Mach Learn (2008) 70: 189–206
DOI 10.1007/s10994-007-5034-6

# Margin-based first-order rule learning

**Ulrich Rückert · Stefan Kramer**

Received: 24 September 2007 / Accepted: 9 October 2007 / Published online: 10 November 2007
Springer Science+Business Media, LLC 2007

**Abstract** We present a new margin-based approach to first-order rule learning. The approach addresses many of the prominent challenges in first-order rule learning, such as the computational complexity of optimization and capacity control. Optimizing the mean of the margin minus its variance, we obtain an algorithm linear in the number of examples and a handle for capacity control based on error bounds. A useful parameter in the optimization problem tunes how evenly the weights are spread among the rules. Moreover, the search strategy for including new rules can be adjusted flexibly, to perform variants of propositionalization or relational learning. The implementation of the system includes plugins for logical queries, graphs and mathematical terms. In extensive experiments, we found that, at least on the most commonly used toxicological datasets, overfitting is hardly an issue. In another batch of experiments, a comparison with margin-based ILP approaches using kernels turns out to be favorable. Finally, an experiment shows how many features are needed by propositionalization and relational learning approaches to reach a certain predictive performance.

**Keywords** First-order learning · Relational learning · Rule learning · Margins · Capacity control

## 1 Introduction

Most systems in ILP employ a rule-based representation language to learn first-order concepts. This is no surprise given that many of the challenges that are inherent in first-order learning are similar to their counterparts in propositional rule learning. Unfortunately, learning disjunctive or conjunctive sets of rules has been proven to be hard to handle from an

Editors: Stephen Muggleton, Ramon Otero, Simon Colton.

U. Rückert (✉) · S. Kramer
Institut für Informatik/I12, Technische Universität München, Boltzmannstr. 3,
85748 Garching b. München, Germany
e-mail: rueckert@in.tum.de

S. Kramer
e-mail: kramer@in.tum.de

analytical point of view. This is in part due to the combinatorial complexity of the classical DNF rule set representation (e.g., learning small sets of $k$-term DNF formulae is NP-hard, Kearns and Vazirani 1994). Consequently, most traditional rule learning algorithms resort to heuristics for model induction. In particular, capacity control (e.g. pruning) comes either at the expense of data or is based on heuristics which are difficult to compare and may or may not work well in a particular learning setting.

To overcome these problems, modern rule learning algorithms are often based on weighted rule sets, that is, sets of rules, where a weight is attached to each rule (Friedman and Popescu 2005). To classify an instance, an algorithm simply adds up the weights of those rules whose conditions are met. If the sum is positive, the system predicts the positive class, otherwise, it predicts the negative class. This introduces the notion of a margin to rule learning and a rich field of analytical results from ensemble or kernel-based methods become applicable to rule learning.

Building on these techniques, we introduce a new margin-based approach to first-order rule learning. The approach uses a novel optimization criterion, *Margin Minus Variance (MMV)* (Rückert and Kramer 2006), that is particularly well-suited for first-order rule learning, because it can be calculated in time linear in the number of examples and allows the derivation of error bounds for capacity control. In previous work (Rückert and Kramer 2006), we introduced MMV and applied it in the context of propositional rule learning to test the feasibility of our ideas. In this paper, we arrive at our goal of MMV for first-order rule learning, extend its scope and evaluate it experimentally on relational data. We call the new system RUMBLE ("Rule and Margin Based Learner"). It is able to handle multiple plugins for various types of data, e.g., for logic, graphs and mathematical terms. Moreover, our implementation of MMV optimization contains a flexible and generic way to specify a declarative inductive bias. It can be configured to base its rule generation process on different kinds of information, thus bridging the gap between propositionalization (Kramer et al. 2001) and relational learning approaches that generate and evaluate discriminative features dynamically.

Another important focus of this work is on the general spectrum of methods between comprehensibility and predictive performance. Traditional logic-based representations are easily comprehensible, but often lack the predictive accuracy of methods from statistical machine learning. On the other hand, statistically motivated methods such as ensembles or SVMs are more accurate, but often induce bulky or incomprehensible models. With MMV optimization, we have an easy way to control the sparseness of the induced rule sets, allowing the user to find a compromise between comprehensibility and predictivity: A subtle, but important extension to our previous work allows the use of an arbitrary $p$-norm in the constraints of the optimization problem instead of the 1-norm. Setting the value of the parameter $p$ accordingly, we can tune to which degree the weights are distributed among the rules.

This paper is organized as follows: In Sect. 2, we briefly discuss related approaches for rule learning and margin-based approaches in ILP. Section 3 explains MMV optimization including the generalization to an arbitrary $p$-norm, and various rule generation strategies for structured data. In Sect. 4, we report on extensive experiments: First, we show that overfitting is hardly an issue in the context of commonly used toxicological datasets. Second, the experiments show that margins can be useful in ILP rule learning even without kernels. Third, we investigate the effect of various feature generation strategies between propositionalization and relational learning. In Sect. 5, we summarize our main points and suggest a direction for further work.

## 2 Related work

Rule learning has a very long tradition in Machine Learning. While early rule learners represented rule sets as disjunctions, more modern (propositional) approaches use weighted rule sets. Most of those systems are based on ensemble theory. In propositional rule learning, one of the first approaches to do so was SLIPPER (Cohen and Singer 1999). It is based on boosting, where the weak learner is a traditional rule generation procedure. More recently, Friedman and Popescu (2005) have introduced a system based on ensembles of rules. The algorithmic framework is conceptually based on bagging, though it can be parameterized to resemble other methods such as Bayesian model averaging. One disadvantage of those systems is that ensemble theory does not provide any justification to stop adding rules to an ensemble. Therefore, rule ensembles tend to be larger than strictly necessary.

Due to its roots in first-order logic, most multi-relational rule learning algorithms are based on disjunctions or conjunctions of rules. Recently, interest in margin-based relational learning has been spawned. Most of the work in this direction builds on kernel methods: Muggleton et al. proposed Support Vector Inductive Logic Programming (Muggleton et al. 2005), where the clauses that are induced by CProgol5.0 are used as features to represent the training data propositionally and thus allow the application of a linear SVM. A different approach is taken by Landwehr et al. (2006). They replace the rule evaluation function in FOIL with a novel scoring function, which rates the quality of a set of clauses as the accuracy of a SVM that is built on a propositional data representation, where the clauses are again used as features. Finally, Woźnica et al. (2005) extend convolution kernels to work on relational data. The main idea is to represent each instance in the training set as a tree whose edges are connections between the tuples in different relations. This instance representation can be dealt with convolution kernels so that standard SVM methods apply. Popescul and Ungar (2003) extend Logistic Regression to the relational setting. They apply refinement operators to generate new features from relational data and add those features to a logistic regression model until the model overfits according to the Bayesian information criterion (BIC).

As our approach aims at finding an explicit representation of the rules, we do not need the power of kernels to implicitly project into a higher dimensional feature space. Consequently we can use margin-based methods without some of the restrictions that are often attached to kernel-based approaches. In the next section we outline the concepts that underlie the proposed system.

