, the method of out-of-bag estimation is not confined to these examples, and may be used to also estimate other statistical properties in supervised learning.

Discriminative subgraph mining is often employed as a preprocessing step to statistical learning, because such subgraphs may be useful as descriptors [3]. Nowadays, there is a variety of well-known statistical learning algorithms available “off the shelf”, which makes a workflow attractive where subgraphs are extracted from the data, represented in a unified format, and fed into a machine learning algorithm [6]. Usually, discriminative subgraphs are mined using a user-defined significance threshold. A subgraph is in the result, if it passes a statistical test with regard to its association to the target classes. However, graph mining often produces huge descriptor sets even with very tight bounds on discriminative power, which would prevent machine learning methods from obtaining models in acceptable time. Thus, post-processing would be required to lower redundancy and eliminate the vast majority of subgraphs [1, 7].

Subgraph boosting [10] is an integrated method that employs subsampling internally, alternating between graph mining and model building. The method presented here is clearly different from boosting, because it calculates descriptors that can be used in wide variety of models afterwards. It is similar in that it calculates a small collection of discriminative subgraphs. However, it is also stable against perturbations of the dataset, which is not the case in boosting.

## 3. METHODS

### 3.1 Basic Graph Theory

A graph database is a tuple  $(G, \Sigma, a)$ , where  $G$  is a set of graphs,  $\Sigma \neq \emptyset$  is a totally ordered set of labels and  $a : G \rightarrow C$ ,  $C \subset \mathbb{N}$ , is a function that assigns one from a finite set of class values to every graph in the database. The set of target classes  $C$  consists of at least two values. We consider labeled, undirected graphs, i.e. tuples  $g = (V, E, \Sigma, \lambda)$ , where  $V \neq \emptyset$  is a finite set of nodes and  $E \subseteq V = \{\{v_1, v_2\} \in \{V \times V\}, v_1 \neq v_2\}$  is a set of edges and  $\lambda : V \cup E \rightarrow \Sigma$  is a label function. An ordered set of nodes  $\{v_1, \dots, v_m\}$  is a *path* between  $v_1$  and  $v_m$ , if  $\{v_i, v_{i+1}\} \in E, i \in \{1, \dots, m-1\}$  and  $v_i \neq v_j$  for all  $i, j \in \{1, \dots, m\}$ . We only consider connected

	$g$	$all$
class 1	$k_1$	$ G^1 $
class 2	$k_2$	$ G^2 $
...	...	...
class $ I $	$k_{ I }$	$ G^{ I } $
$\Sigma$	$k$	$ G $

Table 1: Contingency table for subgraph  $g$ .

graphs here, i.e. there is a path between each two nodes in the graph.

A graph  $g' = (V', E', \Sigma', \lambda')$  subgraph-isomorphic to  $g$  if  $V' \subseteq V$  and  $E' \subseteq E$  with  $V' \neq \emptyset$  and  $E' \neq \emptyset$ ,  $\lambda'(v_1) = \lambda(v_2)$  whenever  $v_1 = v_2$ , and  $\lambda'(e_1) = \lambda(e_2)$  whenever  $e_1 = e_2$ , for all nodes and edges in  $g'$ . Then, graph  $g'$  is also referred to as a subgraph of  $g$  – we also say that  $g'$  covers  $g$ . The subset of the database instances  $G$  that  $g'$  covers is referred to as the occurrences of  $g'$ , and its size as support of  $g'$ . As a special case, the size of the subset of occurrences with  $a(g) = i$ , for any  $g$  in the occurrences, is referred to as the support of  $g'$  for class  $i$ . Thus, any subgraph has associated support values per class, ranging each between 0 and the support of  $g'$ , and summing up to the the support of  $g'$ .

The subgraphs considered in this work are free subtrees. Here, we define a tree as a graph with exactly  $n - 1$  edges that connect its  $n$  nodes. A free tree is a tree without a designated root node. For an introduction to tree mining and the terminology used there, see the overview by Chi *et al.* [5].

