Learning Mixtures of Localized Rules by Maximizing the Area Under the ROC Curve

Tobias Sing[1] and **Niko Beerenwinkel**[2] and **Thomas Lengauer**[3]

Abstract. We introduce a model class for statistical learning which is based on mixtures of propositional rules. In our mixture model, the weight of a rule is not uniform over the entire instance space. Rather, it depends on the instance at hand. This is motivated by applications in molecular biology, where it is frequently observed that the effect of a particular mutational pattern depends on the genetic background in which it occurs. We assume in our model that the effect of a given pattern of mutations will be very similar only among sequences that are also highly similar to each other. On the other hand, a pattern might have very different effects in different genetic backgrounds.

Model inference consists of repeated iteration through a sequence of three steps: First, a new rule is mined from a resampled data set using the apriori algorithm. Next, the localization information for the rule is computed. Finally, the weights of all rules in the mixture model are re-optimized simultaneously. This weight optimization is done using the area under the ROC curve rather than the error rate as the objective function. Correspondingly, the weight of a sample in the resampling procedure is based on the rank of the sample relative to the other samples rather than directly on the score itself (such as in boosting). This strategy can be seen as an adaptation of boosting to the case of AUC optimization. Finally, we apply our method to the problem of predicting HIV-1 coreceptor usage from the amino acid sequence of the viral surface protein.

1 MOTIVATION

Classifiers based on sets of rules have received a lot of attention in the machine learning and data mining communities. This is certainly in large part due to the easy interpretability of the learned classifier, a feature that makes them especially attractive when the intended users are non-experts who want to understand *why* a particular prediction has been made. Our field of application is molecular biology, and consequently, we will formulate our approach for an instance space of proteins or nucleic acids. However, our method could also be used as a general attribute-value learner.

An important issue in rule-based classification is the question of how to use the learned rules to make a prediction. In [6], several strategies towards this so-called *resolution problem* are discussed. One of the strategies that are mentioned is *lowest false positive rate (LFPR)*. In LFPR, among all rules whose preconditions are fulfilled, a prediction is made only based on the rule with the lowest false positive rate. Of course, other quality criteria could be used instead of the false positive rate. In this more general interpretation, LFPR relies on a scoring scheme which can be used to rank individual rules. The prediction for a new instance is then performed using only the highest-scoring rule whose preconditions are fulfilled. While it seems very reasonable to assume that some rules are more important than others, there are a number of pitfalls that arise when the prediction is based only on a single rule and all other rules whose preconditions are fulfilled are ignored: Firstly, it is not clear why the rule with the highest score should be more relevant for the particular instance at hand than a rule with lower score. Secondly, more than one rule could be relevant for a given instance.

These objections motivate taking into account all rules whose preconditions are fulfilled in order to make a prediction. The most simple and most popular strategy for combining the predictions of several rules into a single prediction is *majority voting* (called *equal voting (VOTE)* in [6]). Here, for each class, the number of rules that predict this class are counted. The ensemble prediction is then the class with the largest number of advocates. While being more robust than the LFPR strategy, the idea that not all rules are equally important is abandoned here: The rather strong assumption implicitly underlying majority voting is that all rules are of the same quality. With majority voting, one ignores that in many biological applications some mutational patterns have stronger impact than others.

The advantages of the two strategies described above can be combined by taking a vote among all rules, with an individual weight attached to each rule. Indeed, as reported in [6], this approach, called *weighted voting (WVOTE)*, performed best among the five strategies that had been compared. This also seems to be the appropriate model for many tasks from a biological point of view, as underlined by the following two examples relating to the Human Immunodeficiency Virus (HIV): Firstly, it has been observed that if two particular mutations (each of which would be described by a rule here) associated with HIV drug resistance co-occur in an HIV particle, the virus with both mutations is more resistant than viruses with only one mutation. Secondly, in a recent investigation of HIV coreceptor usage, it has been observed that "although certain mutations may have disproportionate influence on coreceptor usage, such mutations are not necessary for coreceptor switching, provided V3 has accumulated enough other mutations with smaller effect." [11]. In summary, there is much evidence that all mutational patterns (each described by a rule) that occur in a sequence should be taken into account, each of them with an individual weight, modelling the strength of the effect of the particular pattern.

As to the resolution problem, our hypothesis space is a weighted voting model. However, the model goes beyond weighted voting by

[1] (a) Max Planck Institute for Informatics, Saarbrücken, Germany, email: tsing@mpi-sb.mpg.de
(b) Machine Learning and Natural Language Processing Group, University of Freiburg, Germany
[2] Max Planck Institute for Informatics, Saarbrücken, Germany, email: niko@mpi-sb.mpg.de
[3] Max Planck Institute for Informatics, Saarbrücken, Germany, email: lengauer@mpi-sb.mpg.de

challenging an assumption that to our knowledge has not been questioned before: Should the weight of a rule be modelled as uniform over the entire instance space?

In our point of view it is surprising that to date the answer to this question seems to be a universal "yes". Indeed, the assumption of constant rule weights is in sharp contrast to phenomena observed in molecular biology. For instance, it has been reported that the strength of the effect of certain resistance-associated mutations depends on the particular HIV subtype in which they occur [10], [4]. Likewise, in the context of HIV coreceptor usage, it has been observed that "no single set of mutations appears to lead to coreceptor switching in every genetic background" [11]. In summary, there is a body of evidence supporting the notion that the effect of a given pattern of mutations depends on the *genetic background* in which it occurs.

Thus, we argue that in deciding how to use a set of rules for a prediction, addressing the resolution problem (namely the question of how to combine the predictions of the individual rules) is only one side of the coin. In many practical classification problems, it will also be beneficial to deal with what we call the *relevance problem*, which is the problem of determining how relevant a given rule is for the instance at hand.

The rest of this paper is organized as follows. In the next section, we will describe the model class that results from combining the weighted voting strategy towards the resolution problem with a simple approach towards the relevance problem. In section 3, we describe a learning algorithm for these models. It is based on iteratively adding a rule to the partial model, determining its relevance surface, and finally re-optimizing all weights simultaneously with respect to the area under the ROC curve. We have applied our method on the task of predicting HIV-1 coreceptor usage from the viral sequence, and in section 4 we show results of a ROC evaluation of all methods that have been used so far for this task, including our own approach.

