

---

# A Statistical Approach to Rule Learning

---

Ulrich Rückert  
Stefan Kramer

RUECKERT@IN.TUM.DE  
KRAMER@IN.TUM.DE

Technische Universität München, Institut für Informatik/I12, Boltzmannstr. 3, D-85748 Garching b. München, Germany

## Abstract

We present a new, statistical approach to rule learning. Doing so, we address two of the problems inherent in traditional rule learning: The computational hardness of finding rule sets with low training error and the need for capacity control to avoid overfitting. The chosen representation involves weights attached to rules. Instead of optimizing the error rate directly, we optimize for rule sets that have large margin and low variance. This can be formulated as a convex optimization problem allowing for efficient computation. Given the representation and the optimization procedure, we effectively yield weighted clauses in a CNF-like representation. To avoid overfitting, we propose a model selection strategy that utilizes a novel concentration inequality. Empirical tests show that the system is competitive with existing rule learning algorithms and that its flexible learning bias can be adjusted to improve predictive accuracy considerably.

## 1. Introduction

One of the most popular learning schemes in Machine Learning is rule induction. There is plenty of research on rule induction, some of it dating back as far as the sixties. However, unlike decision trees, neural networks and SVMs, most established rule learning systems are hard to analyze from a statistical point of view. This is due to the combinatorial complexity inherent in the setting and the use of heuristics rather than statistically motivated principles in most rule learning algorithms. In the following, we frame

rule learning as a statistical problem. As the combinatorics of boolean formulae is hard to handle with statistical means, we avoid the traditional interpretation of rule sets as formulae in disjunctive normal form. Instead, we follow the lead of rule learning systems such as SLIPPER (Cohen & Singer, 1999) and LRI (Weiss & Indurkha, 2000) and use sets of weighted rules. In this representation, a weight is assigned to each rule. To classify an instance, the algorithm adds up the weights of the rules whose condition is met. If the sum is positive, the system predicts the positive class, if it is negative, it predicts the negative class. For example, a typical rule set might look like this:

*if sky=sunny and humidity=normal then 0.4*

*if airtemp=cold then -0.35*

*if sky=cloudy and forecast=change then -0.15*

This rule set would classify the instance (sunny, normal, cold, change) as positive, because it meets the conditions of the first and the second rule, so the sum of weights is 0.05, which is greater than zero.

The main goal of rule induction is to find a preferably small set of (weighted) rules with high predictive accuracy. In the context of statistical learning, this setting is usually modeled as follows: The user draws an i.i.d. sample of labeled examples from a fixed, but unknown distribution. The user is interested in identifying the hypothesis in a certain hypothesis class, which most accurately predicts the label from the instance. Since the original distribution is unknown, the problem cannot be solved accurately. Instead, one resorts to finding the hypothesis minimizing the empirical error on the training sample. As we deal with the hypothesis class of rule sets, we are facing two main challenges: The first problem is that solving the empirical risk minimization problem is intractable for the class of rule sets. In the next section we propose optimizing the empirical error not directly, but

---

Appearing in *Proceedings of the 23<sup>rd</sup> International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

through a related quantity in order to avoid the computational complexity. We present the three quantities *soft-margin loss*, *empirical margin* and *margin minus variance* that allow for tractable optimization procedures. The optimization of each of these quantities also minimizes the empirical error. Soft-margin loss minimization and margin minus variance maximization also tend to induce small rule sets, as opposed to the larger sets favored by ensemble-based approaches. This makes them especially suited for rule learning. The second problem is the need for capacity control: If one selects a large hypothesis class to choose the rule set from, the system is likely to overfit. Likewise, if the hypothesis class is too small, the algorithm might underfit. Finding the “right” size of hypothesis class is an important part of model selection. We address it in section 2.3 using two new concentration inequalities to estimate the structural risk of a given hypothesis class. In section 3, we put the pieces together to form a practically useful rule learning system. The system is based on a simple iterative model selection strategy. It iteratively adds new rules to an initially empty set of rules and keeps track of the subset of rules that optimizes the empirical risk (as measured on the training set) and the structural risk (as estimated by the risk bound). In section 4, we report on empirical results of the system on benchmark data sets and show how to adjust the algorithm’s bias in order to leverage background knowledge. We conclude in section 5.