## 3 Margin-based rule learning

The approach to margin-based rule learning presented in this paper can be divided into three conceptual subtasks. First, we motivate MMV optimization as a feasible relaxation of the infeasible direct empirical risk minimization. MMV is related to LDA (Linear Discriminant Analysis) in that it considers the variance and covariance information, but it does not make any assumptions about the data generating distribution and it uses the margin rather than the Rayleigh quotient to determine the decision boundary. Second, we use a novel risk bound to perform capacity control and avoid overfitting. Third, we describe a flexible way to adjust the learner's inductive bias, that is able to work in propositionalization-like and traditional ILP settings.[1]

[1] In the general setting, ILP induces rules that can have variables in the head, thus allowing for recursive rule definitions. In this paper we deal only with non-recursive rules.

### 3.1 Margin minus variance optimization

In order to give a more formal definition of the setting, we need some basic definitions. First of all, we presume the "learning from interpretations" setting (De Raedt 1997). In this setting, each instance is represented by a set of ground atoms or, equivalently, a set of tuples taken from a fixed set $\mathcal{R}$ of relations. For example, one can represent a dataset of molecules using the three relations $\mathcal{R} = \{mol,atom,bond\}$: a tuple in the relation *mol* represents a molecule, *atom* contains the atoms in the molecules and *bond* describes the bonds between those atoms. In this example, the *instance* $x = \{\text{mol}(m_1,18.02), \text{atom}(a_1, m_1, \text{H}), \text{atom}(a_2, m_1, \text{H}), \text{atom}(a_3, m_1, \text{O}), \text{bond}(b_1,a_1,a_3), \text{bond}(b_2,a_2, a_3)\}$ represents a water molecule with molecular weight 18.02. Let $\mathcal{X}$ denote the set of all possible instances and $\mathcal{Y} = \{-1, +1\}$ the (binary) set of *class labels*, where the values $-1$ and 1 represent the negative and positive class respectively. We follow the usual convention in classification and assume that the instances are drawn i.i.d. according to a fixed but unknown distribution $D$, ranging over $\mathcal{X} \times \mathcal{Y}$. A sample $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of size $m$ is drawn and given to a learning algorithm. The task of the learning algorithm is to induce a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ whose *predictive accuracy* $P_{(x,y)\sim D}(h(x) = y)$ is as large as possible.

Furthermore, we assume we already have a (possibly infinite) repository of rules $R = \{r_1, r_2, \ldots\}$, where a rule $r_j : \mathcal{X} \to [-1, 1]$ assigns a value between $-1$ and 1 to each instance. A typical rule could be "assign $+1$, if the molecule contains an aromatic ring, and $-1$ otherwise" or "assign $+1$, if the molecular weight is greater than 51.5 and $-1$ otherwise". We will later present a scheme to enumerate the rules in the repository declaratively. The results in the paper also hold for rules that assign intermediate values, such as 0.5 ("probably positive") or 0 ("don't know"), but in general we assume that $r(x) \in \{-1, +1\}$. Let $x_i(j)$ denote the result of the application of rule $j$ on instance $x_i$. If we consider only the first $n$ rules, we can represent the $i$th instance by the vector of rule values $\bar{x}_i := (x_i(1), x_i(2), \ldots, x_i(n))^T$. Likewise, a weighted rule set can be given by a weight vector $w \in [-1, 1]^n$. An individual rule set $w$ assigns class label $\text{sgn}(w^T \bar{x}_i)$, so that instance $x_i$ is positive, if $\sum_{j=1}^n w_j x_i(j) \geq 0$ and negative otherwise. Representing a rule set as a weight vector might seem unusual. However, even if the number of rules $n$ is large, we can still deal with small rule sets, if most components of $w$ are set to zero. Also note that the weight vector defines a hyperplane separating $[-1; 1]^n$ into two half-spaces so that rule sets in our setting are related to linear classifiers and perceptrons.

Usually, the *zero-one loss* $l(w^T \bar{x}, y) = \mathbf{I}[\text{sgn}(w^T \bar{x}) \neq y]$ is used in classification to define the *empirical error* $\hat{\varepsilon}_w := \frac{1}{m} \sum_{i=1}^m l(w^T \bar{x}_i, y_i)$ and the *true error* $\varepsilon_w := \mathbf{E}_{(x,y)\sim D} l(w^T \bar{x}, y)$. We are interested in finding a weighted rule set $w$ that minimizes the true error $\varepsilon_w$. However, since $D$ is unknown, we have no information about the true error and the best we can do is to minimize the empirical error $\hat{\varepsilon}$ instead. Note that scaling a weight factor $w$ with a positive factor does not change the actual classification of the instances. Thus, in practice, one often restricts the space of possible weight factors. Unfortunately, due to the discontinuity of the zero-one loss, empirical risk minimization is a combinatorial optimization problem. It has been shown to be NP-hard and is also hard to approximate up to a fixed constant (Ben-David et al. 2003). One way to avoid those computational expenses is to optimize $\hat{\varepsilon}$ not directly, but through a related quantity. For example, support vector machines restrict the hypothesis space to weight vectors of unit length $w \in \{x \mid \|x\| = 1\}$ (as scaling $w$ with a factor does not change the classification) and search for a $w$ that maximizes the margin (or the soft margin) to the nearest training instances. Optimizing for a *large margin* has two advantages: First, the resulting quadratic programming problem can be solved efficiently, and second, it allows for the application of the kernel trick to elegantly deal with large feature spaces.

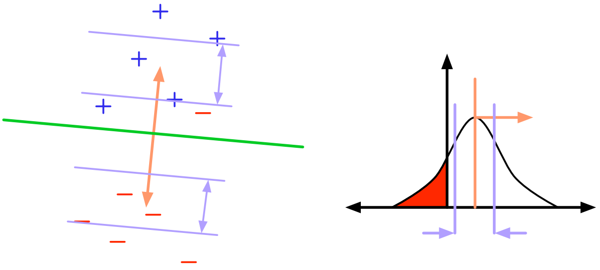

**Fig. 1** The distances between a hyperplane and the training instances (*left*) induce a distribution of margins (*right*). MMV optimization aims for hyperplanes whose margin distribution features a large mean and a small variance

For our purposes, though, the kernel trick is not helpful, because we are looking for an explicit representation of the rule set, not one that is implicitly defined by a kernel. Thus, a different approach might be desirable. One problem of the large margin approach is that the classifier depends only on the support vectors and ignores the other data. If the support vectors are not very representative of the underlying distribution, the classifier performs worse than necessary. One approach to overcome this problem is to introduce regularization, which requires an additional parameter. A different approach is to consider the margin to all training instances. Ideally one would want to maximize the average distance from the separating hyperplane to the training instances and to minimize the sample variance of these distances (see Fig. 1). More formally: given an instance $(x, y)$ let $\mu_w(x, y) := w^T \bar{x} \cdot y$ denote the *margin*. The margin is positive, if the instance is classified correctly by $w$ and negative otherwise. Similarly to true and empirical error, one can define the *empirical margin* $\hat{\mu}_w := \frac{1}{m}\sum_{i=1}^{m} \mu_w(x_i, y_i)$ and the *true margin* $\mu_w := \mathbf{E}_{(x,y)\sim D}\mu_w(x, y)$. One can also estimate the sample variance of the margins from a training set; the *empirical variance* is $\hat{\sigma}_w := \frac{1}{m-1}\sum_{i=1}^{m}(\mu_w(x_i, y_i) - \hat{\mu}_w)^2$, the *true variance* is $\sigma_w := \mathbf{E}_{(x,y)\sim D}(\mu_w(x, y) - \mu_w)^2$. Using this notation, one can look for weight vectors $w$ that maximize the empirical margin but minimize the empirical variance:

$$\begin{aligned} &\underset{w}{\text{maximize}} \;\; \hat{\mu}_w - \hat{\sigma}_w \\ &\text{subject to} \;\; \|w\|_p = 1 \end{aligned} \tag{1}$$

where the constant $p \geq 1$ in the norm of the constraint is a user-adjustable parameter. We call the optimization function *margin minus variance* (MMV) and denote it by $\hat{\gamma}_w := \hat{\mu}_w - \hat{\sigma}_w$. MMV optimization is intuitively appealing, since a low MMV also gives a low error, and technically useful, since the true error can be upper-bounded based on the MMV. The constraint in the above optimization problem is not strictly necessary; MMV is a concave function so that the maximum is unique and global. However, it is convenient to restrict the length of the weight vector $w$ in order to limit the range of possible values the MMV function can take. This is crucial to derive the concentration inequality used in Sect. 3.2. The actual form of the constraint is not determined by the learning setting. It turns out, however, that the constraint based on the $p$-norm $\|x\|_p := (\sum_{i=1}^{n} |x|^p)^{\frac{1}{p}}$ allows for a nice way to regulate the bias of the MMV optimization. The norm value $p$ controls how evenly the weights are spread across the rules. If $p = 1$, most of the weights are set to zero and only the few most informative rules get a non-zero weight. This is similar to the traditional rule learning setting where one aims at a small set of good rules. If $p = \infty$, MMV assigns the weight $+1$ or $-1$ to almost all rules. This setting is related to ensemble techniques such as bagging, where the prediction is determined by a voting process. Often, predictive accuracy is best for values of $p$ near two, because MMV attaches large weights to highly informative rules and small weights to less significant rules. Of course, the best value of $p$ depends on the actual learning problem. For instance, learning from gene expression data works usually best with small values of $p$, because typically only a few genes are relevant. On the other hand, predicting the toxic activity of small molecules favors large values of $p$, because the activity of a compound may depend on many different mechanisms. Thus, the parameter $p$ allows the user to adjust the bias of MMV in a meaningful way. If $p > 1$, maximizing MMV is a semidefinite quadratic optimization problem with nonlinear constraints that can be solved in time linear in the number of examples and, in the worst case, cubic in the number of rules.

### 3.2 Capacity control for MMV optimization

Using the MMV maximization procedure one can efficiently determine "good" weights for the rules in a predefined repository of rules. Obviously, the success depends to a large degree on the right size of the repository, to avoid over- or underfitting. The standard approach to this is to start with a small repository, then iteratively increase its size until an estimate of the structural risk is optimal. However, traditional structural risk minimization is not optimal in our setting, since it is too coarse-grained (i.e., it does not work on the level of individual rules), and is based on error, not MMV. Therefore, we derived a new risk bound to estimate the structural risk of a given repository in terms of the difference between true and empirical MMV. $\mathcal{B}_p^n := \{x \in \mathbb{R}^n \mid \|x\|_p \leq 1\}$ denotes the unit ball according to the $p$-norm in $\mathbb{R}^n$.

**Theorem 1** *Let $D$ be a fixed unknown distribution over $\mathcal{B}_\infty^n \times \{-1, +1\}$ and $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a sample of size $m$ drawn i.i.d. from $D$. Then for all $\delta > 0$ and all $w \in \mathcal{B}_1^n$*:

$$\mathbf{Pr}_{S\sim D}\left[\gamma_w \geq \hat{\gamma}_w - \sqrt{18\frac{2\ln n + \ln\frac{1}{\delta}}{m}}\right] \geq 1 - \delta. \tag{2}$$

See Appendix for a proof. The inequality basically states that the true MMV is with high probability greater than the empirical MMV minus a risk term that depends logarithmically on the size of the class of rules. It can be used also for $w \in \mathcal{B}_p^n$ with arbitrary $p$, if the $w$ is divided by its 1-norm before calculating $\hat{\gamma}_w$ . The inequality provides a pessimistic estimate of the structural risk that is involved with using larger classes of rules. The constant 18 in the risk term is too pessimistic for most real world settings and applicable only for worst-case analysis; for practical model selection applications one should use a smaller constant. The following lemma indicates that one can use the true MMV to bound the true error (see Rückert and Kramer 2006 for further discussions).

**Lemma 1** *Let $D$ be a fixed unknown distribution over $\mathcal{B}_\infty^n \times \{-1, +1\}$ and $w \in \mathcal{B}_1^n$ be a weight vector with true error $\varepsilon_w$ and true MMV $\gamma_w$. Then*:

$$\varepsilon_w \leq \begin{cases} \frac{\gamma_w}{4\gamma_w^2+\gamma_w} & \textit{if } \gamma_w \leq 0.5, \\ \frac{1-\gamma_w}{2-\gamma_w} & \textit{if } \gamma_w > 0.5. \end{cases} \tag{3}$$

See Appendix for a proof.

### 3.3 The learning algorithm

With the tools presented in the preceding section we have the main building blocks to design a statistically motivated rule learning system. An outline is given in Algorithm 1. The system

**Algorithm 1** The RUMBLE learning algorithm

```
procedure Learn($X$)
    $R^{(0)} \leftarrow \emptyset$
    for $i = 1, 2, \ldots$ and while new rules available
        $R^{(i)} \leftarrow$ add new rule to $R^{(i-1)}$
        $T^{(i)} \leftarrow$ apply rules in $R^{(i)}$ to instances in $X$
        $w^{(i)} \leftarrow \mathrm{argmax}_w \hat{\gamma}_w(T^{(i)})$
        $\gamma^{(i)} \leftarrow$ bound calculated from $w^{(i)}$ and $n = |R^{(i)}|$
    end for
    return $(R^{(i)}, w^{(i)})$ with the maximal $\gamma^{(i)}$
end procedure
```

starts with the training set given in a set $X$ and an empty set of rules. In the main loop, it adds a new rule to the set of rules and applies the rules to the examples in the training set. It determines the weight vector $w^{(i)}$ optimizing the empirical MMV on this data and calculates an upper bound on the corresponding true value given the empirical quantity and the number of rules. If no new rules are available, the algorithm terminates the loop and returns the set of rules and the weight vector which achieved the best bound. RUMBLE can be adjusted with two parameters: The constant $b$ replaces the 18 in Theorem 1. It specifies how strict the algorithm should be with regard to overfitting avoidance. If overfitting is not an issue, one can set the constant to zero and thus select the repository with the best empirical MMV. The constant $p$ is used for the $p$-norm in the constraint in (1). It specifies how evenly the weights should be distributed among the rules and quantifies how large and diverse the rule set should be. To complete the system, one needs two additional subroutines: one to solve the empirical optimization problems and another to generate new rules for the training set.