2 MIXTURES OF LOCALIZED RULES

2.1 Mixtures of rules

We consider the binary classification problem: An instance, given by the $\mathcal{X}$-valued random variable X, and its class, given by the $\{1,-1\}$-valued random variable Y, are drawn according to an unknown probability measure $P_{(X,Y)}$. The task is to make a correct prediction for an instance $x \in \mathcal{X}$, based on training data $\mathcal{D} = ((x_1, y_1), \ldots, (x_m, y_m))$. As mentioned before, we have in mind annotated amino acid sequences as an application task. In many cases, these sequences will be of different length, because insertions or deletions can be quite common. Sequences of different length are brought to the same length by inserting copies of a gap symbol "$-$" into them using special *alignment* algorithms, such as the one described in [15]. Thus, for our discussion, $\mathcal{X}$ is the set of all aligned amino acid sequences (including gaps) of length L. For example,

$$\begin{aligned} x_1 &= \texttt{CTRPGNNTRKSIRIGPGQAFYTN-HIIGDIRQAYC} \\ x_2 &= \texttt{CIRPNNNTRKGIYIGPGRAVYTTGNIIGDIRQAHC} \end{aligned}$$

are elements of $\mathcal{X}$ ($L = 35$). Individual elements of an instance are accessed via superscripted indexes. For example, $x_1^{(1)} = C, x_1^{(2)} = T$, and so on.

We restrict ourselves to the most elementary language for rules here (although extensions, for example towards first order logic, are certainly desirable). We only allow rules r of the form $r \equiv (j_1 : a_1 \wedge \ldots j_m : a_m \Rightarrow y)$, with $1 \leq j_1 < \ldots < j_m \leq L$, and $y \in \{1, -1\}$. Here, each a_i is either an amino acid or a gap symbol. A rule r induces a function in the obvious way: if for an instance x, all preconditions are fulfilled, i.e. $x^{(j_1)} = a_1, \ldots, x^{(j_m)} = a_m$, the class y given by its right-hand side is predicted, and otherwise 0. Since rules and the induced functions correspond to each other uniquely, by abuse of notation, we use the same letter r both for the rule and for the induced function.

With these preparations, and given rules $r_1, \ldots, r_K$ and corresponding weights $w_1, \ldots, w_K \in [0,1]$ such that $\sum_{i=1}^{K} w_i = 1$, the weighted voting strategy can be written as a K-mixture model of rules:

$$\begin{aligned} h_{(r_1,w_1),\ldots,(r_K,w_K)} : \mathcal{X} \to & \quad [-1,1], \\ x \mapsto & \sum_{i=1}^{K} w_i r_i(x). \end{aligned} \tag{1}$$

We omit the subscript if it is not relevant for the discussion. Here, K is a fixed integer, treated as a model parameter. Model selection can be performed by standard techniques such as cross-validation, or the Bayesian or Akaike information criterion [9].

The function h is a scoring classifier. It induces a family of binary classifiers $\{h_\gamma\}_{\gamma \in [-1,1]}$, based on the choice of different cutoffs γ:

$$h_\gamma(x) = \begin{cases} 1, & h(x) \geq \gamma \\ -1, & \text{otherwise.} \end{cases}$$

2.2 Localized rules

As motivated in the introduction, our model should allow for rule weights which vary over the instance space. We focus on a rule r_i in the mixture model (1). The corresponding weight $w_i \in [0,1]$ is now interpreted as the maximal weight r_i can attain over the instance space. This weight is modulated by a function $\rho_i : \mathcal{X} \to [0,1]$, the so-called *relevance function* of the rule. As the name suggests, it estimates the relevance of the rule r_i for the given sample. With this modification, the contribution of a rule to the mixture model is $w_i \rho_i(x) r_i(x)$. The remainder of this section will be devoted to defining a model for the relevance function that can be estimated from data.

Ideally, the relevance function of a rule r would be defined as

$$\rho_{\text{ideal}}(x) := P(r(X) = Y | X = x),$$

the probability that the rule's prediction will be correct, given that sample x has been drawn with unknown class. However, it is not feasible to estimate this probability from training data. We now describe a simple and practical model for the relevance function of a rule.

A rule r partitions the training data $\mathcal{D}$ into three sets: The first set consists of those samples for which the rule's preconditions are not fulfilled and is of no further interest for the relevance function. The set $\mathcal{D}_{\text{corr}}(r)$ contains those samples for which the preconditions of r are fulfilled and the prediction is correct. Finally, the set $\mathcal{D}_{\text{incorr}}(r)$ consists of samples that fulfill the preconditions, but are incorrectly predicted. In light of our fundamental assumption that the relevance of a rule depends on the genetic background in which it occurs, the samples in $\mathcal{D}_{\text{corr}}(r)$ are examples of genetic backgrounds where the rule seems to be relevant, whereas $\mathcal{D}_{\text{incorr}}(r)$ contains examples of regions of the instance space where the rule is irrelevant.

For good rules, $\mathcal{D}_{\text{corr}}(r)$ typically should be much larger than $\mathcal{D}_{\text{incorr}}(r)$. Because of this, we decided to split up the model for the relevance function into two parts: the first part is a distribution-free model that only relies on the data in $\mathcal{D}_{\text{corr}}(r)$. The second part of the model adjusts this first part by also taking into account $\mathcal{D}_{\text{incorr}}(r)$.

2.2.1 *First part: the similarity function*

We assume that the effect of a particular pattern of mutations (as described by a rule) will be similar if it occurs within genetic backgrounds that are similar to each other. For a new instance to be classified this means that if it has high sequence similarity to the correctly predicted samples, the rule should be considered to be very relevant, whereas if the sequence similarity is low, so should be the relevance.