## 2. Rule Learning as Model Selection

In the following, we present a statistically motivated approach to rule learning. In particular, we frame rule learning as a model selection problem. Traditional rule learning systems feature a *preference bias*: Theoretically, they are able to generate all possible rule sets, but small and informative rule sets are preferred. In contrast, we pursue a model selection approach with a *restriction bias*. The proposed system keeps an increasing class of potential rules during the learning process. At any time, the algorithm is able to output only rule sets containing rules that are taken from this class. Thus, the rule class serves in effect as a restriction bias upper-bounding the hypothesis complexity: The induced rule sets could theoretically be as complex as a rule set that contains all the rules in the class. In practice, however, the MMV criterion assigns weights only to a minority of the rules. Before we give more details, we elaborate on related work.

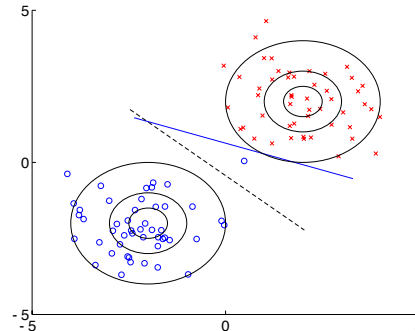


Figure 1. The maximum margin hyperplane (solid) and the margin minus variance hyperplane (dotted) for a small data set where positive and negative instances are drawn from two normal distributions with mean at  $(-2, -2)$  and  $(2, 2)$ .

### 2.1. Related Work

Most traditional rule learning algorithms are based on a separate-and-conquer strategy. They generate one new rule per main loop iteration and remove those training instances that are correctly classified by the new rule before proceeding in the next iteration. When combined with suitable pruning heuristics, this procedure allows for efficient and practically successful rule learning implementations (see (Fürnkranz, 1999) for a survey). The underlying representation language usually resembles DNF formulae: The individual rules are conjunctions of (possibly negated) literals, and a disjunction of rules forms a rule set.<sup>1</sup> Unfortunately, finding good rule sets in this representation is a hard discrete combinatorial problem and small changes can have large consequences. Thus, separate-and-conquer approaches are rather hard to analyze formally. As a consequence, it is difficult to understand on a formal level under which circumstances one heuristic should be preferred over another.

With the advent of ensemble methods a powerful new framework became available for the analysis of rule learning algorithms. In this setting, a rule set is typically seen as an ensemble of rules, i.e., one assigns a weight to each rule and uses a voting scheme for prediction. Typical ensemble-based rule learning systems are SLIPPER (Cohen & Singer, 1999), LRI (Weiss & Indurkha, 2000) and RuleFit (Friedman & Popescu, 2005). According to the theory on ensembles it is possible to avoid overfitting by focusing on large dispersion (i.e., low correlation) between the ensemble mem-

<sup>1</sup>The actual representation differs from learner to learner, but the underlying learning problems are conceptually very similar.

bers (as in bagging). However, the resulting ensembles tend to be rather large. This somewhat contradicts the goal in rule learning, where small, comprehensible rule sets are usually considered to be more desirable. For instance, Friedman’s RuleFit generates 130 rules on the australian data set, considerably more than traditional rule learners. Another well-motivated approach is to aim for ensembles with large margin (as in boosting or SVMs). In this case, overfitting may happen, if the rule set gets too large and thus some form of capacity control is necessary. In the following, we will present a model selection strategy that is designed to work well in typical rule learning settings.