We formulate the MMV optimization task as a standard quadratic programming problem. This is a semidefinite quadratic programming problem with one equality and $2n$ inequality constraints, if $p = 1$, and a quadratic program with quadratic constraints if $p > 1$. It can be solved using any established algorithm for constrained optimization. Usually, the computationally most demanding part of the process is to determine the set of active constraints. As the number of non-zero weights is usually rather small, this optimization step is typically very fast.

The procedure to generate the rules in the repository is outlined in the next section. In practice, it is not necessary to store the covariance matrix for all rules in the repository. Many weights will be too small to make a difference for prediction anyway. In our implementation we therefore limit the number of rules in the model to 100 and remove the rule with the lowest weight whenever the model's size exceeds this limit. This also avoids the cubic time complexity during MMV optimization.

### 3.4 Rule generation for structured data

So far, we have specified, how large the repository should be in order to avoid overfitting, but we have not given a specific procedure to generate the rules. Fundamentally, one would like to generate rules that are as informative as possible about the class label. However, a rule that may contain a lot of information in one learning setting, can be useless in another setting. As the number of rules that can be generated for multi-relational domains is virtually unlimited, it is unreasonable to assume that a fixed rule generation order can work well in many cases. Instead it makes more sense to allow the user to adjust the rule generation order in a flexible and generic manner, so that she can incorporate common sense or explicit background knowledge. Before we specify RUMBLE's approach to rule generation, we point out two interesting observations.

First, it is noteworthy that the system's predictive performance depends not only on the rule repository as a whole, but also on the order, in which the rules are evaluated. If the system generates a large number of rules, it is quite likely that one of the rules agrees with the target class label just by pure coincidence. Hence, the risk of overfitting increases with the number of generated rules and the structural risk term in (2) penalizes rules that are added only late. The user should therefore configure the system to generate presumedly informative rules first. For instance, in the task of predicting carcinogenicity for small molecules, one could generate rules that check for the existence of halogens (which are known to contribute to carcinogenicity) before presumedly less informative rules. This imprecise knowledge is in contrast to the design in many ILP systems, where the background knowledge is encoded in a precise logical form. Also note that the order can be interpreted as a Bayesian prior: early rules have greater prior probability.

The second point is motivated by the question on whether RUMBLE should be regarded as a propositionalization approach or rather as an ILP system in the classical sense. It turns out that the border between those two categories is somewhat blurry and that RUMBLE can be configured to resemble either approach. In the following we try to clarify this distinction in a more precise way. First of all, all learning algorithms need to extract information from the training data. In this sense, classical relational learning systems also generate "features", i.e., queries on the training data. The main difference to propositionalization is the fact that relational learning methods generate those queries only when they are needed during the hypothesis induction stage. Thus, the fundamental difference between relational learning and propositionalization lies in the question of which information is evaluated in order to decide whether a query is generated and included in the hypothesis. The following list categorizes learning algorithms depending on the type and amount of information that is used for making those decisions.[2]

1. *Agnostic feature generation*. This is the most extreme form of propositionalization, where the queries are generated without accessing the training data. The set of queries is fixed before the actual learning starts. Of course, this approach runs the risk of generating many uninformative features, such as queries that are true or false on all training instances. Agnostic propositionalization makes sense only if there is significant evidence that a particular feature set works well for the problem at hand.
2. *Instantiation-dependent feature generation*. In this case, a *query* is generated if it can be expected that its *instantiation* meets certain conditions. Typically, systems in this category try to ensure that the number of covered instances is between a minimum and a maximum threshold. This is a reasonable goal as queries that are satisfied (or violated) by only a few instances can contribute only a limited amount of discriminative information. A typical example of an instantiation dependent propositionalization approach is substructure mining in structure activity relationships.
3. *Interaction-dependent feature generation*. This category differs from the preceding one in that the instantiations of the existing features are also taken into account for deciding a new query's suitability. The main motivation is to generate queries that complement the existing ones, in the sense that they cover a significantly different subset of instances.

[2] Here, the terms "features", "queries" and "rules" are used with the same meaning.

This ensures a certain variability in the feature set and improves the set's discriminative power.

4. *Class-dependent static feature generation*. Methods of this type take into account the class label in the process of generating queries. However, this is done before queries are actually assessed and potentially incorporated into the model.
5. *Class-dependent dynamic feature generation*. While methods in this category also use class information as they generate new queries, they evaluate and incorporate features dynamically, depending on the features in the model so far.

Types one to four belong to the family of propositionalization methods, type five is typically used in traditional ILP systems for decision tree and rule learning. Of course, this categorization is far from being exhaustive and could be easily extended to include other criteria. In the following we describe the rule generation method that is implemented in RUMBLE. It can be configured to work in any of these settings. The main goal for the design of the rule generation process is to allow for a flexible and generic way to specify a declarative bias. Consequently, we do not restrict the data representation to a particular format. For instance, it does not make sense to force the user to encode molecules as Prolog facts, if the data is already available in a much more efficient representation (e.g., a canonical form for a graph mining tool). Instead, we implemented a plugin architecture, where the user can upgrade RUMBLE with plugins for various data representations. Each plugin offers a set of *refinement operators* that are called by the system to generate new rules. The refinement operator can access the existing set of rules, the weights of the current rule set, the training data and generic background knowledge to form new rules. A setup text file specifies, which plugins are loaded and which refinement operators are called in which order to generate the rules. Right now, RUMBLE provides four plugins:

– *Terms*. These are simple mathematical expressions such as $(a_1 + a_2)/2$ or $(a_1 \wedge a_3) \vee a_5$. The plugin provides refinement operators for adding new (fixed) term rules, modifying existing term rules or combining existing terms to form a new rule.
– *FreeTree*. Predicting the biological activity of small molecules is an important and popular application of ILP. Thus, we incorporate a version of the frequent substructure mining tool FreeTreeMiner (Rückert and Kramer 2004) as a plugin. It offers refinement operators that can build rules checking for the occurrence of a particular substructure, create all substructure rules that occur more frequently in the training data than a predefined threshold or refine existing rules depending on the existing rule set or weights.
– *Prolog*. Prolog is a powerful and generic way to represent general data and background knowledge. The Prolog plugin incorporates a fully featured Prolog engine based on Cx-Prolog (Dias 2006). This enables the user to write her own refinement operators in Prolog.
– *Meta*. This plugin can combine existing rules from other plugins using simple logical combinations. The refinement operators can be configured to combine only rules whose instantiations meet certain conditions, e.g. minimum mutual information.

For instance, it is possible to specify that RUMBLE should first generate 200 rules describing the substructures that occur in more than 20% of the molecules in the dataset. Then, the Term plugin can be configured to generate five rules that test the molecules' LogP value as discretized in 5 equal-frequency bins. Finally, the Meta plugin can be used to combine those of the first 200 rules disjunctively, whose mutual information is among the fifty lowest.

## 4 Experimental results

In order to evaluate MMV optimization, we perform a range of experiments. First, we investigate the role of the norm parameter and the bound parameter on the mutagenesis dataset (Srinivasan et al. 1996). We then test RUMBLE on a range of benchmark datasets and compare the results with those of other ILP systems. Finally, we compare the performance of different rule generation biases.

