JOURNAL OF CHEMICAL INFORMATION AND MODELING

ARTICLE

pubs.acs.org/jcim

Large-Scale Learning of Structure−Activity Relationships Using a Linear Support Vector Machine and Problem-Specific Metrics

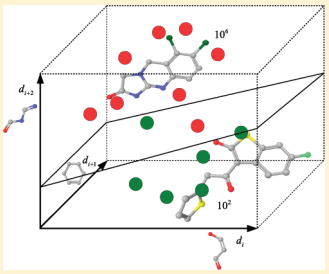
Georg Hinselmann,*[,†] Lars Rosenbaum,[†] Andreas Jahn,[†] Nikolas Fechner,[†] Claude Ostermann,[‡] and Andreas Zell[†]

[†]Center for Bioinformatics (ZBIT), University of Tübingen, Tübingen, Germany

[‡]Nycomed GmbH, Konstanz, Germany

Supporting Information

ABSTRACT: The goal of this study was to adapt a recently proposed linear large-scale support vector machine to large-scale binary cheminformatics classification problems and to assess its performance on various benchmarks using virtual screening performance measures. We extended the large-scale linear support vector machine library LIBLINEAR with state-of-the-art virtual high-throughput screening metrics to train classifiers on whole large and unbalanced data sets. The formulation of this linear support machine has an excellent performance if applied to high-dimensional sparse feature vectors. An additional advantage is the average linear complexity in the number of non-zero features of a prediction. Nevertheless, the approach assumes that a problem is linearly separable. Therefore, we conducted an extensive benchmarking to evaluate the performance on large-scale problems up to a size of 175000 samples. To examine the virtual screening performance, we determined the chemotype clusters using Feature Trees and integrated this information to compute weighted AUC-based performance measures and a leave-cluster-out cross-validation. We also considered the BEDROC score, a metric that was suggested to tackle the early enrichment problem. The performance on each problem was evaluated by a nested cross-validation and a nested leave-cluster-out cross-validation. We compared LIBLINEAR against a Naïve Bayes classifier, a random decision forest classifier, and a maximum similarity ranking approach. These reference approaches were outperformed in a direct comparison by LIBLINEAR. A comparison to literature results showed that the LIBLINEAR performance is competitive but without achieving results as good as the top-ranked nonlinear machines on these benchmarks. However, considering the overall convincing performance and computation time of the large-scale support vector machine, the approach provides an excellent alternative to established large-scale classification approaches.



INTRODUCTION

The aim of this study was to investigate the performance of linear large-scale support vector machines (SVM) on large data sets of chemical compounds. To achieve this, we implemented common performance metrics for virtual screening and integrated these into an open source library for linear support vector machines published by Fan et al.[1] First, we compared the approach to a variety of parametrized and unparameterized approaches. Second, we assessed the generalization performance of the approach by excluding clusters of similar compounds from the training set. Third, the performance was compared to classifiers in the literature. Fourth, we considered the training time, regularization, and the impact of optimizing a virtual screening measure in the model selection of the SVM.

Large-scale learning approaches become increasingly important. Repositories of publicly large data sets of measured chemical compounds in computer-readable formats such as PubChem[2−4] or ChemDB[5] are growing. In most cases, the large-scale data sets have a challenging composition. They may contain more than 100000 chemical compounds and are unbalanced (e.g., an active to inactive ratio of 1:100) because only few of the tested compounds show an activity against a biological target.

Many machine learning approaches for modeling structure−activity relationships are not suited for large-scale learning tasks. For example, in the case of kernel machines, this is caused by the polynomial complexity of most kernels and the limited performance on large data sets because of a cubic computational complexity of the learning algorithm. Consequently, if there are more than several thousands of measured compounds, then there is the choice between training a model on the complete data set using a large-scale classifier or training a more sophisticated approach on a reduced data set. The latter means to sacrifice valuable information. Alternative straightforward approaches are ligand-based similarity searching or multiquery screening with group fusion[6] (over rank or similarity), which do not employ any supervised learning techniques. Examples of supervised large-scale algorithms applied in cheminformatics, which can handle large-scale problems, include Naïve Bayes, decision trees, and neural networks.[7] Bender et al. applied Bayesian learning to molecular fragments.[8] Han et al. trained decision trees as classifiers for structure−activity relationships on PubChem assay data.[9] A recently published

Received: July 20, 2009
Published: January 05, 2011

study by Swamidass et al. introduced the Influence Relevance Voter (IRV),[10] a supervised artificial neural network which can be trained on large data sets.

