# 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 classes, which induces associations between the latter and subgraphs occurring in the database graphs. The subgraphs with strong 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. This is done by recording the subgraph occurrence 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 toxicological datasets. In computational models of toxicology, subgraphs (fragments of chemical structure) are routinely used to describe molecules. Apart from the subgraph associations being statistically validated, the subgraph set sizes created by the proposed methods are much reduced, compared to ordinary discriminative 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 ($p$-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]. 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]. Bagged predictor estimates may be obtained from the out-of-bag instances, which saves computational effort compared to crossvalidation on the bootstrap samples.

This work extends discriminative subgraph mining by out-of-bag estimation for subgraph frequencies on the classes that govern the training data (support values), 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. This yields 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, as well as algorithmic implementation (section 3). Experiments include validation of subgraph support values, $p$-values, and class bias on publicly available databases of various sizes and numbers of 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 class values. Significance tests are applied to these estimated val-

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.

ues and insignificant subgraphs are removed from the results.

- The estimated support values, $p$-values, and 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 potential to generate concise, 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 class were employed. However, the method of out-of-bag estimation is not confined to these examples, and may be used to estimate other statistical properties in supervised learning.

Discriminative graph mining is often employed as a preprocessing step to statistical learning, because discriminative 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 from a graph database using a predefined threshold with regard to target class associations. A subgraph qualifies for the result set if it passes a corresponding significance test. However, such approaches tend to produce huge sets of very similar subgraphs, even with very tight bounds on discriminative power, which would prevent machine learning methods from building models in acceptable time, and post-processing would be required to lower redundancy and eliminate the vast majority of subgraphs [1, 7]. Moreover, the result is not stable with regards to perturbations of the database.

Subgraph boosting [10] is an integrated approach to build models on small sets of subgraphs, alternating between graph mining and model building. The method presented here is clearly different from boosting, because it calculates subgraphs independently from a specific model. It is similar in that it calculates a small collection of discriminative subgraphs, but it is also stable against perturbations of the database, which is generally not the case for 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 $a : G \to I$, $I \subset \mathbb{N}$, is a function that assigns one from a finite set of class values to every graph in the database. The set of target classes $I$ 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 $\lambda : V \cup E \to \Sigma$ is a label function. An ordered set of nodes $\{v_1, \ldots, v_m\}$ is a *path* between $v_1$ and $v_m$, if $\{v_i, v_{i+1}\} \in E, i \in \{1, \ldots, m-1\}$ and $v_i \neq v_j$ for all $i, j \in \{1, \ldots, m\}$. We only consider connected 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 (total) 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 $\chi^2_d$ function for distribution testing, defined as

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

**Table 1: Contingency table for subgraph $g$.**

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

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 $|G^i|$ 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 $g$. The next sections discuss methods to obtain the $k_i$ entries in the table from the support values obtained during bootstrapping (section 3.3).

## 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 into stratified, equal-sized training and test databases $G_{Train}$ and $G_{Test}$. Subsequently, stratified bootstrapping is performed on $G_{Train}$, such that each sample comprises $|G^i_{Train}|$ graphs associated with class $i$, drawn with replacement and uniform probability inside class $i$, for all $i$ in $I$. On average, about 37% $(1/e)$ of training instances will not be drawn in any bootstrap sample (out-of-bag instances). Next, discriminative subgraphs are mined from the drawn instances, according to minimum total support and significance thresholds. Each subgraph in the result set has associated support values, $p$-value, and bias, as calculated on the bootstrap sample. Finally, the method re-assesses the support values by performing isomorphism tests with the result subgraphs on the out-of-bag instances ("matching"), and records these values as follows:

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 $(g, k_1, \ldots, k_{|I|})$ 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 $(\mathbf{k_1} \ldots \mathbf{k_{|I|}})$, where $\mathbf{k_i}$ is a vector $(k_i^1 \ldots k_i^N)$, containing all the support values. 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. The total support is determined from the class specific support values by summing up vectors $\mathbf{k_i}$ across classes: $\mathbf{k} = \sum_{i=1}^{|I|} \mathbf{k_i}$.