### 3.2 Significance Test

For a given subgraph  $g$ , we seek a  $|I| \times 2$  contingency table that lists the support values per class in the first column and the overall distribution of target classes in the second column, as in Table 1. This data serves to check whether  $g$ ’s support values differ significantly from the overall class distribution. The  $\chi_d^2$  function for distribution testing, defined as

$$\chi_d^2(x, y) = \sum_{i=1}^{|I|} \frac{(k_i - E(k_i))^2}{E(k_i)}, \quad (1)$$

where  $E(k_i) = \frac{G^i k}{|G|}$  is the expected value of  $k_i$ , calculates the sum of squares of deviations from the expected support for all target classes. The function value is then compared against the  $\chi^2$  distribution function to obtain a  $p$ -value and conduct a significance test with  $|I| - 1$  degrees of freedom. The next sections discuss methods to obtain the  $k_i$  entries in the table from the support values obtained during bootstrapping (section 3.3). The  $|G^i|$  entries are constants and take the values of the overall  $|G^i|$ . This is possible due to the stratified bootstrapping approach, which maintains the overall class proportions in each sample (see section 3.3). Additionally, the bias of  $g$  is determined as  $\arg \max_i (\frac{k_i}{k} / \frac{G^i}{G})$ , to designate the dominant class for  $g$ .

### 3.3 Out-Of-Bag Discriminative Graph Mining

We refer to the subset that contains all the graphs  $g \in G$  with  $a(g) = i$  as  $G^i$ . The procedure first splits the graph database randomly into equal-sized training and test databases  $G_{Train}$  and  $G_{Test}$ . Subsequently, stratified bootstrapping is performed on the training data, such that each

sample comprises  $|G_{Train}^i|$  graphs associated with class  $i$ , drawn with replacement and uniform probability inside each class  $i$ . On average, about 37% ( $1/e$ ) of training instances will not be drawn in any bootstrap sample (out-of-bag instances). Next, discriminative graph mining mines subgraphs on the drawn instances, according to minimum support and significance thresholds. To test each subgraph against the thresholds, it calculates support and  $p$ -values from the bootstrap sample. Then, it re-assesses the support values by performing isomorphism tests with the subgraphs on the out-of-bag instances ("matching"), and stores this information.

After the initial split, bootstrapping and graph mining is repeated  $N$  times on  $G_{Train}$ , where in each iteration, pairs of subgraphs and out-of-bag support values per class ( $g, k_1, \dots, k_{|I|}$ ) are produced, meaning that subgraph  $g$  occurs in  $k_i$  out-of-bag graphs associated with class  $i$ . The results are recorded over the  $N$  bootstrap samples, such that for each  $g$ , the list of support values is a tuple  $(\mathbf{k}_1 \dots \mathbf{k}_{|I|})$ , where  $\mathbf{k}_i$  is a vector  $(k_i^1 \dots k_i^N)$ , containing all the support values. The total support is determined from the class specific support values by summing up vectors  $\mathbf{k}_i$  across target classes:  $\mathbf{k} = \sum_{i=1}^{|I|} \mathbf{k}_i$ . The vectors  $\mathbf{k}_i$  are generally sparse (contain missing values), due to the instability of discriminative graph mining, i.e. perturbations to the dataset (such as bootstrap sampling) yield almost always a different (but partially overlapping) selection of subgraphs. To cope with the variety of rare subgraphs, a fixed threshold removes all subgraphs with less than  $\lfloor 0.3 * N \rfloor$  entries in the list of support values, after the end of bootstrapping.

Finally, support values are estimated for each remaining subgraph from the recorded results. Two methods are employed (described in the next sections), based either on the sample mean support per class, and using data of the current subgraph only, or on a maximum likelihood estimate, involving some of the other subgraphs. Then, the significance test from section 3.2 is run on the estimated support values.

### 3.3.1 Sample Mean Method

We set the value of  $k_i$  in Table 1 to  $\bar{\mathbf{k}}_i$  for all  $i \in \{1, \dots, |I|\}$ , i.e. the sample mean across the entries of vector  $\mathbf{k}_i$ , ignoring missing values. The value of  $k$  is the sum of the  $k_i$ .