### 4.1 Parameters for MMV optimization

Our implementation of RUMBLE can be adjusted with two parameters. The first parameter is the value of the norm $p$ in (1). This parameter controls how evenly the weights are distributed among the rules. The second parameter is the constant $b$ that is used instead of the pessimistic value 18 in the structural risk bound (2). This parameter determines the degree of overfitting avoidance. We performed a set of experiments to evaluate the impact of those two parameters on the predictive performance of the system on the mutagenesis dataset. We configured the Prolog plugin with a bias that generates rules for the discretized LogP and Lumo values and a Prolog refinement operator that generates all acyclic substructures that appear at least three times in the data. For the first experiment, we set $b$ to zero and choose $p \in \{1, 1.5, 2, 2.5\}$. Figure 2 shows the results for generating 600, 800, 1000, 1400, 1800 and 2400 rules. The first observation is that $p = 1$ performs constantly worse than the other values. Apparently, there is a certain trade-off between predictivity and comprehensibility on this dataset; small rulesets are simply not as predictive as the larger ones. The difference between the other values of $p$ is marginal and $p = 2$ appears to be a good compromise.

The second observation is the fact that MMV optimization does not seem to overfit, even for very large numbers of generated rules. This is a surprising result given that overfitting was a major issue on propositional datasets (Rückert and Kramer 2006), where setting the overfitting parameter $b = 1.0$ gave the best results. The reason for this phenomenon is not yet clear. One explanation might be that MMV tends to assign comparably low weights to the larger substructures, which appear only in a limited number of instances. Hence, rules containing large substructures can make a difference only if many of them vote for the same target class. This resembles the voting process in ensemble methods, which are known to be insusceptible to overfitting. Another explanation could be that several modes of action are involved in chemical mutagenesis. Thus, unlike in pharmacological applications, the weights are distributed among many potentially activating or deactivating substructures, and there is no single structural feature that correlates well with the class label. To test this hypothesis, we conducted two experiments with datasets from pharmacological applications: The

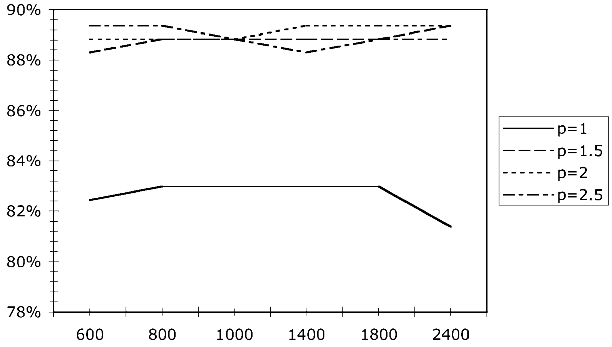

**Fig. 2** The predictive accuracy of MMV on mutagenesis for $p \in \{1, 1.5, 2, 2.5\}$ and the maximum number of rules ranging from 600 to 2400

Yoshida dataset (Yoshida and Topliss 2000) consists of 265 molecules classified according to their bio-availability. The second dataset classifies molecules according to the degree to which they can cross the blood-brain barrier (BBB) (Li et al. 2005). Our hypothesis was partly confirmed by the experiments: In contrast to the toxicological datasets, the difference between training and test set error increased sharply with the number of substructures. However, the best overall models were still found with a large number of features.

### 4.2 Comparison to other ILP systems

In the second batch of experiments we compared the predictive accuracy of MMV optimization to those of other margin-based approaches. Following the discussion in Sect. 4.1, we set $p$ to two and $b$ to zero for all experiments. In a recent paper Landwehr et al. (2006) present kFOIL, a version of FOIL modified to use kernels and SVMs in the rule induction step. The authors give a comparison of kFOIL with nFOIL, Aleph and the propositionalization approach c-ARMR+SVM on six datasets. Mutagenesis (Srinivasan et al. 1996) is a popular ILP benchmark dataset. To warrant a fair comparison we used atom and bond information only and give results for the regression friendly part, as the regression unfriendly part is too small to warrant precise estimation of predictive accuracy. On the Alzheimer dataset (King and Srinivasan 1995) the task is to predict the ranking of certain compounds with regard to four quantities that are known to influence Alzheimer's disease. Following Landwehr et al. (2006) we give results for amine reuptake inhibition (686 instances), low toxicity (886 examples), high acetyl cholinesterase inhibition (1,326 instances) and reversal of memory deficiency (642 instances). We used a Prolog-based bias that resembles the original GOLEM bias (Srinivasan et al. 1996). The NCTRER dataset deals with the prediction of estrogen receptor relative binding affinities of small molecules. We used the FreeTree plugin to build substructure rules for the 232 examples. The left-hand side of Table 1 gives the predictive accuracies as estimated by tenfold cross-validation for the four systems (Landwehr et al. 2006) and Rumble. As can be seen, Rumble outperforms the other approaches on all datasets. For NCTRER, one can improve the predictive accuracy of Rumble even further to 82.76% by setting $p$ to 1.5.

A different approach to relational margin based learning has been taken by Muggleton et al. (2005), where the output of CProgol5.0 is used in a linear kernel SVM. The authors compare their algorithm with partial least squares, multi-instance kernels and an RBF kernel on three chemical features on the DSSTox dataset (576 instances). The goal is to predict whether or not the toxicity of a compound is above average. We used a Prolog refinement

**Table 1** Predictive accuracy according to tenfold cross-validation for several systems. On the left, four systems are compared on six datasets with MMV optimization, on the right, five systems are compared on the DSSTox dataset

| | kFOIL | nFOIL | Aleph | c-ARMR + SVM | Rumble | Method | Pred. Acc. |
|---|---|---|---|---|---|---|---|
| Mutagenesis | 81.3% | 75.4% | 73.4% | 73.9% | 84.0% | CProgol5.0 | 55% |
| Alzh. amine | 88.8% | 86.3% | 70.2% | 81.2% | 91.1% | CHEM | 58% |
| Alzh. toxic | 89.3% | 89.2% | 90.9% | 71.6% | 91.2% | PLS | 71% |
| Alzh. acetyl | 87.8% | 81.2% | 73.5% | 72.4% | 88.4% | MIK | 60% |
| Alzh. memory | 80.2% | 72.9% | 69.3% | 68.7% | 83.2% | SVILP | 73% |
| NCTRER | 77.6% | 78.0% | 50.9% | 65.1% | 79.3% | Rumble | 76% |

operator to generate 300 rules that check for the existence of substructures in the molecules that appear at least three times in the dataset. Then, we apply a Meta refinement operator that calculates all pairs of the first 100 existing rules and combines them disjunctively. The results on the right-hand side of Table 1 give the predictive accuracies as estimated by tenfold-cross validation. RUMBLE outperforms SVILP and the other methods on this dataset. All in all, the results demonstrate that margins are useful in first-order learning without the need for kernels.