## 2.2. Learning Weighted Rule Sets

In order to give a more formal definition of the setting, we need some basic definitions. First of all, we follow the usual convention and assume that the instances are drawn i.i.d. according to a fixed but unknown distribution  $D$ .  $D$  ranges over  $\mathcal{X} \times \mathcal{Y}$ , where  $\mathcal{X}$  is the set of all possible instances and  $\mathcal{Y} := \{-1, 1\}$  contain the target labels. The values -1 and 1 represent the negative and positive class respectively. A sample  $X = \{x_1, \dots, x_m\}$  of size  $m$  is drawn. Furthermore, we assume we already have a (possibly infinite) reservoir of rules  $R = \{r_1, r_2, \dots\}$ , where a rule  $r_j : \mathcal{X} \rightarrow [-1, 1]$  assigns a value between -1 and 1 to each instance. A typical rule could be “assign +1, if the color is red, and -1 otherwise” or “assign +1, if the size is greater than 1.2 and -1 otherwise”. We will later present a scheme to enumerate the rules in the reservoir 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  $x_i := (x_i(1), x_i(2), \dots, x_i(n))^T$ . Likewise, a weighted rule set can be given by a weight vector  $p \in [-1, 1]^n$ . An individual rule set  $p$  assigns class label  $\text{sgn}(p^T x_i)$ , so that instance  $x_i$  is positive, if  $\sum_{j=1}^n p_j x_i(j) \geq 0$  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  $p$  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.

In classification one usually uses the *zero-one loss*  $l(p^T x, y) = \mathbf{I}[\text{sgn}(p^T x) \neq y]$  to define the *empirical error*  $\hat{\varepsilon}_p := \frac{1}{m} \sum_{i=1}^m l(p^T x_i, y_i)$  and the *true error*

$\varepsilon_p := \mathbf{E}_{(x,y) \sim D} l(p^T x, y)$ . We are interested in finding a weighted rule set  $p$  that minimizes the true error  $\varepsilon_p$ . 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  $p$  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. 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  $p \in \{x \mid \|x\| = 1\}$  (as scaling  $p$  with a factor does not change the classification) and search for a  $p$  that maximizes the margin (or the soft margin) to the nearest training instances.<sup>2</sup> In the following, we will present three different quantities that can be used as a substitute for the empirical error and that allow for an efficient optimization procedure.

The first quantity is simply the soft-margin loss as used in the SVM. Optimizing for a *large margin* is theoretically well-founded and the resulting quadratic programming problem can be solved efficiently. For rule learning, we are looking for an explicit representation of the rule set, not one that is implicitly defined by a kernel. Thus, we do not depend on the applicability of the kernel trick and can use other hypothesis spaces than the weight vectors of unit length  $p \in \{x \mid \|x\| = 1\}$ . In particular, for ensembles it is much more common to demand that the absolute weights in  $p$  sum to one, i.e., to use the 1-norm instead of the 2-norm. This leads to the so-called *1-norm support vector machine* (Zhu et al., 2004):

$$\begin{aligned} & \underset{p}{\text{minimize}} \sum_{i=1}^m \left[1 - p^T x_i\right]_+ & (1) \\ & \text{subject to } \|p\|_1 = \sum_{j=1}^n |p_j| = s \end{aligned}$$

where  $s$  is a free tuning parameter. Unlike the 2-norm SVM, the 1-norm SVM sets most weights to zero and assigns non-zero weights only to a few highly relevant rules. This behavior matches well with our purposes in rule learning.

The second quantity is the *empirical margin*. One problem of the SVM’s large margin approach is that

<sup>2</sup>Of course, the actual optimization problem is formulated differently.

the classifier depends only on the support vectors and ignores the other data. If – as in figure 1 – the support vectors are not very representative of the underlying distribution, the classifier performs worse than necessary. 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. More formally: Given an instance  $(x, y)$  let  $\mu_p(x, y) := p^T x \cdot y$  denote the *margin*. The margin is positive, if the instance is correctly classified by  $p$  and negative otherwise. Similar to true and empirical error, one can define the *empirical margin*  $\hat{\mu}_p := \frac{1}{m} \sum_{i=1}^m \mu_p(x_i, y_i)$  and the *true margin*  $\mathbf{E}_{(x,y) \sim D} \mu_p(x, y)$ . So, a straightforward approach is to maximize  $\hat{\mu}_p$  subject to  $\|p\|_1 = 1$ . However, it is easy to see that this procedure assigns the full weight to the single rule that agrees with the target class on most examples. It sets weight zero to all the other rules. In essence, aiming for maximum empirical margin with the 1-norm constraint is equivalent to selecting the single best rule. We therefore resort to using the 2-norm and minimize for large  $\hat{\mu}_p$  subject to  $\|p\|_2 = 1$ . This sets almost all weights to non-zero values. The main advantage of this criterion is the fact that its solution can be calculated in closed form (see section 3). From a theoretical point of view, the following lemma guarantees that optimizing for large  $\mu_p$  also minimizes the true error  $\varepsilon_p$ :