Linear large-scale SVMs are an actively researched field and have shown to work on large-scale learning applications in other research fields like text mining. The performance gain is achieved by a linear kernel with a specialized optimization routine. The major drawback compared to standard SVMs is that the approach works only in combination with a linear kernel (dot product). The major advantage is that the training time grows linearly with the number of samples. Like a standard SVM, the supervised learning algorithm solves a convex optimization problem, which minimizes the error of the discriminant function. The properties of the linear kernel are exploited by an efficient optimization routine[11] for the problem formulation. Another advantage of the linear SVM used in this study is the linear complexity of a prediction with the number of non-zero features. To exploit the advantages of the linear machine, we encoded the compounds as sparse molecular fingerprints. Chemical fingerprint encodings are usually sparse and high-dimensional, with the exception of predefined structural keys. Molecular fingerprints are fragment-type descriptors derived by, for example, graph search algorithms (depth-first search, breadth-first search), radial atom environments (e.g., Molprint2D[8]), or structural keys (e.g., MACCS Keys[12]).

The success of learning approaches on unbalanced large-scale problems cannot be measured with standard machine learning quality measures, like accuracy. In practice, a screening reveals few actives and many inactives. To evaluate the outcome of virtual high-throughput screening (vHTS) experiments, we mainly focused on the practically relevant first ranks of a screening (i.e., the early recognition problem) and the so-called chemotype enrichments.[13] Therefore, we extended the LIBLINEAR[1] library by three vHTS metrics: the arithmetic weighted AUC (awAUC) and the arithmetic weighted ROC Enrichment (awROCE)[14] (both weighted by chemotype clusters), as well as the BEDROC score.[15]

We tested the performance on different large-scale problems: the IJCNN 2007 data mining challenge,[16] a large Ames toxicity benchmark data set,[17] the MUV data sets,[18] a subset of problems from the virtual screening study of Nasr et al.,[19] and 11 targets of the MDDR[20] drug database. The result scores on all problems were often significantly better than the reference approaches random decision forest, Naïve Bayes, and maximum similarity ranking. Furthermore, the computation times for training and prediction were promising. The prediction quality on the data mining problems was comparable with the state-of-the-art.[10,17] However, the top-ranked results of the nonlinear algorithms were not achieved.

We conclude that the proposed method is a valuable approach to infer sensitive models from large-scale learning problems consisting of chemical compounds. The approach is suited for the early recognition as well as the chemotype enrichment on large-scale problems. Although this method assumes a linear separability of the data, the results on the benchmarks were convincing. The modified LIBLINEAR library and the molecule fingerprinting toolbox are available as open source programs or executable Java JAR files.

MATERIALS AND METHODS

First, we give a short review of the applied molecular encodings. Then we briefly discuss the metrics that are applied in the model selection to assess the model performance on large unbalanced data sets. Next, we introduce a recently published large-scale linear SVM and discuss the advantages of this machine learning algorithm for large data sets of samples with a high-dimensional sparse encoding. Finally, the protocol for data set preparation, chemotype clustering, and experimental setup is illustrated.

Molecular Encoding and Kernels on Nominal Features. Molecular fingerprints are a common way to encode molecules by mapping fragment-type features like linear walks, topological or geometrical p-point patterns, subgraphs, or radial atom environments to a large feature vector. Popular fingerprints are, for example, depth-first (DFS) fingerprints, structural keys, radial fingerprints, or other fragment-type fingerprints. Many graph kernels rely on such a representation, like the marginalized graph kernel,[21] optimal assignment kernel,[22] DFS fingerprint kernel,[23] or pharmacophore kernel.[24]

In this paper, we used DFS fingerprints and Molprint-like[8,25] fingerprint algorithms. We implemented all fingerprinting algorithms using the Chemistry Development Kit[26,27] (CDK) library.