From the recorded results, support values, $p$-value, and bias are estimated. Two methods, described in the next sections, are employed, 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 $\overline{\mathbf{k_i}}$, i.e. the sample mean across the entries of vector $\mathbf{k_i}$ (ignoring missing values), for all $i \in \{1, \ldots, |I|\}$, and $k$ 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:

$$\alpha_{\mathbf{g'}} = \left( \frac{1 + |\mathbf{k_1}|_1}{|I| + |\mathbf{k}|_1}, \ldots, \frac{1 + |\mathbf{k_{|I|}}|_1}{|I| + |\mathbf{k}|_1} \right) \quad (2)$$

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 $(k_1^j, \ldots, k_{|I|}^j)$ pertaining to $g$, a probability distribution is determined from this collection of multinomials:

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

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

$$\overline{\mathbf{k_i}} = \frac{\sum_j k_i^j p((k_1^j, \ldots, k_{|I|}^j))}{\sum_j p((k_1^j, \ldots, k_{|I|}^j))} \quad (4)$$

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

## 3.4 Algorithm

The mining step from section 3.3 may be implemented using a discriminative graph mining algorithm of choice as "mother algorithm". Here, we employ backbone refinement class mining (BBRC) [9]. It achieves a certain degree of 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. 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 total 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[1]. This approach allows to store out-of-bag support values $\mathbf{k_i}$ in a hash structure, using SMARTS as keys.

Algorithm 1 switches between out-of-bag estimation and ordinary discriminative graph mining (ODGM). We consider the former first. Line 4 creates a hash table to gather results from mining bootstrap samples (line 7). The resulting subgraphs are matched on the out-of-bag instances (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 incurs negligible overhead. It consists of removing subgraphs that (line 11) do not have enough entries in the hash table, as determined by $minHashLength$, or which (line 13) do not significantly deviate from the overall distribution of classes, as assessed by the significance test (section 3.2), with contingency table calculated according to mean (section 3.3.1) or maximum likelihood estimation (section 3.3.2) method (line 12).

If $method = ODGM$, 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 esti-

[1] see `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`

**Data**: $dataBase, numBoots, minHashLength, method, minSupport, alpha$
**Result**: $[subgraphs, values]$ ; $values$ include support, $p$-values, and bias
1 **if** $method = ODGM$ **then**
2  $[subgraphs, values] \leftarrow BBRC(dataBase, minSupport, alpha)$;
3 **else**
4  $hashTable \leftarrow \{\}$;
5  **for** $i := 1 \rightarrow numBoots$ *(Parallel Processing)* **do**
6  $sample, OOB \leftarrow drawBsSample(dataBase)$;
7  $[subgraphs, values] \leftarrow BBRC(sample, minSupport, alpha)$;
8  $insert(hashTable, match(subgraphs, OOB))$;
9  $[subgraphs, values] \leftarrow []$;
10  **for** $subgraph \in keys(hashTable)$ **do**
11  **if** $length(hashTable[subgraph]) \geq minHashLength$ **then**
12  $[candidateSupportValues, candidatePValue, candidateBias] \leftarrow method(hashTable[subgraph])$;
13  **if** $candidatePValue < alpha$ **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**

mation (line 2).

# 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 are provided by Algorithm 1:

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. ODGM: Ordinary discriminative graph mining.

The process was repeated 100 times for the three methods, with minimum hash entry length of 30 for each subgraph, to ensure a statistically reliable number of support values.

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 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, residual vectors $E_1 - E_5$ capture the differences between $values^B$ and $values^T$.

Five different error measures were assessed to compare the methods:

1. $E_1$, the mean of $p^B - p^T$ over subgraphs, i.e. the bias of $p$-value errors.

2. $E_2$, the mean of $\left| p^B - p^T \right|$ 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. $E_4$, One minus the mean of $\delta(p^B \leq \alpha, p^T \leq \alpha)$ over subgraphs, i.e. the fraction wrongly recognized as significant.