**Lemma 1.** *Let  $S$  be drawn according to  $D$  and let  $p$  be a weight vector with true error  $\varepsilon_p$  and true margin  $\mu_p$ . Then:*

$$\varepsilon_p \leq 1 - \mu_p \quad (2)$$

(Proof omitted)

As the lemma is valid for all possible distributions, it also applies to every empirical distribution induced by a training set. Hence, the empirical error can also be upper-bounded by the empirical margin. Thus, optimizing  $\hat{\mu}$  can be seen as a computationally feasible relaxation of the infeasible direct empirical risk minimization problem. In particular, if  $\hat{\mu}_p$  reaches the largest possible value one, the empirical error is guaranteed to be zero.

The third approach tries to improve on the second one by also incorporating the variance. One can estimate the sample variance of the margins from a training set; the *empirical variance* is  $\hat{\sigma}_p := \frac{1}{m-1} \sum_{i=1}^m (\mu_p(x_i, y_i) - \hat{\mu}_p)^2$ , the *true variance* is  $\sigma_p := \mathbf{E}_{(x,y) \sim D} (\mu_p(x, y) - \mu_p)^2$ . Using this notation, one can look for weight vectors  $p$  that maximize the empirical margin but mini-

mize the empirical variance.

$$\begin{aligned} & \underset{p}{\text{maximize}} \hat{\mu}_p - \hat{\sigma}_p & (3) \\ & \text{subject to } \|p\|_1 = \sum_{j=1}^n |p_j| = 1 \end{aligned}$$

We call the optimization function *margin minus variance* (MMV) and denote it by  $\hat{\gamma}_p := \hat{\mu}_p - \hat{\sigma}_p$ . It is easy to see that maximizing  $\hat{\gamma}$  is a quadratic optimization problem and can therefore be solved efficiently. In figure 1, the dotted line denotes the maximum-MMV hyperplane, while the solid line indicates the maximum-soft-margin hyperplane. Even though the MMV hyperplane misclassifies one instance in the training set, it obviously approximates the decision boundary for the underlying data distribution (the bisecting line at the origin) better than the soft-margin hyperplane. Just as the 1-norm SVM, MMV sets most weights to zero and assigns non-zero weights only to the best few rules.

Another nice property is that  $\hat{\gamma}_p$  is an unbiased estimator of the *true margin minus variance*  $\gamma_p := \mu_p - \sigma_p$  and this quantity can be used to upper-bound the true error. This is intuitively intriguing: A hyperplane that features a large true margin and a low true variance should have a low true error, because most instances are clearly classified correctly (large margin) and there are only a few exceptions (low variance). The following lemma quantifies this:

**Lemma 2.** *Let  $S$  be drawn according to  $D$  and let  $p$  be a hyperplane with true error  $\varepsilon_p$  and true MMV  $\gamma_p$ . Then:*

$$\varepsilon_p \leq \begin{cases} \frac{\gamma_p}{4\gamma_p^2 + \gamma_p} & \text{if } \gamma_p \leq 0.5 \\ \frac{1 - \gamma_p}{2 - \gamma_p} & \text{if } \gamma_p > 0.5 \end{cases} \quad (4)$$

(Proof omitted)

Just as lemma 1 did for the empirical margin, this lemma ensures that  $\hat{\gamma}$  can be seen as a computationally feasible relaxation of the infeasible direct empirical risk minimization problem. Again, if  $\hat{\gamma}_p$  reaches the largest possible value one, the empirical error is guaranteed to be zero.

### 2.3. Capacity Control for Rule Learning

In the preceding section we presented three quantities that can be used to determine weights for a class of predefined rules. Obviously, the success of this algorithm depends to a large degree on the choice of this class, and, in particular, on its size. If the class is too small, the algorithm will most likely underfit, if it is too large, it will overfit. The standard approach

to tackle this dilemma is to start with a small class, then iteratively increase the size of the class. In each step, one determines for each class the hypothesis that explains the training set best, and an estimate of the structural risk of the class. The best hypothesis class is the one which minimizes the sum of empirical risk and structural risk. Similar approaches include structural risk minimization or model selection based on Rademacher risk bounds.