### 3.3.2 Maximum Likelihood Estimation Method

Here, we employ some of the other subgraphs to form estimates for the  $k_i$ . The first step in this process is to extract the subgraphs with the same class bias as  $g$  (see section 3.2). Local ties are broken in favor of the dominant global class. In case of a further tie on the global level, one of the globally dominant classes is chosen with uniform probability. In a second step, the subgraphs with the same class bias as  $g$  are used to correct  $g$ 's local frequencies by weighting. This approach has some similarity to the work by Bylander [4], however, his aim is to correct instance predictions, and his correction employs similar out-of-bag instances, whereas our correction happens across bootstraps, and on the subgraphs (not instances) obtained collectively from all the bootstrap samples. For each class, we model the event that each  $k_i^j \in \mathbf{k}_i$  would occur for each of the subgraphs with the same class bias as  $g$  as a multinomial selection process. More specifically, we determine the class probabilities for each subgraph  $g'$  with the same class bias as

$g$  with a maximum likelihood estimator. It is the smoothed vector of relative class specific support values, defined as:

$$\alpha_{\mathbf{g}'} = \left( \frac{1 + |\mathbf{k}_1|_1}{|I| + |\mathbf{k}|_1}, \dots, \frac{1 + |\mathbf{k}_{|I|}|_1}{|I| + |\mathbf{k}|_1} \right) \quad (2)$$

where the  $\mathbf{k}_i$  and  $\mathbf{k}$  pertain to  $g'$ , and  $|\cdot|_1$  is the one-norm (the sum of the vector elements ignoring missing values). Following that, for each non-missing tuple  $(k_1^j, \dots, k_{|I|}^j)$  pertaining to  $g$ , a probability distribution is determined from this collection of multinomials:

$$p((k_1^j, \dots, k_{|I|}^j)) = \frac{\sum_{g'} p((k_1^j, \dots, k_{|I|}^j); \alpha_{\mathbf{g}'})}{\sum_{g'} 1} \quad (3)$$

Finally, the  $k_i^j$  values pertaining to  $g$  are corrected in a weighted average based on this probability distribution:

$$\bar{\mathbf{k}}_i = \frac{\sum_j k_i^j p((k_1^j, \dots, k_{|I|}^j))}{\sum_j p((k_1^j, \dots, k_{|I|}^j))} \quad (4)$$

Again, we set the value of  $k_i$  in Table 1 to  $\bar{\mathbf{k}}_i$ .

## 3.4 Algorithm

The mining step from section 3.3 may be implemented using a discriminative graph mining algorithm of choice. Here, we employ backbone refinement class mining (BBRC) [9]. It achieves high compression by partitioning the subgraph search space into non-disjoint sets (classes) with the same longest paths (backbones) and selecting only one representative from each class. Inside each partition, the patterns are ordered by the subgraph relationship (see figure 1). The representative is chosen to minimize the  $p$ -value (section 3.2), unless no member passes the user-defined significance constraint, in which case no representative is chosen. The search is also bounded by minimum subgraph support, as described in section 3.3. In the BBRC output, two isomorphic subgraphs are always represented by the same string identifier. We employ SMARTS strings as identifiers, a kind of regular expression to encode molecular fragments as strings<sup>1</sup>. This approach allows to store out-of-bag support values  $\mathbf{k}_i$  in a hash structure, using SMARTS as keys. The algorithm using  $numBoots = N$  bootstrap samples is shown in Algorithm 1.

We consider the case  $numBoots > 1$  first. Line 4 creates an initially empty hash table to gather results from mining the bootstrap samples in line 7. The resulting subgraphs are matched on the out-of-bag instances in line 8, and the results are stored in the hash. On termination of the loop, each hash entry  $\mathbf{k}_i$  has at most  $N$  support values. Post-processing the results is very fast and incurs negligible overhead compared to the graph mining step. It consists of removing subgraphs that (line 11) were not generated often enough by the sampling process (have not enough entries in the hash table, as determined by  $minSamplingSupport$ ), or which (line 13) do not significantly deviate from the overall distribution of classes, as assessed by the  $\chi^2$  test (section 3.2), with contingency table calculated according to mean (section 3.3.1) or maximum likelihood estimation (section 3.3.2) method.