The depth-first search (DFSx) fingerprint encodes a compound by all unique paths obtained by running an exhaustive DFS from every atom up to a defined depth x. During the search, vertices could be traversed multiple times, with the exception of cycles. The resulting features consist of all valid paths encoded by the sequence of vertices and bonds traversed during the search.

Our Molprint-like fingerprints were based on augmented atoms describing the local atomic neighborhood of each atom. Each augmented atom encodes the topological (R2Dx) or geometrical (R3Dx) shells around an atom up to a certain depth x, usually 2 or 3. The distances were taken from the all-shortest path matrix of the atoms for the topological Molprint or the binned Euclidean atom distance matrix for the geometrical Molprint. We included all shells as features with depth $\leq x$, resulting in $n \cdot x \in \mathbb{N}$ features for each fingerprint for a molecule with n heavy atoms. To discretize the geometrical distances, we calculated $(d_{ij})/(b)$, where d_{ij} is the Euclidean distance between two atomic coordinate vectors i and j, and b, a binning constant in Å. The resulting distance was then mathematically rounded to an integer value. Examples of the fingerprints are illustrated in Figure 1.

Most kernel approaches are too expensive to be used on large-scale molecular data because of the polynomial computation time of the kernel and the complexity of the kernel machine, which is between $O(n^2)$ and $O(n^3)$, where n denotes the number of input samples. All kernels on nominal features are closely related to molecular fingerprint methods and are based on the logical AND between two features. Convenient kernel functions include the Tanimoto, MinMax, and dot product kernel. When computed on chemical fingerprints as we have defined them, each of these functions can be thought of as a type of spectrum kernel. Graph kernels that are based on nominal representations and employ the dot product kernel on sets of nominal features can be estimated by sparse fingerprints. The basic difference is that a compound C is regarded as the full nominal feature spectrum by applying a fingerprinting algorithm f. Then the set of features of $f(C)$ is mapped to a sparse and high-dimensional vector by a hash function h (eq 1). The hash function is necessary to restrict the dimensionality p of the feature vector.

$$h \circ f : C \rightarrow \{0,1\}^p \tag{1}$$

Evaluation Metrics for Virtual Screening Experiments. A plethora of evaluation metrics for vHTS experiments were suggested[28] for measuring the overall performance and the early enrichment performance. Thus, it is convenient to benchmark

the results of a method using a combination of evaluation metrics.

We extended LIBLINEAR with three different metrics. Two of them are based on the Receiver-Operating-Characteristic (ROC). In the following, AUC denotes the area under the ROC curve, a measure for the evaluation of the performance on the complete data set. Another evaluation metric employing the ROC is the ROC enrichment (ROCE), which can be used as a measure to determine the early recognition performance. This metric defines the enrichment by the true positive ratio at a predefined false positive rate.[29,30] Additionally, the dependency on the ratio of actives and inactives is removed. The enrichment factor at x% (REF) is the fraction of actives found in the top x% of the known inactives multiplied by 100 (see Jain et al.[29] for further details). The BEDROC score[15] represents an early recognition metric using a decreasing exponential weighting function to reduce the influence of lower ranked structures. The α parameter of the BEDROC score defines the influence of the early recognition to the final score. Consequently, high α values increase the importance of top-ranked structures. For example, $\alpha = 100.0$ implies that 80% of the final BEDROC score is based on the performance in the first 1.5% percent of the ranked data set; $\alpha = 20.0$ implies that 80% of the final BEDROC score is based on the first 8% of the ranked data set. Additional details are described in the article by Truchon et al.[15]