### 4.3 Comparing the biases

As outlined in Sect. 3.4, the performance of RUMBLE depends to a large degree on the order and type of rules that are generated. For most of the benchmark datasets in the preceding section, the bias essentially contains rules that express the occurrence of a particular substructure in a molecule. Thus, if one aims at finding good biases for those datasets, the main question is which substructures should be generated in which order. In particular, it is an open question whether or not a propositionalization-like approach is more effective than an approach that closer resembles traditional ILP systems. In the following we investigate this empirically on the Yoshida dataset. We compare six biases:

- *Unsorted MinFreq*. This generates all substructures that occur more frequently than a predefined threshold in the dataset. The rules are added in the order as output by FreeTreeMiner, e.g. sorted according to the canonical form that is used in FreeTreeMiner.
- *Sorted MinFreq*. This is the same as the preceding bias, but the rules are ordered according to the size (i.e. number of atoms) of the substructures. The idea is that smaller substructures cover more instances and are thus more likely to be informative than large substructures that cover only a few molecules.
- *Maximum dispersion*. This bias is designed to maximize the variety and dispersion of the rules. To do so, it rates each generated rule according to a *dispersion score* that measures to which degree a rule classifies differently than the other rules. The calculation of the dispersion score is as follows: for each pair of training instances $(x_i, x_j)$, the bias calculates $s_{ij}$, the number of rules which assign the same label to both instances in the pair. Then, the dispersion score of a new rule is the sum of the $s_{kl}$ for those instance-pairs $(x_k, x_l)$ which are assigned the same label by the new rule. Hence, the dispersion score is a measure of how many instance pairs are classified in a similar way by the other rules. The bias proceeds as follows: first, it generates all rules that check for the existence of one atom substructures. Then, it iteratively extends this rule set by refining that rule with the lowest dispersion score, that is, the rule that classifies in the most different manner. It stops as soon as a maximum number of rules are built.
- *Greedy*. This bias aims at refining those rules that have proven to be informative. Similar to the preceding approach, it starts with a rule set that checks for the existence of all the elements. Then, it iteratively refines the rule with the largest absolute weight in the current model.
- *Maximum dispersion MinFreq*. This is the same as maximum dispersion, but it generates only those refinements that occur more frequently than a predefined threshold in the dataset.
- *Greedy MinFreq*. This is the same as greedy with a minimum frequency threshold.

These biases fall into different categories when categorized according to the scheme in Sect. 3.4. Unsorted and sorted MinFreq are in the second, the two maximum dispersion biases in the third, and greedy and greedy MinFreq fall into the fifth category. Thus, these biases can be regarded to span the range between propositionalization and classical ILP approaches. Figure 3 gives the results of our experiments for the Yoshida dataset. The left side denotes the predictive accuracy as estimated by tenfold cross-validation in relation to the number of rules that have been generated. The right side gives the average training error. A good bias generates informative rules first, so that the maximal predictive accuracy is reached after a few iterations and the rule generation loop can stop early without sacrificing predictive accuracy. According to this criterion, maximum dispersion MinFreq performs best, followed by sorted and unsorted MinFreq. In this and similar experiments (not reported here) we found that greedy approaches and biases without minimum frequency constraints fail to capture much of the relevant information, while sorting by size and dispersion tends to improve predictive accuracy.

## 5 Discussion and conclusion

In the paper, we introduced a new margin-based approach to first-order rule learning. The approach and in particular the quantity used for optimization, MMV, give theoretically appealing answers to many of the challenges in first-order rule induction, e.g., the computational complexity of optimizing a rule set, and overfitting avoidance. Moreover, it provides several possibilities for adjusting the learning algorithm to the problem: First, the degree to which the weights are distributed among the rules can be tuned by a parameter. Second, the effectiveness of the error bound for capacity control can be adapted as needed. Third, various rule generation strategies can be explored, depending on the way information is used to generate the next rule in forward selection, to be included in the model. This also implies that prior knowledge can be incorporated easily: Features assumed to be useful could be generated earlier than features just included for completeness. Fourth, RUMBLE offers specific plugins for various types of data, such as logic, graphs, and mathematical terms. In experiments, we investigated the overfitting behavior, the performance compared to margin-based ILP approaches using kernels, and the effectiveness of various search strategies. The latter experiments highlight an interesting aspect of propositionalization and relational learning, namely the number of features needed to achieve a certain predictive performance. In future work, we are planning to investigate this issue of “feature efficiency” of various feature generation strategies in more detail.

## Appendix: Proofs

*Proof of Theorem 1* We assume that the training data is given as a $m \times n$ matrix $X \in \mathbb{R}^{m \times n}$. Each of the $m$ row vectors represents an instance $x_i := (x_{i1}, \ldots, x_{in})$ and each of the $n$ columns represents a rule. When given a weight vector $w \in [-1; 1]^n$, we are interested in the distribution of the empirical margins $y_i x_i w$. If a training set contains a negative instance (i.e. $y_i = -1$) one can simple multiply $y_i$ and $x_i$ with $-1$ to gain a positive training instance with the same margin. Therefore, we assume without loss of generality that $y_i = 1$ for all $1 \leq i \leq m$ and omit the $y_i$ in the following. To compute the MMV, we need the (unweighted) empirical mean vector $\hat{\mu} \in \mathbb{R}^n := \frac{1}{m} X^T \mathbf{1}_m$ and the (unweighted) empirical covariance matrix $\hat{\Sigma} \in \mathbb{R}^{n \times n} := \frac{1}{m-1}(X^T X - \frac{1}{m} X^T \mathbf{1}_m \mathbf{1}_m^T X)$, where $\mathbf{1}_m$ denotes the vector containing $m$ ones. Later on, we will find it convenient to compute the individual entries $\hat{\sigma}_{ij}$ in $\hat{\Sigma}$ by $\hat{\sigma}_{ij} = \frac{1}{m(m-1)} \sum_{k<l} (x_{ki} - x_{li})(x_{kj} - x_{lj})$. The corresponding true quantities are $\mu := \mathbf{E}\hat{\mu}$, $\Sigma := \mathbf{E}\hat{\Sigma}$