There are several ways to measure the similarity of a sequence x to a family of sequences, but they all amount to computing the probability of seeing x under a probabilistic sequence model induced by the family. Here, the family is the set $\mathcal{D}_{\text{corr}}(r)$, which consists of realizations of the random pair (X, Y) for which $r(X) = Y$. In summary, the similarity $\lambda(x)$ of a sequence x to the family of correctly predicted sequences is the probability of drawing the sequence when randomly drawing a correctly predicted instance. In fact, we use the logarithm of this probability:

$$\lambda(x) := \log P(X = x | r(X) = Y)$$

Our approach to estimating this probability based on the set $\mathcal{D}_{\text{corr}}(r)$ is described in section 3.

2.2.2 *Second part: the relevance adjustment*

If the model is to take into account the set $\mathcal{D}_{\text{incorr}}(r)$ as well, i.e. examples of genetic backgrounds where the rule does not work, this has to be done using strong parametric assumptions, due to the limited size of this set (as compared to $\mathcal{D}_{\text{corr}}(r)$) in good rules. If the similarity function is estimated well, and if our assumption that the effect of a given mutation is similar in similar genetic backgrounds is valid, one might expect that the similarity function λ will assign a higher score to samples where the rule is relevant than for those where it is not.

In this second part of the relevance model, the task is to determine how optimistically or how pessimistically the similarity function should be translated into a relevance estimate, on the basis of both $\mathcal{D}_{\text{corr}}(r)$ and $\mathcal{D}_{\text{incorr}}(r)$. Such a translation has to be a mapping from $(-\infty, 0]$ to $[0, 1]$. A class of functions that allows for a simple adjustment of this mapping are the sigmoid functions known from artifical neural networks:

$$\sigma_{\alpha,\tau}(z) = \frac{1}{1 + e^{-\alpha(z-\tau)}},$$

where $z \in (-\infty, 0]$. This class is parametrized by two variables, α and τ: The parameter τ shifts the graph of σ horizontally, while α adjusts its slope.

Figure 1 shows how a sigmoid function is used to squeeze the similarity function into the interval $[0, 1]$. The circles denote the training samples. On the x-axis, the value of the similarity function, $\lambda(x)$, is plotted. On the y-axis, the value is 1 for samples from $\mathcal{D}_{\text{corr}}(r)$, and 0 for samples from $\mathcal{D}_{\text{incorr}}(r)$. The idea in selecting appropriate values for α and τ is that if the correctly predicted and incorrectly predicted samples are substantially separated by λ, the transformation into a relevance function should be more optimistic than if there is not much space in between. The optimality criterion chosen by us is to minimize the sum of squared distances in the graph, regularized by a term that penalizes large values of α, i.e. very steep sigmoid

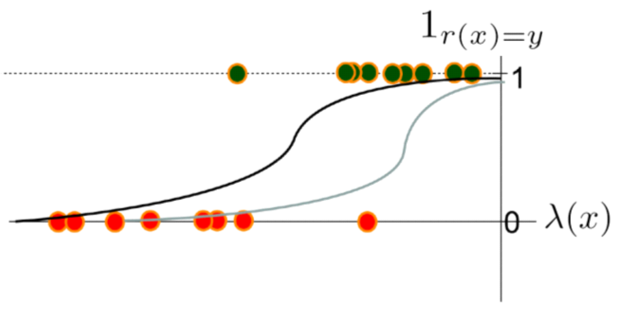


Figure 1. Sigmoid adjustment of the similarity function.

functions (the rationale for regularizing here is to prevent overfitting). Let $A = \mathcal{D}_{\text{corr}}(r) \cup \mathcal{D}_{\text{incorr}}(r)$, then the optimal parameters are defined as

$$\begin{aligned}(\alpha^*, \tau^*) &:= \arg\min_{\alpha,\tau} \\ &\kappa \frac{1}{|A|} \sum_{i \in A} (\sigma_{\alpha,\tau}(\lambda(x_i)) - 1_{r(x_i)=y_i})^2 + (1-\kappa)\alpha^2. \quad (2)\end{aligned}$$

2.3 Mixtures of localized rules

In summary, the space of mixtures of localized rules consists of functions of the form

$$\begin{aligned} h &: \mathcal{X} \to [-1, 1] \\ x &\mapsto \sum_{i=1}^{K} w_i \frac{1}{1 + e^{-\alpha_i(\lambda_i(x_i) - \tau_i)}} r_i(x). \end{aligned} \quad (3)$$

with rules $r_1, \ldots, r_K$, their corresponding similarity functions $\lambda_1, \ldots, \lambda_K$, adjustment parameters $(\alpha_1, \tau_1), \ldots, (\alpha_K, \tau_K)$ for the sigmoid functions, and rule weights $w_1, \ldots, w_K$. In the next section we describe our approach to learning these parameters.

3 LEARNING ALGORITHM

Figure 2 shows a high-level description of the learning algorithm. It consists of K iterations of the following three steps: first (section 3.1), a new rule is mined from a resampled dataset. Next (section 3.2), the relevance surface for the rule is computed. Finally (section 3.3), the rule is added to the mixture model by re-optimizing all rule weights simultaneously, with the area under the ROC curve as the objective function.

3.1 Step 1: Mining a new rule

Step 1 consists of five substeps:

1. Resampling the data set.
2. Mining potentially interesting rules using the apriori algorithm.
3. Scoring the rules.
4. Generalizing the rules.
5. Selecting the highest-scoring rule that is compatible with the already selected rules.

ALGORITHM **learn_LRM** (HIGH-LEVEL DESCRIPTION)
INPUT:

- Training data $\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X}, \mathcal{Y})^n$
- Number of rules in the model, $K \in \mathbb{N}$.