To reduce the bias of enrichments of analogue structures (i.e., structurally highly similar molecules), we integrated the clustering information (in our case from a hierarchical linkage clustering using Feature Trees[31]) into the AUC as suggested by Clark and Webster-Clark.[32] We decided to use the arithmetic weighting scheme as proposed by Mackey et al.[13] This scheme assigns each active structure a weight, which is inversely proportional to the number of structures in this cluster. The result is a modified equation for the calculation of the sensitivity as shown in eq 2. N_{clusters} is the number of clusters, N_j is the number of structures in the j-th cluster, β_{ij} is 1 if the i-th active structure of the j-th cluster was found, and $w_{ij} = (1)/(N_j)$ is the weight of an active structure. This alternative sensitivity definition (SE_{aw}) can be used in combination with ROC curves. Therefore, the alternative definition can also be integrated into the ROC enrichment. We refer to the arithmetic weighted AUC, ROC, and ROCE as awAUC, awROC, and awROCE, respectively.

$$\text{SE}_{\text{aw}} = \frac{\sum_{j}^{N_{\text{clusters}}} \sum_{i}^{N_j} \beta_{ij} w_{ij}}{N_{\text{clusters}}} \tag{2}$$

There have been recent attempts for optimizing the performance of vHTS classifiers by large-scale performance metrics. For example, Swamidass et al. used the BEDROC to optimize the parameters of their classifiers.[10] A more general metric framework was presented by the same author[33], providing examples also of successfully trained classifiers using the so-called concentrated ROC.

Linear Support Vector Classification. LIBLINEAR is a large-scale linear SVM for classifying large data sets. It was published by Fan et al.[1] in 2008. For an input set of instance-label pairs $(\mathbf{x}_i, y_i)$, $i = 1,\ldots,l$, $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{-1,+1\}$, it optimizes the unconstrained optimization problem

$$\min_{w} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \xi(\mathbf{w}_i; \mathbf{x}_i, y_i) \tag{3}$$

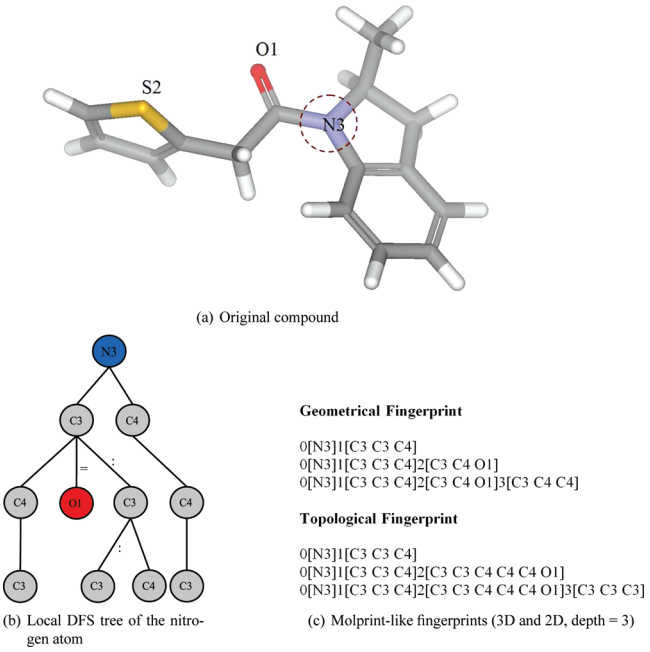


Figure 1. Fingerprint representations used in this study. In this example, the fingerprints describe the local environment of the nitrogen atom of a compound. The DFS path tree is shown in subfigure (b) where the edge labels ":" indicate an aromatic ring bond and "=" a double bond. The geometrical and topological shells of the Molprint-like fingerprint are regarded as separate features in subfigure (c). In all subfigures, the atom type was set to element symbol plus number of neighboring atoms.

where $\xi(\mathbf{w}_i; \mathbf{x}_i, y_i)$ is a loss function, and $C \geq 0$ is a penalty parameter. The two commonly used loss functions for SVMs are the L_1 and L_2 loss.

Hsieh et al.[11] proposed a new coordinate descent method to solve the dual problem[34] of optimizing the Lagrange multipliers α

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) = \frac{1}{2}\alpha^T\overline{Q}\alpha - \mathbf{e}^T\alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq U, \forall i \end{aligned} \tag{4}$$

where $\overline{Q} = Q + D$, D is a diagonal matrix and $Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$. For the L_1 loss, $U = C$ and $D_{ii} = 0$, $\forall i$. In case of the L_2 loss, $U = \infty$ and $D_{ii} = (1/2)C$ for all i.