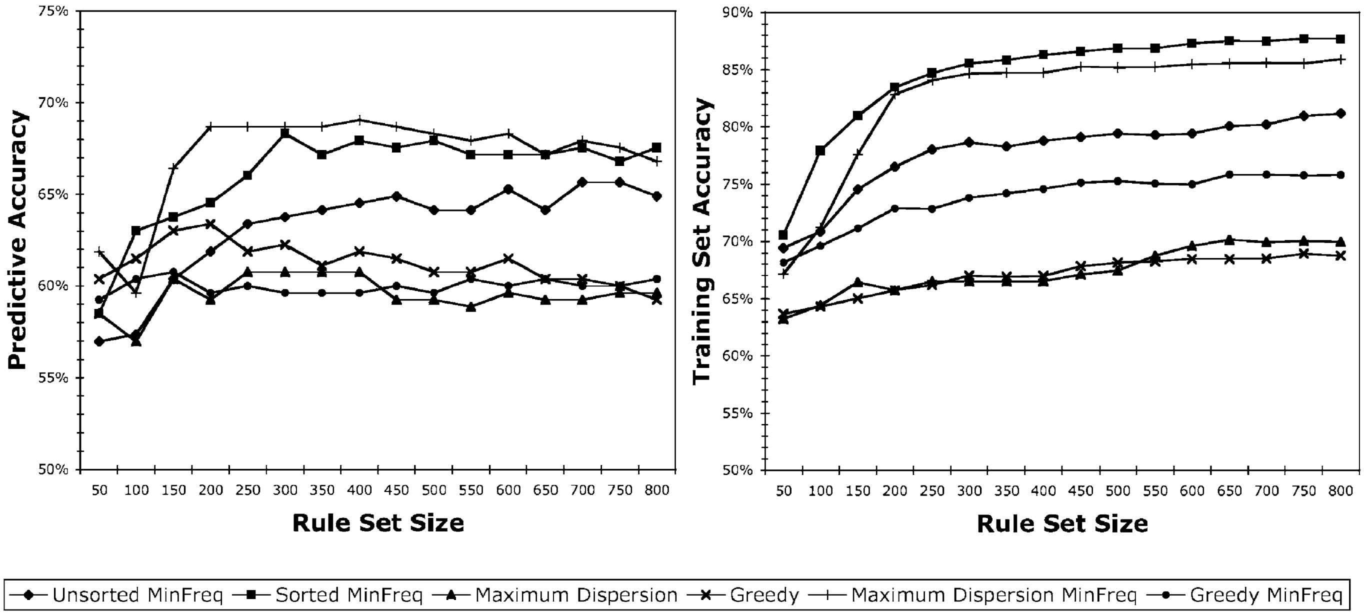

**Fig. 3** The predictive accuracy according to tenfold cross-validation (*left*) and training set accuracy over all folds (*right*) for six biases on the Yoshida data set depending on the number of generated rules

and $\sigma_{ij} := \mathbf{E}\hat{\sigma}_{ij}$. The empirical MMV $\hat{\gamma}_w$ can be defined as $\hat{\gamma}_w := \hat{\mu}^T w - w^T \hat{\Sigma} w$, and the true MMV is just the expectation over all training sets $\gamma_w := \mathbf{E}_{X\sim D}\hat{\gamma}_w$. We would like to find an upper bound of the estimation risk $\mathbf{Pr}[\sup_{w\in\mathcal{B}_1^n}(\hat{\gamma}_w - \gamma_w) \geq \varepsilon]$ that the empirical MMV differs from its expectation by more than $\varepsilon$ for the worst possible weight vector. First of all, observe that

$$\begin{aligned}
\sup_{w\in\mathcal{B}_1^n}(\hat{\gamma}_w - \gamma_w) &= \sup_{w\in\mathcal{B}_1^n}((\hat{\mu}-\mu)^T w - w^T(\hat{\Sigma}-\Sigma)w) \\
&= \sup_{w\in\mathcal{B}_1^n}\left(\sum_{k=1}^n w_k(\hat{\mu}_k - \mu_k) - \sum_{k,l=1}^n w_k w_l(\hat{\sigma}_{kl} - \sigma_{kl})\right) \\
&\leq \sup_{w\in\mathcal{B}_1^n}\left(\sum_{k,l=1}^n w_k w_l[(\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})]\right) \qquad (4) \\
&\leq \sup_{v\in[-1;1]^{n\times n}}\left(\sum_{k,l=1}^n v_{kl}[(\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})]\right) \qquad (5) \\
&\leq \sup_{1\leq k,l\leq n}((\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})). \qquad (6)
\end{aligned}$$

Inequality (4) holds because the $w_l$ sum up to at most one, (5) is a relaxation of the condition in the supremum, and (6) is due to the fact that a linear function on a convex hull reaches its optimum at one of the vertices. Then, we can upper-bound the estimation risk as follows, where $s > 0$ is a free parameter that can be tuned to make the inequality sharper later on:

$$\begin{aligned}
\mathbf{Pr}\Big[\sup_{w\in\mathcal{B}_1^n}(\hat{\gamma}_w - \gamma_w) \geq \varepsilon\Big] &= \mathbf{Pr}\Big[\sup_{w\in\mathcal{B}_1^n} e^{s(\hat{\gamma}_w - \gamma_w)} \geq e^{s\varepsilon}\Big] \\
&\leq e^{-s\varepsilon}\mathbf{E}\Big[\sup_{w\in\mathcal{B}_1^n} e^{s(\hat{\gamma}_w - \gamma_w)}\Big] \qquad (7) \\
&\leq e^{-s\varepsilon}\mathbf{E}\Big[\sup_{k,l} e^{s[(\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})]}\Big] \qquad (8) \\
&\leq e^{-s\varepsilon}\sum_{k,l=1}^n \mathbf{E}[e^{s[(\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})]}]. \qquad (9)
\end{aligned}$$

Inequality (7) is an application of Markov's inequality, (8) is given above, and (9) follows because $e^x > 0$ and the sum over all $k, l$ includes the summand that achieves the supremum.

Now, define the set of random variables

$$Z^{(k,l)} := (\hat{\mu}_k - \mu_k) - (\hat{\sigma}_{kl} - \sigma_{kl})$$

depending on $X$. If we can find an upper bound for the $Z^{(k,l)}$, (9) gives us an upper bound of the estimation risk. We follow the method by McDiarmid (1989) and write $Z^{(k,l)}$ as a sum of martingale differences to find such a bound. Let

$$V_r^{(k,l)} := \mathbf{E}[Z^{(k,l)} \mid x_1, \ldots, x_r] - \mathbf{E}[Z^{(k,l)} \mid x_1, \ldots, x_{r-1}].$$

Then, $Z^{(k,l)} = \sum_{r=1}^m V_r^{(k,l)}$. Replacing the $\hat{\sigma}_{ij}$ and $\hat{\mu}_i$ in $V_r^{(k,l)}$ with their definitions yields:

$$V_r^{(k,l)} = \frac{1}{m}(x_{rk} - \mathbf{E}x_{rk}) \tag{10}$$

$$- \frac{1}{m(m-1)} \sum_{j \neq r} [\mathbf{E}[\hat{\sigma}_{rj}^{kl} \mid x_1, \dots, x_r] - \mathbf{E}[\hat{\sigma}_{rj}^{kl} \mid x_1, \dots, x_{r-1}]] \tag{11}$$

where $\hat{\sigma}_{ij}^{kl} := (x_{ik} - x_{jk})(x_{il} - x_{jl})$. It turns out that the random variable $V_r^{(k,l)}$ can take only values in the interval $[c; c + \frac{6}{m}]$ for some constant $c$. The first part in (10) depends only on $x_{rk} \in [-1; 1]$, so it can differ from its expectation by at most $\frac{1}{m}$. In the second part (11) each summand depends only on $x_{rk}$ and $x_{rl}$, so that the $\hat{\sigma}_{ij}^{kl}$ can take values that differ by at most four. Since there are $m-1$ summands, the part in (11) changes within a range of at most $\frac{4}{m}$ depending on $X$. Also, since the conditional expectation $\mathbf{E}[V_r^{(k,l)} \mid x_1, \dots, x_{r-1}] = 0$, we can apply a conditional version of Hoeffding's inequality (Hoeffding 1963):

**Theorem 2** (Hoeffding's inequality) *Let $X$ be a random variable with $\mathbf{E}X = 0, a \leq X \leq b$. Then, for $s > 0$,*

$$\mathbf{E}\big[e^{sX}\big] \leq e^{s^2(b-a)^2/8}.$$

Applying this inequality conditionally to the $V_r^{(k,l)}$, we obtain:

$$\mathbf{E}[e^{sV_r^{(k,l)}} | x_1, \dots, x_{r-1}] \leq e^{\frac{1}{8}s^2(\frac{6}{m})^2}. \tag{12}$$