5. $E_5$, One minus the mean of $\delta(bias^B, bias^T)$ over subgraphs, i.e. the fraction with wrongly recognized class bias.

In the above, $\delta(\cdot, \cdot)$ is the Kronecker function, which returns 1, if the arguments are equal, and 0 else.

Six molecular, class labeled datasets were used in the experiments. Four were drawn from the carcinogenic potency database (CPDB)[2], 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. An appropriate value for minimum support (parameter $s$ in Algorithm 2) was determined a priori for each $G_{Train}$ and kept constant.

## 4.2 Results

Table 2 details the results in the form of mean values and standard deviations across $N = 100$ bootstraps for all error measures. Figure 1 shows win/loss statistics. It compares out-of-bag methods MLE and MEAN with ODGM, by pooling the former (OOB). This is possible, since they showed uniform win/loss behaviour against ODGM. It also compares MLE against MEAN. Numbers indicate the total amount of wins, numbers in brackets the subset of significant wins, as obtained by Wilcoxon signed rank tests ($p = 0.001$).

E1 and E2 measure the numerical bias and error on the $p$-value threshold that determines whether a subgraph is included in the result set. E3, E5, and E4 describe errors

[2] `http://potency.berkeley.edu/cpdb.html`

**Data**: $graphDatabase, method$ (MLE, MEAN, or ODGM), $s$
**Result**: $E_1, E_2, E_3, E_4, E_5$
$E_1 = E_2 = E_3 = E_4 = E_5 = []$;
**for** $i := 1 \rightarrow 100$ **do**
  $[trainSet, testSet] \leftarrow splitStratified(graphDatabase, 0.5)$;
  $[subgraphs, values^B] \leftarrow Algorithm\,1(trainSet, 100, 30, method, s, 0.05)$;
  $[subgraphs, values^T] \leftarrow match(subgraphs, testSet)$;
  **for** $j := 1 \rightarrow 5$ **do**
    $E_j \leftarrow [E_j, errorJ(values^T, values^B)]$;

**Algorithm 2**: **Calculation of error measures**

| Method | E1 | E2 | E3 | E4 | E5 |
|---|---|---|---|---|---|
| OOB | 6 (6) | 6 (5) | 5 (4) | 6 (5) | 6 (5) |
| ODGM | 0 (0) | 0 (0) | 1 (0) | 0 (0) | 0 (0) |
| MLE | 2 (1) | 2 (0) | 6 (4) | 5 (3) | 4 (0) |
| MEAN | 4 (3) | 4 (2) | 0 (0) | 1 (0) | 2 (0) |

**Figure 1: Wins and losses**

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

**Table 2: Bias and accuracy**

on relative support values, class bias, and significance (as a binary attribute), respectively. The latter measures are readily interpretable, thus we discuss them separately from E1 and E2.

We first consider E1 and E2. Judging from Table 2, for four datasets (apart from INT and KAZ), there is a clear improvement when out-of-bag methods are applied, compared to single runs of discriminative subgraph mining. The advantages for OOB methods over ODGM are lowest for datasets MOU and INT. For INT, $p$-value bias (E1) is much smaller, but not $p$-value error (E2). In the significance tests, a difference to ODGM could not be found here. All other pairs of OOB methods against ODGM were significantly different. For KAZ, E1 and E2 are so low, that they do not show in Table 2, due to rounding. However, the differences are statistically significant, also for KAZ.

Comparing OOB methods among each other, both show similar values, according to Table 2. Both seem to be relatively unbiased (E1), however, Figure 1 shows that MEAN seems to have advantages over MLE. For absolute error (E2), the situation is less clear.

We now consider E3, E4, and E5. For E3, the relative support value error, OOB methods estimate the support values always better than ODGM, as Table 2 shows. However, the difference was not significant for INT and KAZ. MLE does always better than MEAN, and most of the time significantly better.

For E4, the error on binary significance estimation, ODGM errors are very large: Most of the time, the majority of subgraphs were falsely estimated as significant. MLE and MEAN do both much better for all datasets, also significantly better, except for KAZ. Also, MLE performs better than MEAN in all cases except KAZ (three times with significant difference).