If  $numBoots = 1$ , the data (support value,  $p$ -value, and bias) is obtained from a single BBRC run on the full training data, without bootstrapping and out-of-bag estimation.

<sup>1</sup>see <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>

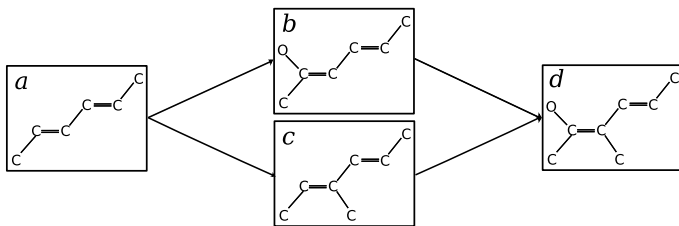


Figure 1: Two non-disjoint partitions  $(a, b, d)$  and  $(a, c, d)$ . The members are ordered by subgraph relationship.

**Data:** *dataBase, numBoots, minSamplingSupport, method, minFrequencyPerSample, alpha*

**Result:**  $[subgraphs, values]$

```

1 if numBoots = 1 then
2    $[subgraphs, values] \leftarrow BBRC(dataBase, minFrequencyPerSample, alpha);$ 
3 else
4    $hash \leftarrow \{\};$ 
5   for  $i := 1 \rightarrow numBoots$  (Parallel Processing) do
6      $sample, OOB \leftarrow drawBsSample(dataBase);$ 
7      $[subgraphs, values] \leftarrow BBRC(sample, minFrequencyPerSample, alpha);$ 
8      $insert(hash, match(subgraphs, OOB));$ 
9    $[subgraphs, values] \leftarrow [];$ 
10  for  $subgraph \in keys(hash)$  do
11    if  $length(hash[subgraph]) \geq minSamplingSupport$  then
12       $[candidateSupportValues, candidateBias] \leftarrow method(hash[subgraph]);$ 
13      if  $candidatePValue \leftarrow SignificanceTest(candidateSupportVals) < alpha$  then
14         $subgraphs \leftarrow subgraphs \cup subgraph;$ 
15         $values \leftarrow values \cup [candidateSupportValues, candidatePValue, candidateBias];$ 

```

Algorithm 1: Calculation of subgraph significance on out-of-bag instances

## 4. EXPERIMENTS

### 4.1 Setup

Three methods were compared by their ability to estimate the discriminative potential of subgraphs, by assessing the deviations between the class specific support values,  $p$ -values, and biases of subgraphs, as a) estimated by the respective method, and b) obtained by matching the subgraphs onto an independent test set. The methods compared are

1. MLE: Out-of-bag estimation with Algorithm 1, according to section 3.3.2, with  $numBoots = 100$ .
2. MEAN: Out-of-bag estimation with Algorithm 1, according to section 3.3.1, with  $numBoots = 100$ .
3. BBRC: The BBRC algorithm, with  $numBoots = 1$ .

The process was repeated 100 times for methods 1, 2, and 3. Five different error measures were assessed to compare the methods:

1.  $E_1$ , the mean of  $p_i^B - p_i^T$  over subgraphs, i.e. the bias of  $p$ -value errors.
2.  $E_2$ , the mean of  $|p_i^B - p_i^T|$  over subgraphs, i.e. the absolute  $p$ -values errors.
3.  $E_3$ , the mean of  $\left(\frac{1}{|I|} \sum_{i=1}^{|I|} \left| \frac{k_i^B}{k^B} - \frac{k_i^T}{k^T} \right| \right)$  over subgraphs, i.e. the relative support value errors.
4.  $E_4$ , 1 - the mean of  $\delta(bias_i^B, bias_i^T x)$  over subgraphs, i.e. the fraction with wrongly recognized class bias.
5.  $E_5$ , 1 - the mean of  $\delta(p_i^B \leq \alpha, p_i^T \leq \alpha)$  over subgraphs, i.e. the fraction wrongly recognized as significant.