Unfortunately, most of the established model selection strategies do not work well in our setting. The first problem is, that often only a few large hypothesis classes are used, so that the model selection is rather coarse. For instance, a popular scheme is to use cascading classes of polynomial kernels with increasing exponent in SVMs. In rule learning, though, even very small hypothesis classes can overfit. For example the hypothesis class of 1-term DNF formulae is a subset of a linear kernel (i.e., a polynomial kernel with exponent one), and nevertheless, empirical risk minimization overfits with this hypothesis class on the UCI hearth data set (Rückert & De Raedt, 2006). Apparently, in rule learning one needs more fine-grained and (at least initially) slower increasing hypothesis class sizes.

The risk bounds that are used for model selection usually deal directly with the empirical error. This does not match very well with the approaches taken in the preceding section. First of all, the empirical error is a discrete quantity and often does not change in steps from one hypothesis class to the other, making it hard to handle. The quantities presented in section 2.2 provide a more fine-grained estimate of the empirical risk. In the following, we present two concentration inequalities that work directly on the empirical margin and the MMV. It seems to be more sensible to use those risk bounds depending on the the actual quantities that are optimized instead of the (possibly fluctuating) empirical error.

The following theorem provides a bound for the true margin depending on the empirical margin

**Theorem 1.** *Let  $D$  be a fixed unknown distribution over  $[-1, 1]^n \times \{-1, +1\}$  and  $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$  be a sample of size  $m$  drawn i.i.d. from  $D$ . Let  $\mathcal{P} := \{p \in [-1, 1]^n \mid \sum_{j=1}^n |p_j| = 1\}$  be the space of weight vectors of size  $n$ . Then for all  $\delta > 0$  and all  $p \in \mathcal{P}$ :*

$$\Pr_{S \sim D} \left[ \mu_p \geq \hat{\mu}_p - \sqrt{\frac{2}{m} \ln \frac{2n}{\delta}} \right] \geq 1 - \delta \quad (5)$$

This theorem provides the same service for the MMV:

---

**Algorithm 1** The rule learning algorithm.

---

```

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

```

---

**Theorem 2.** *Let  $D$ ,  $S$  and  $\mathcal{P}$  be as above. Then for all  $\delta > 0$  and all  $p \in \mathcal{P}$ :*

$$\Pr_{S \sim D} \left[ \gamma_p \geq \hat{\gamma}_p - \sqrt{\frac{18}{m} (2 \ln 2n + \ln \frac{1}{\delta})} \right] \geq 1 - \delta \quad (6)$$

The proofs of the theorems are omitted due to space constraints. The inequalities basically state that the true margin or MMV is with high probability larger than its empirical counterpart minus a risk term that depends logarithmically on the size of the class of rules. A straightforward application of the preceding lemmas also give bounds on the true error. The inequalities provide a pessimistic estimate of the structural risk that is involved with using larger classes of rules. The constants in the the two risk terms are too pessimistic for most real world distributions and applicable only for worst-case analysis; for practical model selection applications one should use smaller constants.

The bounds depend only on the number of rules that might be used to generate a weighted rule set, not on any structural properties of the rules. While it is tempting to provide a fixed default order of rule classes, best results can be achieved if the classes are selected in an application-specific manner. For example, if the user has some information on which kind of rules might be more informative than others, she should start with a rule class containing the informative rules and add the presumably non-informative rules only later on. This is an easy, yet flexible and powerful way to adjust the learner's bias and incorporate background knowledge, without the need to adjust parameters or kernels. Section 4 contains a small empirical study on how to adjust the rule classes in order to boost predictive accuracy.

### 3. A Statistical Rule Learning System

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 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  $p^{(i)}$  optimizing the soft margin loss, the empirical margin or the empirical MMV on this data and calculates an upper bound on the corresponding true values given the empirical quantity and the number of rules. If there are no new rules available, the algorithm terminates the loop and returns the set of rules and the weight vector which achieved the best bound. 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.