OUTPUT:

- A mixture model of localized rules

$$h(x) = \sum_{i=1}^{K} w_i \frac{1}{1 + e^{-\alpha_i(\lambda_i(x_i) - \tau_i)}} r_i(x).$$

PROCEDURE:

For $k = 1, \ldots, K$:

1. **Mine a new rule (section 3.1):** Mine a new rule r_k from a resampled dataset (drawing the learner's attention to the most problematic instances) using the apriori algorithm.
2. **Localization (section 3.2):** Determine the similarity function λ_k and the adjustment parameters α_k and τ_k for the rule r_k.
3. **Weight optimization (section 3.3):** Find an optimal combination of weights $w_1, \ldots, w_k$ for all rules $r_1, \ldots, r_k$ selected so far, i. e. solve the following optimization problem:

$$\underset{\mathbf{w}_1,\ldots,\mathbf{w}_k \in [0,1], \sum \mathbf{w}_i = 1}{\text{maximize}} \quad \text{AUC}(h_{(r_1,\mathbf{w}_1,\lambda_1,\alpha_1,\tau_1),\ldots,(r_k,\mathbf{w}_k,\lambda_k,\alpha_k,\tau_k)}).$$

Figure 2. Learning algorithm (high-level description).

3.1.1 *Resampling the data set.*

Resampling from a data set is done by first defining a probability measure on the data and then drawing with replacement a data set of the same size as the original set, according to the given distribution. The distribution used in our method to determine the weight of a sample in resampling is the fraction of training samples from the other class than the sample that score higher (if it is a positive sample) respectively lower (if the sample is negative). If we denote the positive samples by $\mathcal{D}_+$ and the negative samples by $\mathcal{D}_-$, this quantity is given by

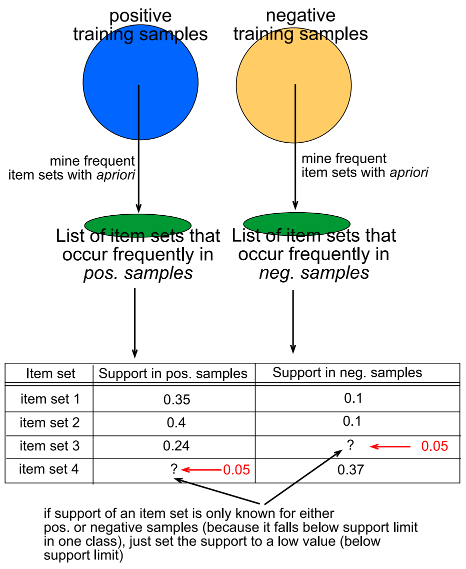
$$\tilde{p}_i := \begin{cases} \frac{\sum_{j \in \mathcal{D}_-} 1_{h(x_j) \geq h(x_i)}}{|\mathcal{D}_-|}, & y = 1 \\ \frac{\sum_{j \in \mathcal{D}_+} 1_{h(x_j) \leq h(x_i)}}{|\mathcal{D}_+|}, & y = -1. \end{cases}$$

In order to get a probability measure, the $\tilde{p}_i$ are normalized in the usual way:

$$p_i := \frac{\tilde{p}_i}{\sum_{j \in \mathcal{D}} \tilde{p}_j}$$

3.1.2 *Mining rules with the apriori algorithm.*

Figure 3 shows how the apriori algorithm [1] is used to find potentially interesting rules. First, the resampled data set, which we denote by $\mathcal{D}'$, is divided into $\mathcal{D}'_{\text{pos}}$, the positive samples, and $\mathcal{D}'_{\text{neg}}$, the negative samples. On each of these sets we apply apriori to mine frequent



positive training samples
negative training samples
mine frequent item sets with *apriori*
mine frequent item sets with *apriori*
List of item sets that occur frequently in *pos. samples*
List of item sets that occur frequently in *neg. samples*
0.05
0.05
if support of an item set is only known for either pos. or negative samples (because it falls below support limit in one class), just set the support to a low value (below support limit)

Item set	Support in pos. samples	Support in neg. samples
item set 1	0.35	0.1
item set 2	0.4	0.1
item set 3	0.24	?
item set 4	?	0.37

Figure 3. Mining rules with the apriori algorithm.

item sets[4], with as low a support bound as computationally feasible. The two families of item sets are then merged as shown in the figure, resulting in a list of item sets for which the support both on the positive as well as on the negative samples is known.

The item sets are then turned into rules in the obvious way: if the support on the positive samples is higher than on the negative samples, class 1 is added as a consequent, otherwise class -1 is added.

3.1.3 *Scoring the rules.*

To identify interesting rules, a scoring scheme for rules is needed. Intuitively, a good rule is one for which $\mathcal{D}'_{\text{corr}}(r)$ is large, while $\mathcal{D}'_{\text{incorr}}(r)$ is small. Consequently, the most obvious idea would be to take the fraction

$$\frac{|\mathcal{D}'_{\text{corr}}(r)|}{|\mathcal{D}'_{\text{incorr}}(r)|}$$

between the correctly and the incorrectly predicted samples as the score for a rule r.

However, the scoring scheme should also take into account the generality of a rule. Thus, for a rule that predicts class 1 (for rules that predict class -1 replace $\mathcal{D}_+$ with $\mathcal{D}_-$ in the formula below) we define:

$$\sigma(r) := \begin{cases} \left(\frac{1}{\tau} \operatorname{supp}_{\mathcal{D}'_+}(r)\right)^{\alpha} \frac{\operatorname{supp}_{\mathcal{D}'_{\text{corr}}}(r)}{\operatorname{supp}_{\mathcal{D}'_{\text{incorr}}}(r)}, & \text{if } \operatorname{supp}_{\mathcal{D}'_+}(r) \geq \tau \\ \frac{\operatorname{supp}_{\mathcal{D}'_{\text{corr}}}(r)}{\operatorname{supp}_{\mathcal{D}'_{\text{incorr}}}(r)} & \text{otherwise,} \end{cases}$$

where the support $\operatorname{supp}_{\mathcal{D}}(r)$ of a rule r on a data set $\mathcal{D}$ is defined as the fraction of samples in $\mathcal{D}$ for which the preconditions are fulfilled. Here, τ and α are treated as fixed model parameters that determine the trade-off between the generality of a rule and the $\mathcal{D}_{\text{correct}}/\mathcal{D}_{\text{incorrect}}$ ratio.