The whole procedure is described in Algorithm 2. Line 1 initializes empty vectors that capture the residuals in estimation. Inside the main loop, stratified splitting (such that proportions of target classes inside each split equal overall proportions) generates a training and a test set of equal size. The training set is treated by the selected method, which returns a vector of subgraphs and a vector  $values^B$  of values, including support values,  $p$ -values, and bias (line 5). The subgraphs are matched on the test set, yielding analogous values  $values^T$  (line 6). Finally, the residual vectors capture the differences between  $values^B$  and  $values^T$  by the error measures  $E1 - E5$ .

Six molecular, class labeled datasets were used in the experiments. Four were drawn from the carcinogenic potency database (CPDB)<sup>2</sup>, namely “Combined Carcinogenicity and Mutagenicity” (MUL, 4 classes, 677 compounds), “Mouse Carcinogenicity” (MOU, 2 classes, 914 compounds), “Multi-Cell Call” (MCC, 2 classes, 1050 compounds), and “Rat Carcinogenicity” (RAT, 2 classes, 1128 compounds). MUL’s labels consist of the four cross-combinations of binary carcinogenicity and mutagenicity labels from the CPDB, for all chemicals with both values for both labels present. A rather small dataset, describing human intestinal absorption (INT, 3 classes, 458 compounds) [11], as well as the rather large Kazius/Bursi mutagenicity dataset (KAZ, 2 classes, 4069 compounds), [8] were also used.

### 4.2 Results

<sup>2</sup><http://potency.berkeley.edu/cpdb.html>

**Data:** *graphDatabase*, *method* (method is MLE, MEAN, or BBRC)  
**Result:**  $E_1, E_2, E_3, E_4, E_5$   
1  $E_1 = E_2 = E_3 = E_4 = E_5 = []$ ;  
2 **for**  $i := 1 \rightarrow 100$  **do**  
3    $[trainSet, testSet] \leftarrow splitStratified(graphDatabase, 0.5)$  ;  
4    $numBoots \leftarrow 100$ ; **if**  $method = BBRC$   $numBoots \leftarrow 1$ ;  
5    $[subgraphs, values^B] \leftarrow Algorithm\ 1(trainSet, numBoots, alpha = 0.05)$ ;  
6    $[subgraphs, values^T] \leftarrow match(subgraphs, testSet)$  ;  
7   **for**  $j := 1 \rightarrow 5$  **do**  
8      $E_j \leftarrow [E_j, errorJ(values^T, values^B)]$ ;

**Algorithm 2:** Calculation of error measures

Error Measure	MLE	MEAN
E1	<b>1</b> (2)	<b>3</b> (4)
E2	<b>0</b> (2)	<b>2</b> (4)
E3	<b>4</b> (6)	<b>0</b> (0)
E4	<b>0</b> (4)	<b>0</b> (2)
E5	<b>3</b> (5)	<b>0</b> (1)