Unlike other SVMs, LIBLINEAR does not use a nonlinear kernel function to map the input vectors into a high-dimensional feature space. The new dual coordinate descent method takes advantage of this fact. It optimizes α from an initial starting point $\alpha^0 \in \mathbb{R}^l$ through a sequence of vectors $\{\alpha\}_{k=0}^{\infty}$ until an ε-accurate solution is found. An outer step α^k to α^{k+1} is computed by l inner iterations updating α_i for all i. To determine if α_i needs to be updated, we have to calculate

$$\begin{aligned} \nabla_i f(\alpha) &= y_i \mathbf{w}^T \mathbf{x}_i - 1 + D_{ii}\alpha_i \\ \text{with} \quad \mathbf{w} &= \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j \end{aligned} \tag{5}$$

At this step, LIBLINEAR exploits the dot product kernel, as otherwise, the direct calculation of $\mathbf{w}$ would be computationally infeasible. The calculation of $\nabla_i f(\alpha)$ has a complexity of $O(\bar{p})$, where $\bar{p}$ is the average number of non-zero features per instance in a data set. As a result, the sparsity of the input vectors plays a

crucial role for the computation time needed for an optimization. The weight vector $\mathbf{w}$ can be maintained and updated throughout the process by

$$\mathbf{w} \leftarrow \mathbf{w} + (\alpha_i - \overline{\alpha}_i) y_i \mathbf{x}_i \tag{6}$$

where $\overline{\alpha}_i$ is the current value and α_i is the value after updating. In case of a nonlinear kernel, we would have to solve $\nabla_i f(\alpha) = (\overline{Q}\alpha)_i - 1$ instead. For a large-scale application, $\overline{Q}$ is generally too large to be stored in memory. This would require the calculation of the i-th row of $\overline{Q}$ at a cost of $O(l\overline{p})$. The computation becomes overly expensive in a practical application for a large l.

An additional speed-up compared to a standard kernel machine stems from the linear discriminant function learned by LIBLINEAR (eq 7).

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \tag{7}$$

The prediction of an unknown sample $\mathbf{x}$ can be computed by LIBLINEAR in $O(p_x)$, where p_x is the number of non-zero features in $\mathbf{x}$. Again, the sparsity determines the computation time needed for the prediction, not the dimension of the samples.

As a consequence, the most adapted input for LIBLINEAR are high-dimensional and sparse (sparseness in the number of non-zero features) vectors. Such data also speeds up the predictions because LIBLINEAR predicts with the averaged number of non-zero bits (e.g., at most 60 bits for a compound with 30 heavy atoms using R2D2 encoding with depth 2, or at most 90 bits in a vector for the R3D3 encoding with depth 3). Considering the prediction time, LIBLINEAR is the fastest classifier in this study.

The discriminant function of a standard kernel machine needs to calculate the kernel function $k(\mathbf{x},\mathbf{y})$ for all support vectors (eq 8). A evaluation of the dot product kernel generally requires $O(\overline{p})$, where $\overline{p}$ is the average number of non-zero features. This step becomes even more expensive if more sophisticated kernels are applied. Thus, the calculation of the decision value is expensive if the set of support vectors (i.e., the training samples where $\alpha_i \neq 0$) is large.

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{8}$$

For unbalanced data sets, the LIBLINEAR library allows to set an individual class penalty parameter W^{class} for each class. In an experiment of this study, the BEDROC score, awROCE, and awAUC were optimized by the model selection. During this process, a grid search was conducted to optimize the parameters for C and W^{class} for each metric.

Reference Classifiers. LIBLINEAR was compared against three supervised large-scale classifiers, which are frequently used in cheminformatics to solve large-scale retrieval problems. We considered a one-nearest-neighbor classifier and Naïve Bayes, both approaches are fast and unparameterized. The most sophisticated reference classifier in the comparison was the parametrized random decision forest.