For the 1-norm SVM, we reformulate equation 1 as a linear program:

$$\begin{aligned} & \underset{p}{\text{minimize}} && \sum_{i=1}^m c_i && (7) \\ & \text{subject to} && \forall i \ y_i p^T x_i - c_i \geq 1, \\ & && \sum_{j=1}^n |p_j| = s, \forall i \ c_i \geq 0 \end{aligned}$$

This can be solved using any of the established methods for linear programming. Optimizing for large empirical margin  $\hat{\mu}_p$  is even easier. Setting  $d = \frac{1}{m} \sum_{i=1}^m x_i$ , the problem can be stated as  $\hat{\mu}_{opt} = \min_p -d^T p$  subject to  $\|p\|_2^2 = 1$ . The corresponding Lagrangian is  $\mathcal{L}(p, \lambda) = -d^T p + \lambda(\|p\|_2^2 - 1)$  and setting its gradient  $\nabla_p \mathcal{L}(p, \lambda) = 0$  yields  $p_{opt} = \frac{1}{2\lambda} d$ . It follows from duality that  $\hat{\mu}_{opt} = \max_{\lambda} \mathcal{L}(\frac{1}{2\lambda} d, \lambda)$  and setting the derivative with regard to  $\lambda$  to zero we have  $\lambda_{opt} = \frac{1}{2} \|d\|_2$ . Thus,  $p_{opt} = \frac{d}{\|d\|_2}$  is a solution to this optimization problem.

We formulate the MMV optimization task as a standard quadratic programming problem. The MMV  $\hat{\gamma}$  contains two parts: the empirical margin  $\hat{\mu}$  is linear, while the quadratic part is given by the empirical covariance matrix. If the training set is given as a  $m \times n$  matrix  $T$  with  $t_{ij} := x_i(j)y_i$ , then the linear part is  $d := \frac{1}{m} (\vec{1}^T T)$  and the quadratic part is  $G = -\frac{2}{m-1} [(T^T T) - \frac{1}{m} (\vec{1}^T T)^T (\vec{1} T)]$ , where  $\vec{1}$  is a vector containing  $m$  ones. In order to remove the discontinuity in the absolute value of the original 1-norm-constraint, one can decompose each weight  $p_i$  in the weight vector  $p$  into a positive part  $p_i^+ \geq 0$  and a negative part  $p_i^- \geq 0$  so that  $p_i = p_i^+ - p_i^-$ . Using the notation  $p = (p_1^+, \dots, p_n^+, p_1^-, \dots, p_n^-)^T$  the optimization

problem becomes:

$$\begin{aligned} & \underset{p}{\text{maximize}} && \frac{1}{2} p^T \begin{pmatrix} G & -G \\ -G & G \end{pmatrix} p + p^T \begin{pmatrix} d \\ -d \end{pmatrix} \\ & \text{subject to} && \sum_{j=1}^n (p_j^+ + p_j^-) = 1, \forall j \ p_j^+, p_j^- \geq 0 \end{aligned}$$

This is a semidefinite quadratic programming problem with one equality and  $2n$  inequality constraints. 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.

In principle, one can use any arbitrary method to generate new rules. Ideally, one would like to generate only rules that are informative for the prediction task at hand. However, selecting the rules depending on the class labels in the training set can easily lead to overfitting.<sup>3</sup> It is a better idea to select the rules based on information about the structure of the instances without the class labels or on other background knowledge. For example, one could use information obtained in a previous clustering step to generate rules that describe certain characteristics of the clusters in the data. It is not clear how to devise a general scheme that generates “good” rules for all kinds of data sets. Instead, we found it more sensible to keep the rule generation process as generic and flexible as possible. In this way the user can adjust the rule generation depending on her knowledge of the particular domain. In the following we will present a rule generation strategy that resembles the approach taken in traditional rule learning systems. It appears to work well on typical attribute-value data sets as those taken from the UCI repository and is easy to adjust to more complex data, for example in multi-relational settings.