**Figure 2:** Win-Loss ( $p=0.001$ )

Assay	Method	E1	E2	E3	E4	E5	Subgraphs
INT	MLE	-0.0107 (0.0610)	0.0414 (0.0487)	0.1086 (0.0365)	0.0029 (0.0137)	0.1626 (0.1948)	15.96
INT	MEAN	-0.0062 (0.0553)	0.0341 (0.0469)	0.1112 (0.0340)	0.0013 (0.0085)	0.1729 (0.1879)	18.36
INT	BBRC	0.0630 (0.0955)	0.0635 (0.0952)	0.1177 (0.0247)	0.1733 (0.1071)	0.4672 (0.1318)	103.53
MUL	MLE	-0.0129 (0.0154)	0.0188 (0.0132)	0.0610 (0.0200)	0.1457 (0.1723)	0.0806 (0.1405)	9.67
MUL	MEAN	-0.0098 (0.0101)	0.0155 (0.0104)	0.0694 (0.0239)	0.1576 (0.1655)	0.1040 (0.1392)	10.93
MUL	BBRC	0.0795 (0.0510)	0.0795 (0.0510)	0.1041 (0.0187)	0.4076 (0.1083)	0.5616 (0.1214)	94.95
MOU	MLE	0.0086 (0.0704)	0.0397 (0.0607)	0.0944 (0.0508)	0.0155 (0.0851)	0.2357 (0.2719)	4.51
MOU	MEAN	0.0161 (0.0593)	0.0353 (0.0526)	0.1050 (0.0504)	0.0132 (0.0716)	0.2976 (0.2761)	5.52
MOU	BBRC	0.1285 (0.1249)	0.1315 (0.1222)	0.1209 (0.0355)	0.1298 (0.1059)	0.5919 (0.1471)	48.17
MCC	MLE	0.0040 (0.0386)	0.0159 (0.0382)	0.0664 (0.0348)	0.0075 (0.0424)	0.1618 (0.1716)	12.71
MCC	MEAN	0.0062 (0.0337)	0.0194 (0.0327)	0.0791 (0.0373)	0.0097 (0.0389)	0.2089 (0.1743)	14.75
MCC	BBRC	0.0636 (0.0511)	0.0663 (0.0485)	0.0971 (0.0225)	0.0583 (0.0524)	0.5327 (0.1137)	79.47
RAT	MLE	-0.0028 (0.0125)	0.0084 (0.0116)	0.0573 (0.0286)	0.0022 (0.0141)	0.0936 (0.1435)	13.00
RAT	MEAN	-0.0017 (0.0115)	0.0092 (0.0109)	0.0659 (0.0317)	0.0101 (0.0365)	0.1280 (0.1439)	14.68
RAT	BBRC	0.0573 (0.0544)	0.0621 (0.0507)	0.0912 (0.0246)	0.0712 (0.0687)	0.5059 (0.1253)	70.01
KAZ	MLE	-0.0000 (0.0000)	0.0000 (0.0000)	0.0181 (0.0067)	0.0000 (0.0000)	0.0068 (0.0155)	26.09
KAZ	MEAN	-0.0000 (0.0000)	0.0000 (0.0000)	0.0182 (0.0069)	0.0000 (0.0000)	0.0052 (0.0140)	25.57
KAZ	BBRC	0.0000 (0.0000)	0.0000 (0.0000)	0.0181 (0.0066)	0.0015 (0.0075)	0.0106 (0.0217)	25.24

**Table 2:** Bias and accuracy

Table 2 details the results (mean values across bootstraps) Table 2 shows the win/loss statistics for MLE and MEAN.

Figure 3 plots the results.

We first consider E1, E2, and E3. For four of the six datasets (apart from SAL and KAZ), there is a clear improvement when out-of-bag estimation is applied (MLE or MEAN), compared to single runs of discriminative subgraph mining (BBRC). In all these cases, the differences are statistically significant at the  $\alpha = 0.025$  level. This means that here, the proposed out-of-bag estimation of support values improves significantly on mining subgraphs directly (BBRC), regardless of the specific method (MEAN or MLE). However, E3 is not significantly different for SAL and KAZ between BBRC and MEAN or MLE. Instead, since SAL’s E1 and E2 are significantly lower for BBRC, compared to MLE and MEAN, BBRC works better for this dataset. For KAZ, E1 and E2 are practically zero, and E3 is practically the same for all three methods.

Comparing the sampling methods, for the four “well-behaved” datasets, there is a draw concerning  $p$ -value bias (E1), which is, however, low (in absolute values  $< 0.02$ ). Concerning absolute  $p$ -value error (E2), MOU has a quite high value between 0.035 - 0.04 for both MLE and MEAN, which is only 3 - 4 times lower than that of BBRC. The others remain low, with no clear winner.

However, for E5, the error on significance estimation, MLE

performs significantly better than MEAN in all four cases. Interestingly, for E5, MLE and MEAN are both significantly better than BBRC. This means that, despite getting closer to the test values of support and  $p$ , it still judges a lot of subgraphs as significant at the 0.05 significance level, that are actually insignificant, or vice versa. E4, the estimation of class bias, is the easiest exercise. However, MLE and MEAN perform significantly better than BBRC for all datasets except KAZ. The effect is most drastic for the multi-class dataset MUL, but also remarkable for MOU and RAT.

Table 2 also gives the mean number of subgraphs generated in the last column. There is a general trend for the smaller numbers of subgraphs generated with the sampling methods with shrinking dataset sizes. This reflects the higher uncertainty due to the lack of data. However, as the line plots in Figure 3 show, there is a trend towards a higher gap in errors between the sampling methods on the one hand, and BBRC on the other hand with shrinking dataset sizes. It becomes also clear from these plots, that SAL behaves different than the other datasets (as discussed before).