Maximum Similarity-Based Ranking. In the following, we regard a data set D encoded by the fingerprinting function (eq 1) with encoded structures $\mathbf{q}, \mathbf{d} \in D$.

A maximum similarity based ranking sorts a data set according to an encoded query structure $\mathbf{q}$. For each query $\mathbf{q}$, the similarity to all compounds $\mathbf{d} \in D - \mathbf{q}$ in a database is computed using a real-valued similarity function s.

We chose the Tanimoto coefficient[35] as the similarity function s. For two binary fingerprints $\mathbf{c}, \mathbf{c}' \in D$, $s(\mathbf{c}, \mathbf{c}') \rightarrow \mathbb{R}$ computes the following similarity

$$s(\mathbf{c}, \mathbf{c}') = \frac{\sum_i^p c_i \wedge c_i'}{\sum_i^p c_i \vee c_i'} \in [0, 1] \tag{9}$$

This process can easily be extended to multiple queries. Here, a set of active compounds is compared to each structure in the database and ranked according to the highest similarity. In the following, this approach will be referred to as MAX-SIM. Despite its simplicity, the approach requires a computation time of $O(n^{1.6} \cdot \log k)$ using an efficient implementation,[36] where n equals the number of structures and k the number of neighbors.

Naïve Bayes Classifier. The Naïve Bayesian classifier (NBC) is trained with a data set of input vectors $\mathbf{x} \in \mathbb{R}^p$ under the assumption that the features $x_1,...,x_p$ of an input vector $\mathbf{x}$ are independent. Using Bayesian inference, the NBC predicts the probability $P(y_j|\mathbf{x})$ of a sample $\mathbf{x}$ belonging to class y_j and chooses the class with highest probability.

$$P(y_j|\mathbf{x}) = \frac{P(y_j) \prod_i P(x_i|y_j)}{P(\mathbf{x})} \tag{10}$$

The probabilities can be estimated in linear time in the number of non-zero features. This results in fast training times. However, the prediction is linear in the number of features, as the probability $P(x_i|y_j)$ has to be looked up, even if $x_i = 0$. For a large number of features, this can be computationally expensive.

Random Decision Forest. A random decision forest (RF) is an ensemble classifier of decision trees trained on random subspaces of the feature space. As Svetnik et al. published a detailed study of random decision forests in the field of cheminformatics,[37] we skip further details here. The random decision forest classifier is included to provide another parametrized reference classifier for assessing the performance of LIBLINEAR.

Complexity of Training and Prediction. We briefly review the complexity of the training and prediction phase of the classifiers. n equals the number of samples, p is a constant and equals the dimension of a fingerprint, and $\overline{p}$ denotes the average number of non-zero bits.

The complexity of LIBLINEAR is $O(n \cdot \overline{p})$ for n samples (considering an ε-accurate solution, where ε is a constant) and a gradient descent with constant complexity. In practice, the computation time also depends on ε and C (high values for C increase the training time). For further details on the computational complexity of the dual coordinate descent method of LIBLINEAR, please refer to the studies of Hsieh et al.[38] and Bottou and Bousquet.[39] The prediction time depends on the average of non-zero features of the samples to be predicted and is therefore $O(\overline{p})$.

The computation time for the random decision forest is $O(t \cdot p \cdot n \log n)$, a prediction requires $O(t \cdot p)$, where t is the constant number of decision trees and p the random feature space of a decision tree.

The Naïve Bayes classifier can be trained in one pass and does not require to adjust any parameters. The complexity of the NBC is $O(n \cdot p)$, where n denotes the number of samples and p the number of features. A prediction requires $O(p)$. Therefore, depending on p, the NBC can be slower than LIBLINEAR in some applications for a large p.