[4] We have used the apriori implementation provided by Christian Borgelt (`http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/apriori/apriori.html`).

3.1.4 *Generalizing the rules.*

We say that a rule $(j_1 : a_1, \ldots, j_m : a_m \Rightarrow y)$ is *more general* than a rule $(i_1 : b_1, \ldots, i_n : b_n \Rightarrow y)$, if $\{j_1 : a_1, \ldots, j_m : a_m\} \subseteq \{i_1 : b_1, \ldots, i_n : b_n\}$. A rule r is *more specific* than a rule r' iff r' is more general than r. Denote by $\mathcal{G}(r)$ the set of all rules which are more general than r, and by $\mathcal{S}(r)$ those that are more specific than r.

Our scoring scheme does not take into account the number of preconditions of a rule. Thus, it can happen that instead of the correct version of a rule, a more specific rule will attain a higher score. Therefore, to prevent overfitting, the rules that have been found using the apriori algorithm should be generalized as much as possible without too much loss in score.

Below, we define which rule should be considered as the appropriate generalization of a rule r by the learner. Let σ_{max} be the maximal score among the rules in $\mathcal{G}(r)$. Certainly, the score of the generalized rule should not be too far from the maximal score. Of all rules in $\mathcal{G}(r)$ that fulfill this criterion, we are interested in a shortest one (i.e. one with a minimal number of preconditions). If there is more than one such rule, the one with the highest score is selected. In summary, using $|r|$ to denote the number of preconditions of r, we can define the *preferred generalization* of r as the uniquely determined rule $r^* \in \mathcal{G}(r)$ such that

- $\sigma(r^*) \geq \sigma_{\text{max}} - \epsilon$,
- $\forall r' \in \mathcal{G}(r) : \quad |r'| < |r^*| \Rightarrow \sigma(r') < \sigma_{\text{max}} - \epsilon$, and
- $\forall r' \in \mathcal{G}(r) : |r'| = |r^*| \Rightarrow \sigma(r') < \sigma(r^*)$.

3.1.5 *Selecting the highest-scoring compatible rule.*

In the final part of the rule-mining step, the task is to select one of the generalized rules for integration into the mixture model. Intuitively, it is clear that it would make no sense to add a rule which is more general or more specific than an already selected rule. Consequently, we require that a rule can only be added to the mixture model if it is *compatible* to the already selected rules: Let $r_1, \ldots, r_k$ be the rules already incorporated into the mixture model. The rule r is said to be *compatible* to $r_1, \ldots, r_k$ iff none of the r_i is more general or more specific than r, i.e. if $\forall i \in \{1, \ldots, k\} : r_i \notin \mathcal{G}(r)$, and $r_i \notin \mathcal{S}(r)$. In particular, of all the generalized rules, the highest-scoring one which is compatible to the data set is chosen.

3.2 Step 2: Estimating the relevance function of a rule

When a new rule r_i has been selected for integration into the model, the next task is to determine its relevance function. As described in section 2.2, this task consists of two parts. First, the similarity function λ_i is estimated. Next, the shift τ_i and slope α_i of the sigmoid adjustment have to be computed.

3.2.1 *Estimating λ_i*

First, we show how to estimate the similarity function λ_i. As motivated in section 2.2.1, λ_i is intended to model $P(X = x|r(X) = Y)$, the probability of encountering instance x when randomly drawing a pair of an instance and a class for which the rule's prediction is correct.

Without some (conditional) independence assumptions, this probability cannot be estimated in practice. The simplest approach is to assume independence among the columns in the alignment (alternatives to this rather strong assumption are discussed in the concluding section of this paper). Thus, if we denote by L the number of columns in the alignment,

$$P(X = x|r(X) = Y) = \prod_{j=1}^{L} P(X^{(j)} = x^{(j)}|r(X) = Y).$$

The probabilities $p_{ij} := P(X^{(j)} = a_i)|r(X) = Y)$ can be estimated using a robust estimator, similarly as in the case of the naive Bayes classifier, by

$$\hat{p}_{ij} = \frac{|\{i \in \mathcal{D}'_{\text{corr}}(r)|x_i^{(j)} = x^{(j)}\}| + mp}{|\mathcal{D}'_{\text{corr}}(r)| + m}.$$

Here, p is the prior estimate of the probability, and m is a constant which determines how heavily the prior estimate should be weighted relative to the observed data. In computational biology, the matrix (p_{ij}) is called a *sequence profile* for the family of sequences which are predicted correctly by the rule r.

3.2.2 *Estimating α_i and τ_i*

The optimization problem stated in formula 2 can be solved using gradient descent, since the objective function is differentiable as a function of α and τ. Several restarts can be performed to avoid getting trapped in local minima.

3.3 Step 3: Optimizing the weights for all rules

In the final step, the weights for all rules within the mixture model are re-optimized simultaneously with the AUC as the objective function. Although there is no definitive answer yet, there are indications that under certain circumstances such as considerable class skew, or in difficult classification problems, optimizing with respect to the AUC might yield a more robust classifier than optimizing with respect to the error rate [3].

Formally, the goal is to find weights $w_1^*, \ldots, w_k^*$, such that the AUC of the mixture model, when parametrized with these weights is maximal among all $w_1, \ldots, w_k \in [0, 1]$, for which $\sum w_i = 1$. One approach towards optimizing this discontinuous function would be to use combinatorial methods such as combinatorial simulated annealing or taboo search. However, we use a different strategy here, which has also been investigated in [16]. This strategy is built on two ideas: Firstly, it exploits the equivalence of the area under the ROC curve with the Wilcoxon-Mann-Whitney (WMW) statistic [5]. Secondly, it approximates the step functions of which the WMW statistic is composed, by sigmoid functions. The resulting approximation of the AUC is differentiable with respect to the weights, and therefore, gradient descent can be used to find the optimal weights.