We start by generating rules that describe the original features found in the data set. For a nominal-valued feature, we generate one rule of the form “feature=value” for each possible value. We apply equal-frequency discretization using four thresholds and generate one rule per threshold. Each rule outputs +1, if its condition is met, -1 if it is not, and 0, if the value is missing. This gives a certain base set of rules encoding most information in the training sample. In a second step we apply refinement operators on the existing rules to generate more complex rules. One straightforward approach is to combine two rules by calcu-

<sup>3</sup>In the extreme case, one could generate just one rule that perfectly explains the training set. Of course, this would render the bound useless and lead to overfitting.

Data Set	PART	SLIP-	1-norm	emp.	MMV
	PER	SVM	margin		
australian	84.3	85.1	85.5	80.9	85.5
breastcancer	69.6	71.4	70.3	52.8	69.2
breastw	94.9	96.0	90.8	96.6	94.0
colic	84.4	84.6	81.5	80.7	82.9
diabetes	74.0	73.5	74.2	65.0	75.8
german	70.0	71.8	70.0	61.1	74.4
glassg2	80.0	83.1	61.3	68.1	80.4
heartc	78.5	79.3	75.9	79.5	85.1
hearth	80.5	79.2	78.6	78.9	81.6
heartstatlog	78.9	78.5	75.2	78.5	83.3
hepatitis	80.2	79.5	79.4	59.3	84.5
ionosphere	90.6	93.2	78.6	78.1	86.9
krvskp	99.3	59.4	68.3	77.7	91.2
labor	77.3	92.3	75.4	84.2	80.7
mushroom	100.0	100.0	88.7	97.3	91.8
sick	98.6	98.5	93.9	66.0	93.9
sonar	76.5	74.2	56.7	69.2	77.4
vote	95.9	94.1	95.6	89.2	95.6

Table 1. Results: percentage of correct classifications.

lating the maximum of two base conditions. This is motivated by the observation that aiming for a weight vector with large margin resembles a relaxed conjunction of the rules. Consequently, it makes sense to use the “or” (i.e., maximum) operation for the rules to induce a kind of relaxed conjunctive normal form (CNF).

#### 4. Experiments

In order to test the performance of the proposed system on real world data, we implemented a version in C++ on Linux. In practice, the overhead of optimizing for rules with very low weights is not necessary, as their weight does not contribute in a significant way. Thus, whenever the number of generated rules exceeds one hundred, we ignore the rule with the lowest weight assigned during the optimization step (but not during the rule refinement step). We apply the system to all two-class datasets from (Frank & Witten, 1998). We select the rule generation bias outlined above: nominal attributes are added as one “feature=value” rule for each feature value, continuous attributes are discretized into four bins. After those base rules, we build all combinations of two rules and form new rules by combining the conditions in each rule pair with the maximum operator. We use the model selection procedure using theorem 5 for the 1-norm SVM and the empirical margin maximization, and theorem 6 for MMV optimization. We set the risk term constants to 0.1 for the 1-norm SVM and empirical margin, and to

Data Set	PART	SLIPPER	MMV
australian	32	9	15
breastcancer	20	2	20
breastw	10	15	8
colic	9	2	15
diabetes	13	21	17
german	78	24	17
glassg2	7	30	17
heartc	25	19	22
hearth	10	8	10
heartstatlog	24	30	25
hepatitis	8	3	10
ionosphere	10	19	10
krvskp	23	35	6
labor	3	3	12
mushroom	13	10	5
sick	20	22	1
sonar	8	26	22
vote	7	4	2

Table 2. Results: number of induced rules.

1.0 for MMV to avoid overly pessimistic model selection. Also, for the empirical margin optimization, we adjusted the training sets in a preprocessing step to contain equally many positive and negative instances. Otherwise, it would have performed worse on data sets with skewed class distribution.

Table 1 gives the predictive accuracy of the induced rule sets as estimated by tenfold cross-validation for the proposed system in comparison to PART (Frank & Witten, 1998) and SLIPPER (Cohen & Singer, 1999). The table indicates that MMV performs better than the 1-norm SVM in 14 of the 18 cases and better than empirical margin optimization in 15 cases. To assess the statistical significance of this result we use a Wilcoxon signed ranks test. The test yields a chance probability of less than 0.02 in both comparisons. Thus, we can be very certain that MMV outperforms the 1-norm SVM and empirical margin optimization on similar data sets. MMV is also better than PART and SLIPPER in 10 of the 18 cases. The corresponding chance probabilities are 0.71 and 0.81. Even with this rather limited rule generation bias, MMV optimization appears to be competitive with existing rule learning algorithms.