## 5. CONCLUSION

Especially for small graph databases, which are typical in toxicology, the proposed methods can significantly improve on the estimation of statistical quantities, compared to ordi-

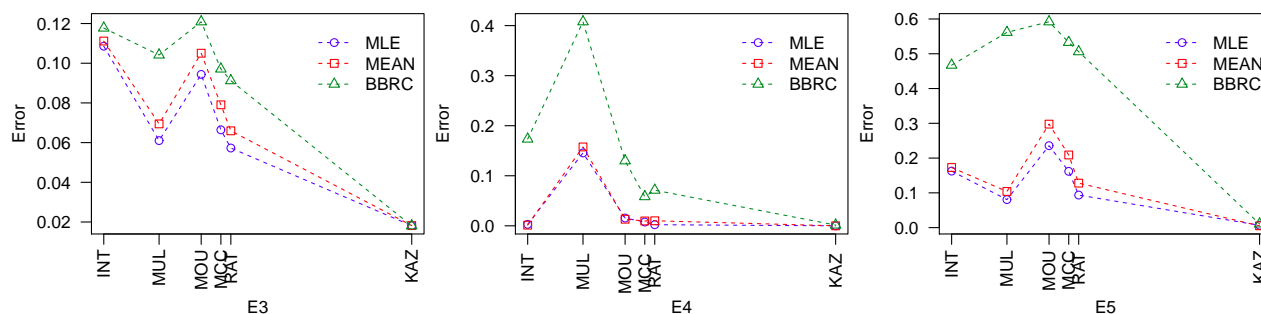


Figure 3: Error measures along increasing logarithmic dataset size

nary discriminative graph mining. For larger datasets, this advantage shrinks, which we attribute to increasing accuracy of ordinary discriminative graph mining in the presence of more data.

## 6. REFERENCES

- [1] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Jeremy Besson, and Mohammed J. Zaki. ORIGAMI: Mining Representative Orthogonal Graph Patterns. *ICDM 2007. Seventh IEEE International Conference on Data Mining*, pages 153–162, Oct. 2007.
- [2] Leo Breiman. Out-Of-Bag Estimation. *Technical Report, Statistics Department, University of California*, 1996.
- [3] Björn Bringmann, Siegfried Nijssen, and Albrecht Zimmermann. From Local Patterns to Classification Models. In Saso Džeroski, Bart Goethals, and Pance Panov, editors, *Inductive Databases and Constraint-Based Data Mining*, pages 127–154. Springer New York, 2010.
- [4] Tom Bylander. Estimating Generalization Error on Two-Class Datasets Using Out-of-Bag Estimates. *Machine Learning*, 48(1-3):287–297, 2002.
- [5] Yun Chi, Richard R. Muntz, Siegfried Nijssen, and Joost N. Kok. Frequent Subtree Mining - An Overview. *Fundamenta Informaticae*, 66(1-2):161–198, 2004.
- [6] Christoph Helma, Stefan Kramer, and Luc De Raedt. The Molecular Feature Miner MOLFEA. In *Proceedings of the Beilstein Workshop 2002: Molecular Informatics: Confronting Complexity*, 2003.
- [7] Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. SPIN: Mining Maximal Frequent Subgraphs From Graph Databases. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 581–586, New York, NY, USA, 2004. ACM.
- [8] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and Validation of Toxicophores for Mutagenicity Prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005.
- [9] Andreas Maunz, Christoph Helma, and Stefan Kramer. Efficient Mining for Structurally Diverse Subgraph Patterns in Large Molecular Databases. *Machine Learning*, 83:193–218, 2011.
- [10] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. gBoost: A Mathematical Programming Approach to Graph Classification and Regression. *Machine Learning*, 75(1):69–89, 2009.
- [11] Claudia Suenderhauf, Felix Hammann, Andreas Maunz, Christoph Helma, and Jörg Huwyler. Combinatorial QSAR Modeling of Human Intestinal Absorption. *Molecular Pharmaceutics*, 8(1):213–224, 2011.