Before going into the details, a remark on notation. Denote by $\mathcal{D}_+$ respectively $\mathcal{D}_-$ the positive respectively negative training samples. For a weight vector $w = (w_1, \ldots, w_k)$ we denote the mixture model with these weights by h_w. Our task is to solve the optimization problem:

$$\underset{w \in [0,1]^k, \sum_{r=1}^{k} w_r = 1}{\text{maximize}} \quad \text{AUC}(h_w).$$

It is easy to see [14] that the area under the curve is equal to the WMW test statistic:

$$\text{AUC}(h_w) = \frac{1}{|\mathcal{D}_+||\mathcal{D}_-|} \sum_{i \in \mathcal{D}_+} \sum_{j \in \mathcal{D}_-} 1_{h_w(x_i) > h_w(x_j)}.$$

Thus,

$$\max_w \mathrm{AUC}(h_w) = \max_w \sum_{i\in\mathcal{D}_+} \sum_{j\in\mathcal{D}_-} 1_{h_w(x_i)>h_w(x_j)}.$$

As to the the second idea mentioned above, namely the differentiable approximation of the step function, the observation is simply that the sequence of functions

$$g_n : z \mapsto \frac{1}{1+e^{-nz}}$$

converges pointwise to the step function $1_{z>0}$, with $n \to \infty$. Therefore, with β sufficiently large,

$$\max_w \mathrm{AUC}(h_w) \approx \sum_{i\in\mathcal{D}_+} \sum_{j\in\mathcal{D}_-} \frac{1}{1+e^{-\beta(h_w(x_i)-h_w(x_j))}}. \quad (4)$$

If we further define

$$\tilde{x}_i := \frac{1}{1+e^{-\alpha_i(\lambda(x_i)-\tau_i)}} r_i(x), \quad i = 1,\dots,k$$

and use $\tilde{x}$ to denote the vector $(\tilde{x}_1,\dots,\tilde{x}_k)$, we can write

$$h_w(x) = \langle w, \tilde{x}\rangle.$$

The right-hand side of (4) can then be written more conveniently as

$$g(w) = \sum\sum \frac{1}{1+e^{-\beta\langle w, \hat{x}_i - \hat{x}_j\rangle}}$$

The function g is differentiable, and the partial derivatives are given by

$$\frac{\partial g}{\partial w_r}(w) = \sum\sum \beta(\hat{x}_i - \hat{x}_j)\frac{e^{-\beta\langle w, \hat{x}_i - \hat{x}_j\rangle}}{(1+e^{-\beta\langle w, \hat{x}_i - \hat{x}_j\rangle})^2}.$$

With these partial derivatives available, the AUC can be maximized using gradient ascent.

4 PREDICTION OF HIV-1 CORECEPTOR USAGE

We applied our method to the task of predicting HIV-1 coreceptor usage from viral genetic information. Like all viruses, HIV is dependent on a host cell to make copies of itself. In order to enter a cell, it successively attaches to two receptors on the cell surface. The first one, which is called the *main receptor*, is always the same for each virus particle. However, there are two receptors, called CCR5 and CXCR4, that can serve as the second receptor (*coreceptor*). HIV particles fall into three classes according to which of those two receptors they can use: some can only use CCR5 (*R5 viruses*), some can only use CXCR4 (*X4 viruses*), and some can use either of them (*R5X4 viruses*). One is interested in finding out about viral coreceptor usage, because a new class of anti-HIV drugs is being developed that tries to prevent HIV from binding to one of the two coreceptors. There is particularly strong interest in predicting coreceptor usage based on sequence data alone because these approaches would only require sequencing of the virus, which is a cheap and fast routine task as compared to the more expensive and time-consuming experimental assays for determining coreceptor usage.

In the application we report here, the task was to recognize virus that can use CXCR4. Therefore, we have X4 and R5X4 viruses as class 1, and R5 viruses as class -1. Prediction was based on the third variable region (V3 region) of the HIV envelope protein gp120, which is known to be the strongest determinant of coreceptor usage (reviewed in [2]).

4.1 Data set

Our data set consists of 1110 sequences of the V3 region (769 R5, 210 X4, 131 R5X4) from 332 different patients. These were obtained from the HIV Sequence Database in Los Alamos[5], as well as from the literature. Sequences were aligned to a V3 reference alignment provided by the HIV Sequence Database.

4.2 Compared learning methods

We have compared the methods that have been proposed so far for the prediction of HIV coreceptor usage, including the method described in this paper. In the case of support vector machines, different kernels and instance representations have been tested: in the indicator representation, each position in the alignment was represented by 21 entries (20 amino acids and one gap symbol) in the instance vector, one of which was set to 1, while all others were set to zero. The physico-chemical representation was similar, except that the entries were not representing the amino acids themselves, but their physico-chemical properties.

Method	Reference
SVM with linear kernel, indicator representation	[12]
Decision tree	[12]
Artificial neural network	[13]
Charge rule	[7]
PSSM	[11]
SVM with linear kernel, physico-chemical representation	[14]
SVM with RBF kernel, indicator representation	[14]
Mixtures of localized rules	[14]

4.3 Experiments

Parameter optimization was performed using a grid search with the area under the ROC curve as the objective function. Neural networks were parameter optimized with respect to the weight decay parameter (the backpropagation algorithm was used for training). In the case of SVMs, the parameters C (linear kernel) respectively C and γ (RBF kernel) were optimized.

Classifier performance was measured using stratified 10 times 10-fold cross-validation. ROC curves were obtained for all the compared classifiers without averaging, as well as using vertical and threshold averaging, to get error bars as described in [5]. Results did not depend on the particular averaging method (cf. [14] for details).

4.4 Results

Figure 4 shows the ROC curves for all the compared classifiers. Error bars (1 standard error) were obtained from threshold averaging. As can be seen in the plot, support vector machines with different kernels and instance representations are superior over all other methods on the entire range of false positive rates. Position-specific scoring matrices have good performance on higher false positive rates. Our method has good performance at higher false positive rates as well, but has some weaknesses at the practically important regions with small error rate.