For E5, the class bias error, both MLE and MEAN drastically improve on ODGM, as Table 2 shows. The differences are statistically significant for all datasets, except KAZ. Between MLE and MEAN, practically no significant differences could be found.

Table 2 also gives the mean number of subgraphs generated in the last column. Much less subgraphs are generated by OOB methods. There is a general trend for lower fractions of subgraphs with shrinking dataset sizes, compared to ODGM.

## 5. DISCUSSION AND CONCLUSIONS

Especially for rather small databases, which are typical in the field of toxicology, the proposed methods can significantly improve on the estimation of statistical quantities, compared to ordinary discriminative graph mining. Thus, for such settings, we conclude that it does not to suffice treat database graphs in a single pass, as is done by ordinary discriminative graph mining, in order to obtain significantly class-associated subgraphs. Instead, resampling methods should be used that repeatedly assess subgraph sig-

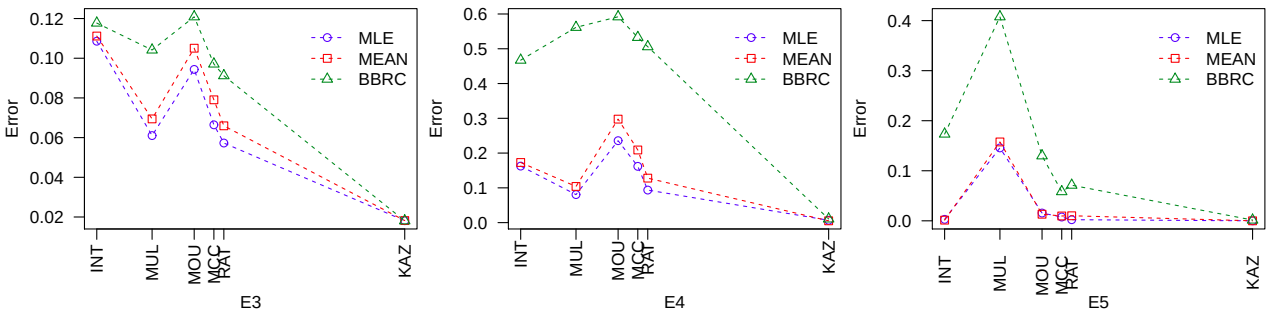


**Figure 2: Error measures along increasing logarithmic dataset size**

nificance on data sets that obtained by perturbations to the training data. To validate the data, out-of-bag estimation is a feasible approach. The procedure may also be embedded in a workflow to to learn a bagged predictor.

We investigate the interpretable error measures E3, E4, and E5 again in the charts in Figure 2, where their values are plotted along increasing dataset size. The latter was logarithmically transformed, to visually better separate similarly sized datasets MOU, MCC, and RAT.

Along the three smallest datasets, error measures vary. However, the difference to ODGM is always pronounced (apart from KAZ), and for all methods, the variations per method are similar along the datasets. For the KAZ dataset, there is no advantage for OOB methods.

The estimation of the prominent class for each subgraph, captured by E5, is obviously the easiest task. Apart from MUL, OOB methods have E5 values close to zero. The corresponding values for ODGM are disturbingly high.

E3 and E4 are related: $p$-values, analyzed in E4, are based on the support values, which are the subject of E3. The improvements provided by OOB methods in the estimation of support values seems to help the estimation of significance a lot, as the slopes of the lines are similar, but the difference to ODGM is much larger for the latter.

For larger datasets, there is a trend towards increasing performance for all methods, as indicated by dashed lines. We attribute this to the increasing precision of ODGM (which is wrapped by the OOB methods) in the presence of more learning data. However, the fraction of circa 50 % of subgraphs wrongly determined by ODGM in E4 even for the second largest dataset (RAT) indicates that mining significant subgraphs from training data directly is not reliable. Only for the KAZ dataset, E4 drops to levels similar to OOB methods.

## 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.