MAX-SIM does not require a training or a parameter optimization. The prediction depends on the n_a active samples in the

Table 1. Information on Data Sets with Random ChemDB Background[a]

abbreviations	actives	clusters[b]	singletons[c]	AD[d]	reference
benz	404	9	0	427	40
d2	330	20	0	523	41
cdk2	151	14	2	1143	41
cox2	125	4	0	1381	42
dhfr	755	10	0	229	40
estro	393	4	0	439	42
fxa	107	10	1	1613	41
hiv	417	35	4	414	16

[a] The total number of substances in the background data set was 172600. [b] Chemotype clusters of active compounds were identified by a complete linkage clustering with Feature Trees. [c] Singletons are clusters of size one and are included in the total number of clusters. [d] Inverse active to inactive ratio.

training set and the number of non-zero features. The classification can be conducted in $O(n_a \cdot \bar{p})$ for simple similarities such as the Tanimoto coefficient.

Benchmark Data Sets and Preparation Protocol. *Publicly Available Data Sets.* We used several publicly available large-scale data sets in this study. The first data set compilation consists of the benchmark problems with more than 100 confirmed actives from the study of Nasr et al.[19] The compilation is described in detail in Table 1. The data sets contain confirmed active compounds[40−42] against a target and a random background data set (175000 compounds) sampled from ChemDB.[43] The second major compilation, the Maximum Unbiased Validation (MUV) data sets, was introduced by Rohrer et al.[18] Each MUV data set contains 15000 decoy compounds and 30 active compounds. A third data set, the DTP HIVA Screen, was taken from the IJCNN 2007 data mining challenge. It contains 41066 inactive compounds and 1465 active compounds. According to the competition rules, we included the "CM" (moderately active) labeled samples in the set of actives for the HIVA data set. Finally, we conducted benchmarks on a large Ames toxicity data set[17] consisting of 6512 compounds.

The data sets were compiled and prepared as follows. To compute an initial geometry for each compound, we applied CORINA3D.[44] The resulting seed structures were further optimized by Schrödinger MacroModel 9.6[45] and the OPLS 2005 forcefield using the Broyden−Fletcher−Goldfarb−Shanno minimization method with a convergence criterium (rms gradient of 0.0001) and a maximum of 1000 iterations. All structures that could not be processed by CORINA3D or Schrödinger MacroModel 9.6 were filtered out. In the worst case, about 5% of the compounds of the HIVA data set were filtered out.

MDDR Data Sets. We used 11 published lists of known ligands[46] for various targets of the MDDR 2006 drug database (Table 2) to set up another benchmark compilation. Each large-scale problem comprised a different activity class and the remaining structures as inactive samples. During the preprocessing, all duplicates were removed from the MDDR database. The seed structures were generated using CORINA3D from their stereochemical canonical SMILES representation. The resulting seed structures were further optimized with Schrödinger MacroModel 9.6 with the same protocol as for the public data sets.

Chemotype Clustering. To avoid artificially high enrichments by similar compounds of the same chemotype, we clustered the compounds in the active class.

Table 2. Information on MDDR Data Sets[a]

abbreviations	activity class	actives	clusters[b]	singletons[c]	AD[d]
5HT1A	5HT1A agonists	797	25	1	167
5HT3	5HT3 antagonists	729	18	0	183
5HT	5HT reuptake inhibitors	341	14	0	392
AT2	angiotensin II antagonists	932	13	0	143
COX	COX inhibitors	627	22	0	213
D2	D2 antagonists	389	15	0	343
HIV	HIV protease inhibitors	698	17	0	191
PKC	protein kinase C inhibitors	433	19	1	308
renin	renin inhibitors	899	10	0	148
SP	substance P antagonists	1200	24	1	111
thrombin	thrombin inhibitors	759	29	0	175

[a] The total number of compounds in this data set was 133937. [b] Chemotype clusters of active compounds were identified by a complete linkage clustering with Feature Trees. [c] Singletons are clusters of size one and are included in the total number of clusters. [d] Inverse active to inactive ratio.

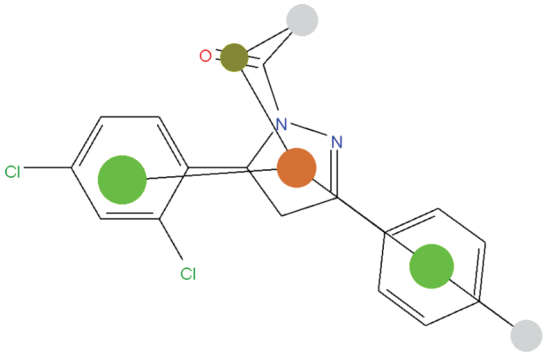


Figure 2. The encoding of compounds as trees ensures a high similarity for compounds with similar scaffolds and similar placed topological features.