[5] `http://www.hiv.lanl.gov/content/hiv-db/mainpage.html`

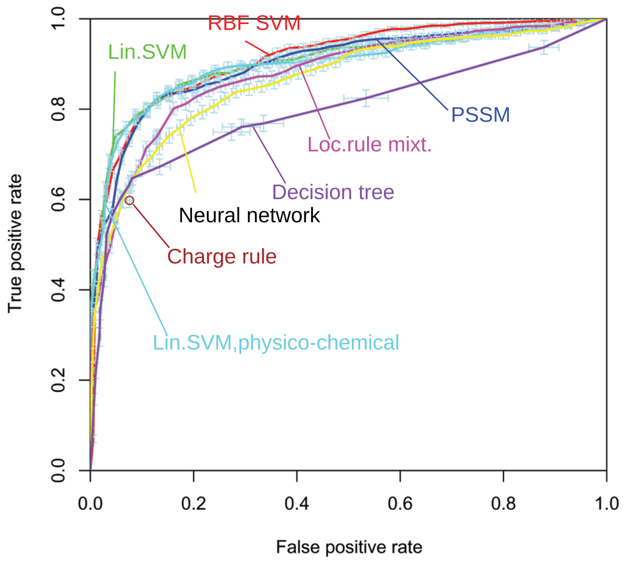


Figure 4. ROC plot showing the performance of the compared methods over the range of all possible cutoffs.

First, we compare the methods on a global range. The use of the area under the ROC curve is a popular measure for this. This geometric measure can also be given a probabilistic interpretation: $\mathrm{AUC}(h) = P(h(X) > h(X')|Y = 1, Y' = -1)$. This means, the AUC is equal to the probability that the classifier will assign a higher score to a randomly drawn positive sample than to a randomly drawn negative one.

Rank	Method	mean AUC
1.	RBF SVM indicator	0.9121
2.	Lin. SVM indicator	0.9046
3.	PSSM	0.9018
4.	Lin. SVM physico-chemical	0.8986
5.	LRM	0.8807
6.	Neural network	0.8664
7.	Decision tree	0.7474

The variance in the data is surprisingly high, as can be seen in figure 5, where boxplots of the measured AUCs during the different cross-validation runs are shown. Still, when testing the null hypothesis of equal AUC against the alternative hypothesis that the linear SVM has higher AUC, the differences in AUC between the linear SVM with indicator representation and localized mixtures of rules ($p = 0.0009$), neural networks ($p = 9.1 \cdot 10^{-5}$), or PSSMs ($2.2 \cdot 10^{-16}$) are highly significant (Wilcoxon-Mann-Whitney test). In contrast, the differences in AUC to the other methods are not significant, in particular to the PSSM (p=0.19).

A global measure such as the AUC should not be the only criterion when comparing classifiers. It is also important to look in more detail at the performance of classifiers at regions of practical relevance, i.e. at small false positive rates. Figure 6 shows the details: The three versions of SVMs perform best, with a slight performance gap to the PSSM (p = 0.0009 at 1%, $p = 3.2{\cdot}10^{-5}$ at 5%, and $p = 0.03$ at 7.5% error rate). The method described by us here performs comparative to decision trees and neural networks.

5 DISCUSSION

In this paper we have proposed a rule-based model class, whose elements are called mixtures of localized rules, along with an algorithm to learn these models from data. One of the benefits of this model class is that it combines the easy interpretability of rule sets with the power of non-symbolic hypothesis spaces via the notion of relevance surfaces defined for each rule over the instance space.

Indeed, this idea of localized rules (i.e. the assumption that the relevance of a rule depends on the genetic background) is one of the two distinguishing features of our approach, the other being the combination of rank-based resampling and AUC optimization in the learning algorithm. We shall now discuss some further aspects of these two features.

5.1 Localization

As already mentioned in section 2.2, one can think of many alternatives to our strategy towards the relevance problem. For example, it could be beneficial to replace the strong assumption of independence among the columns in the alignment with a slightly more realistic sequence model. For example, one of these models is based on the assumption that the probability that a given amino acid a will occur at position j in the alignment is not independent from the other positions, but conditionally independent, given the amino acid at the position $(j-1)$:

$$P(X^{(j)} = x^{(j)}|X^{(j-1)} = x^{(j-1)}, \ldots, X^{(1)} = x^{(1)}) = \\ P(X^{(j)} = x^{(j)}|X^{(j-1)} = x^{(j-1)}).$$

With this assumption, the appropriate probabilistic sequence model would be a hidden Markov model rather than a sequence profile.

As another alternative, one could abandon our two-step approach towards localization, and instead take into account the correctly and the incorrectly predicted training samples simultaneously, for example using position-specific scoring matrices. However, this will most likely give rise to robustness problems, given the small size of $\mathcal{D}_{\text{incorr}}(r)$.

A third, and entirely different, approach towards the relevance problem would consist in performing a pre-clustering of the training

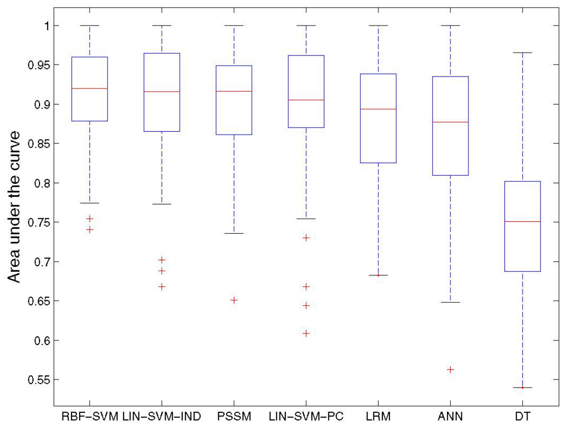


Figure 5. High variance of the AUC.