Table 2 gives the number of rules that were generated by PART, SLIPPER and MMV. MMV generated the most compact rule set in 7 cases, PART in 6, and SLIPPER in 7 cases, so there is hardly a significant win for any of the three systems. However, in contrast to PART, MMV never generated an excessively large

rule set (more than 30 rules). Also, unlike PART and SLIPPER, each of MMV's rules contains at most two literals, so the individual rules are much more compact than those of the other systems.

One of the benefits of the flexible rule generation process, though, is the fact that the user can adjust the learner's bias by rearranging the order in which the rules are added or by including new rule conditions. The order matters, because the risk of overfitting increases with the number of generated rules. Hence, the structural risk term in (6) penalizes rules that are added only very late. To avoid the pruning of good rules, the user should configure the system to evaluate the (probably) more informative rules first. For instance, looking at the labor data set, one immediately notices that there are lots of missing values. The examples represent contracts and it seems to be sensible that the inclusion or omission of a certain attribute value is on purpose rather than a random phenomenon. Consequently, it might make sense to include new rules that are satisfied only if a specific attribute value is missing. Also, some of the features are obviously correlated (e.g. wage-increase-first-year and wage-increase-second-year). To avoid combinations of attributes with similar informative value, we calculate the mutual information of all pairs of base features and add the 200 disjunctions with the smallest mutual information as rules. These two modifications boost the estimated predictive accuracy from 80.7% to 91.2%, an increase of over ten points.

In another experiment (not shown in the tables), we tested MMV with a smaller language bias without combinations. As it turns out, the results are similar except for three datasets (breastw, glassg2 and mushroom), where the combinations clearly outperform the simple rules. On the labor data, combining the variables harms performance. However, as illustrated above, we can take advantage of a more sophisticated bias to overcome the limitations.

## 5. Conclusion

This paper presents a new approach to rule learning based on statistical considerations and providing a theoretically founded alternative to existing methods. Using the set of weighted rules representation, we avoid some of the combinatorial complexity of traditional DNF rule learning. Instead, we propose margin minus variance maximization as a relaxed version of empirical risk minimization. MMV optimization features two properties that make it particularly suited for rule learning:

- The optimization problem can be solved efficiently. In particular, the time complexity is linear in the number of instances.
- MMV optimization assigns non-zero weights only to a small number of the best rules. This ensures the induction of compact and comprehensible rule sets. In contrast, most other theoretical approaches to rule learning are based on ensemble methods (Cohen & Singer, 1999; Rückert & Kramer, 2004), which tend to favor large ensemble sizes.

We frame the problem of overfitting as a model selection task, leveraging a novel risk bound. An implementation of the proposed system appears to be competitive with two established rule learning algorithms. Additionally, it allows for a flexible way to adjust the bias in order to boost the predictive accuracy. Of course, there are plenty of opportunities for further research. For example, one could investigate the use of MMV maximization in relational domains or investigate other suitable optimization criteria.

## References

- Cohen, W., & Singer, Y. (1999). A simple, fast, and effective rule learner. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 335–342). AAAI Press.
- Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proc. 15th International Conf. on Machine Learning* (pp. 144–151). Morgan Kaufmann.
- Friedman, J. H., & Popescu, B. E. (2005). *Predictive learning via rule ensembles* (Technical Report). Stanford University.
- Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13, 3–54.
- Rückert, U., & De Raedt, L. (2006). An experimental evaluation of comprehensibility in rule learning. Submitted.
- Rückert, U., & Kramer, S. (2004). Towards tight bounds for rule learning. *Machine Learning, Proceedings of the 21st International Conf.*. ACM.
- Weiss, S., & Indurkha, N. (2000). Lightweight rule induction. *Proc. 17th International Conf. on Machine Learning* (pp. 1135–1142). Morgan Kaufmann.
- Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2004). 1-norm support vector machines. *NIPS*. MIT Press.