Since the instances are drawn independently from each other, we can apply (12) iteratively on all summands in $Z^{(k,l)} = \sum_{r=1}^m V_r^{(k,l)}$:

$$\begin{aligned}
\mathbf{E}[e^{sZ^{(k,l)}}] &= \mathbf{E}[e^{s\sum_{i=1}^{m-1} V_i^{(k,l)}} \cdot e^{sV_m^{(k,l)}}] \\
&= \mathbf{E}[e^{s\sum_{i=1}^{m-1} V_i^{(k,l)}} \cdot \mathbf{E}[e^{sV_m^{(k,l)}} | x_1, \dots, x_{m-1}]] \\
&\leq \mathbf{E}[e^{s\sum_{i=1}^{m-1} V_i^{(k,l)}} \cdot e^{\frac{1}{8}s^2(\frac{6}{m})^2}] \\
&\leq \dots \leq e^{m\frac{1}{8}s^2(\frac{6}{m})^2} = e^{\frac{9}{2}s^2\frac{1}{m}}.
\end{aligned}$$

Finally, we may plug this result into (9):

$$\begin{aligned}
\mathbf{Pr}\Big[\sup_{w \in \mathcal{B}_1^n} (\hat{\gamma}_w - \gamma_w) \geq \varepsilon\Big] &\leq e^{-s\varepsilon} n^2 e^{\frac{9}{2}s^2\frac{1}{m}} \\
&\leq n^2 e^{-\frac{1}{9}\varepsilon^2 m + \frac{1}{18}\varepsilon^2 m} \quad \left(\text{by choosing } s = \frac{1}{9}\varepsilon m\right) \\
&= n^2 e^{-\frac{1}{18}\varepsilon^2 m}.
\end{aligned}$$

Choosing

$$\varepsilon = \sqrt{\frac{18(2\ln n + \ln\frac{1}{\delta})}{m}}$$

yields

$$\mathbf{Pr}\Bigg[\sup_{w \in \mathcal{B}_1^n} (\hat{\gamma}_w - \gamma_w) \geq \sqrt{\frac{18(2\ln n + \ln\frac{1}{\delta})}{m}}\Bigg] \leq \delta.$$

This is equivalent to the statement in Theorem 1. □

*Proof of Lemma 1* The Chebyshev inequality states that

$$\varepsilon_w = \Pr[\mu_w \leq 0] \leq \frac{\mathrm{Var}[\mu_w]}{\mathrm{Var}[\mu_w] + \mu_w^2} = \frac{\mu_w - \gamma_w}{\mu_w - \gamma_w + \mu_w^2} \tag{13}$$

Denote the right hand side of (13) by $f(\mu_w, \gamma_w)$. To yield the final result, we determine $\mu'_w := \mathrm{argmax}_{\mu_w} f(\mu_w, \gamma_w)$ maximizing the right hand side subject to the constraint that $\gamma_w \leq \mu_w \leq 1$ (since the variance is always positive). To find this optimum, we set the derivative to zero:

$$\frac{\partial f}{\partial \mu_w} = \frac{-\mu_w^2 + 2\mu_w \gamma_w}{(\mu_w - \gamma_w + \mu_w^2)^2} = 0.$$

The only positive solution to this equation is $\mu'_w = 2\gamma_w$. So, for $\gamma_w \leq 0.5$ we yield:

$$\varepsilon_w \leq f(2\gamma_w, \gamma_w) = \frac{\gamma_w}{4\gamma_w^2 + \gamma_w}.$$

If $\gamma_w > 0.5$, the optimal value of $\mu_w$ is violating the constraint that $\mu_w \leq 1$. As $f$ is increasing until $2\gamma_w$, the optimal margin is thus 1 and we yield:

$$\varepsilon_w \leq f(1, \gamma_w) = \frac{1 - \gamma_w}{2 - \gamma_w}.$$ □

## References

Ben-David, S., Eiron, N., & Long, P. M. (2003). On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, *66*(3), 496–514.

Cohen, W., & Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proceedings of the sixteenth national conference on artificial intelligence (AAAI-99)* (pp. 335–342). Menlo Park: AAAI Press.

De Raedt, L. D. (1997). Logical settings for concept-learning. *Artificial Intelligence*, *95*(1), 187–201.

Dias, A. M. (2006). CxProlog. http://ctp.di.fct.unl.pt/~amd/cxprolog/.

Friedman, J. H., & Popescu, B. E. (2005). *Predictive learning via rule ensembles* (Technical report). Stanford University.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory*. Cambridge: MIT Press.

King, R., & Srinivasan, A. (1995). Relating chemical activity to structure: an examination of ILP successes. *New Generation Computing, Special issue on Inductive Logic Programming*, *13*(3–4), 411–434.

Kramer, S., Lavrac, N., & Flach, P. (2001). Propositionalization approaches to relational data mining. In S. Dzeroski & N. Lavrac (Eds.), *Relational Data Mining* (pp. 262–291). Berlin: Springer.

Landwehr, N., Passerini, A., De Raedt, L., & Frasconi, P. (2006). kFOIL: Learning simple relational kernels. In *Proceedings of the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference*, Boston, Massachusetts, USA, 16–20 July 2006. Menlo Park: AAAI Press.

Li, H., Yap, C. W., Ung, C. Y., Xue, Y., Cao, Z. W., & Chen, Y. Z. (2005). Effect of selection of molecular descriptors on the prediction of blood-brain barrier penetrating and nonpenetrating agents by statistical learning methods. *Journal of Chemical Information and Modeling*, *45*(5), 1376–1384.

McDiarmid, C. (1989). On the method of bounded differences. In *London mathematical society lecture note series: Vol. 141. Surveys in combinatorics* (pp. 148–188). Cambridge: Cambridge Univ. Press.

Muggleton, S., Lodhi, H., Amini, A., & Sternberg, M. J. E. (2005). Support vector inductive logic programming. In A. G. Hoffmann, H. Motoda, & T. Scheffer (Eds.), *Discovery science* (pp. 163–175). New York: Springer.

Popescul, A., & Ungar, L. (2003). Statistical relational learning for link prediction. In *IJCAI workshop on learning statistical models from relational data*.

Rückert, U., & Kramer, S. (2004). Frequent free tree discovery in graph data. In H. Haddad & A. Omicini (Eds.), *Proceedings of the ACM symposium on applied computing* (pp. 564–570). New York: ACM.

Rückert, U., & Kramer, S. (2006). A statistical approach to rule learning. In *Machine learning, proceedings of the twenty-third international conference (ICML 2006)* (pp. 785–792), Pittsburgh, Pennsylvania, USA, 25–29 June 2006. New York: ACM.

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*(1–2), 277–299.

Woźnica, A., Kalousis, A., & Hilario, M. (2005). Kernels over relational algebra structures. In T. B. Ho, D. Cheung, & H. Liu (Eds.), *Lecture notes in computer science: Vol. 3518. PAKDD* (pp. 588–598). Berlin: Springer.

Yoshida, F., & Topliss, J. (2000). QSAR model for drug human oral bioavailability. *Journal of Medicinal Chemistry*, *43*, 2575–2585.