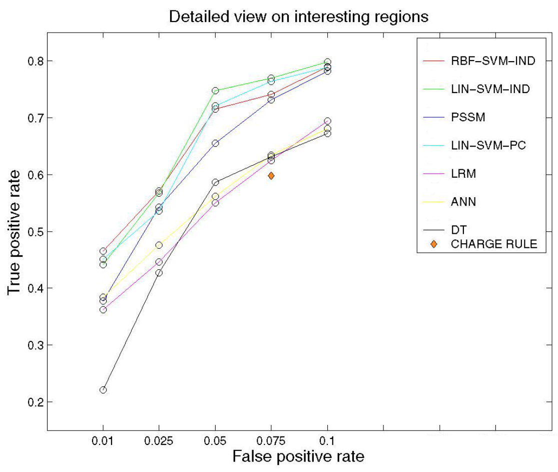


Figure 6. ROC plot: focusing on regions of practical interest.

samples. In this framework, the relevance region of a rule could be a cluster, or if a hierarchical clustering method is used, some function that varies according to the height in the dendrogram.

5.2 AUC boosting

In the beginning of this paper we have argued that the combination of rank-based resampling with AUC optimization which is used in our learning algorithm can be seen as an adaptation of boosting to the case when the AUC is used as an objective function.

To see why, let us first consider why the traditional notions of the margin of a sample are not appropriate for this situation. The reason is that they all take into account the predicted score $h(x)$ directly. Consider for example the exponential margin used by the AdaBoost algorithm [8]. In AdaBoost, the weight of a training sample (x_i, y_i) in the resampling procedure is $e^{-y_i h(x_i)}$. Yet, in classifiers optimizing the area under the ROC curve, it could well be the case that all training samples are scored below 0. The classifier could still perform perfectly, since the cutoff can be chosen freely. However, since the exponential margin implicitly assumes a cutoff at 0, in our case (where $h(x) < 0$ for all $x \in \mathcal{D}$) this would result in a situation where the positive training samples are considered more problematic than the negative ones, just because they are scored below 0.

Of course it is possible to adjust the exponential margin to other cutoffs than 0. However, it would be more desirable to have a resampling scheme that is entirely independent of the choice of a cutoff. One example of such a scheme is the rank-based resampling performed by our method. The choice of ranks as resampling weights in the context of AUC optimization also seems to be appropriate in the light of the fact that the AUC is proportional to the sum of ranks of the samples, relative to the samples of the other class [14].

Finally, future work will have to show whether rank-based resampling in combination with AUC optimization could also be useful when other models, such as decision trees or neural networks, are used as basis classifiers instead of rules.

5.3 Conclusion

While our method performs significantly worse than support vector machines or PSSMs, it shows comparable performance to decision trees and neural networks at low false positive rates. At higher false positive rates, it outperforms both methods. Since this is work in progress, we still expect significant improvements of our method. Certainly, evaluation on a wider range of data sets is needed to get a more complete picture of its benefits and drawbacks. Approaches to localize the influence of rules or the combination of rank-based resampling with AUC maximization described in this paper might also be beneficial in other contexts.

ACKNOWLEDGEMENTS

T. S. wants to thank Luc de Raedt from the Machine Learning and Natural Language Processing Lab at the University of Freiburg for co-supervising this work. N. B. was supported by the German National Science Foundation (DFG) under grant no. HO 1582/1-3.

REFERENCES

[1] R. Agrawal and R. Srikant, ‘Fast algorithms for mining association rules’, in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, eds., Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, pp. 487–499. Morgan Kaufmann, (1994).

[2] J.M. Coffin, ‘Molecular biology of HIV’, in *The Evolution of HIV*, ed., K. Krandall, Johns Hopkins UP, Baltimore, Maryland, (1999).

[3] C. Cortes and M. Mohri, ‘AUC optimization vs. error rate minimization’, in *Advances in Neural Information Processing Systems 16*, eds., Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, MIT Press, Cambridge, MA, (2004).

[4] R. T. D’Aquila et al., ‘Drug resistance mutations in HIV-1’, *Top HIV Med*, **11**(3), 92–96, (May 2003).

[5] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. HP Labs Tech Report HPL-2003-4.

[6] T. Fawcett, ‘Using rule sets to maximize ROC performance’, in *Proceedings of the 2001 IEEE International Conference on Data Mining*, San Jose, CA, (2001).

[7] R. A. Fouchier, M. Brouwer, S. M. Broersen, and H. Schuitemaker, ‘Simple determination of Human Immunodeficiency Virus type 1 syncytium-inducing V3 genotype by PCR’, *J Clin Microbiol*, **33**(4), 906–911, (Apr 1995).

[8] Y. Freund and R.E. Schapire, ‘Experiments with a new boosting algorithm’, in *International Conference on Machine Learning*, pp. 148–156, (1996).

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, New York, 2001.

[10] R. Jaideep et al., ‘HIV-1 protease and reverse transcriptase mutation patterns responsible for discordances between genotypic drug resistance interpretation algorithms’, *J Acquir Immune Defic Syndr*, **33**(1), 8–14, (May 2003).

[11] M. A. Jensen et al., ‘Improved coreceptor usage prediction and genotypic monitoring of R5-to-X4 transition by motif analysis of Human Immunodeficiency Virus type 1 env V3 loop sequences’, *J Virol*, **77**(24), 13376–13388, (Dec 2003).

[12] S. Pillai, B. Good, D. Richman, and J. Corbeil, ‘A new perspective on V3 phenotype prediction’, *AIDS Res Hum Retroviruses*, **19**(2), 145–149, (Feb 2003).

[13] W. Resch, N. Hoffman, and R. Swanstrom, ‘Improved success of phenotype prediction of the Human Immunodeficiency Virus type 1 from envelope variable loop 3 sequence using neural networks’, *Virology*, **288**(1), 51–62, (Sep 2001).

[14] T. Sing, *Learning localized rule mixtures by maximizing the area under the ROC curve, with an application to the prediction of HIV-1 coreceptor usage*, Master’s thesis, Max Planck Institute for Informatics, 2004.

[15] J. D. Thompson, D. G. Higgins, and T. J. Gibson, ‘CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice’, *Nucleic Acids Res*, **22**(22), 4673–4680, (1994).

[16] L. Yan, R. Dodier, M. C. Mozer, and R. Wolniewicz, ‘Optimizing classifier performance via the Wilcoxon-Mann-Whitney statistic’, in *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, eds., Tom Fawcett and Nina Mishra, pp. 848–855. AAAI Press, (2003).