The active structures were divided into disjoint chemotype clusters by applying an approach similar to the automated generation of chemotypes by Good and Oprea,[14] which is based on a clustering of reduced graphs. The Feature Tree algorithm[31] (FTrees version 2.0.1) converted each structure into a reduced graph representation as shown in Figure 2. Based on this representation, the match-search algorithm of Feature Trees calculated all pairwise similarity values. After this step, a hierarchical complete linkage clustering with a similarity threshold of 0.65 was performed.

The resulting cluster information is summarized in Table 1 and Table 2.

Experimental Setup. *Molecular Fingerprints.* We employed three molecular fingerprint types in the experiments. First, we used a DFS fingerprint containing all paths up to depth 8 (DFS8). The DFS fingerprint contained additional bond types (single, double, triple, and aromatic) as bond label. Second, we applied a Molprint-like fingerprint algorithm for 2D shells up to depth 2 (R2D2) and 3D shells up to depth 3 (R3D3). In the experiments, the atoms were labeled by a concatenation of the element symbol plus the count of neighboring heavy atoms. R3D3 employs an 1 Å binning. Figure 1 shows examples of the geometrical and topological features of the R2D2, R3D3, and DFS fingerprints. The resulting patterns were hashed to a binary fingerprint using the default string hash function of Java 1.6.

If not noted otherwise, the hash function for the patterns used a mapping space of 2.[18] Preliminary tests showed that a choice of $\log_2 p$ with $p > 12$ results in nearly collision-free fingerprints for the applied encodings.

To produce a fair setup for the direct comparison in the result section between LIBLINEAR and the reference classifiers, the compounds were hashed to 5kbit fingerprints using the R2D2 encoding. The main reason for this is that the number of features is a multiplier of the complexity of the NBC and the random decision forest (because the size of the random subspace for the trees must be increased for high values of p). Consequently, all machine learning approaches in the direct comparison benchmarks work on the same representation.

Cross-Validation, Model Selection, and Evaluation. For all evaluations, the data sets were split using a seeded random number generator. Thus, all folds for all methods are equal, making the performance of the methods comparable.

To assess the quality of the machine learning algorithms, we performed a stratified nested cross-validation. The validation consisted of an inner model selection for optimal parameters and an outer loop for external validation. The parameters of the parametrized learning algorithms were optimized by an inner model selection with a grid search. The optimal parameter combination was determined in the inner loop of the cross-validation within a 2-fold cross-validation. The performance of the selected parameters was evaluated within an outer 10-fold cross-validation. To avoid a bias induced by the initial seed for the random number generator, we conducted the validation over 10 runs (10 × 10 cross-validation). Therefore, each validation run resulted in 100 external evaluations on equal test sets. The results on these test sets can be compared by a paired statistical test.

We performed a model selection to optimize the performance of the classifiers. It determined the best LIBLINEAR parameters: the weight of the negative class ($\log_2 W^- \in \{-5, -4,...,0\}$) and the SVM C parameter ($\log_2 C \in \{-5, -4,...,1\}$). For the random decision forest, the number of trees ($\log_2 t \in \{2,3,...,7\}$) and the size of the feature space for each tree ($\log_2 f \in \{3, 4,...,7\}$) was optimized. Parameters were selected according to their AUC performance. For both classifiers, the model selection was evaluated within a 2-fold cross-validation.

The different evaluation metrics were computed either by a ranking of the LIBLINEAR decision function (eq 7), overall vote of the random decision forest, Naïve Bayes probability estimates (eq 10), or MAX-SIM similarity (eq 9). This is crucial because the evaluation relies on a smooth ranking of the samples. Additionally, it allows for an interpretation of the classifier's confidence about its prediction, which is especially important for a practical application.

For the MUV benchmark set, we did not use active cluster weights for the computation of awAUC and awROCE because of the small number of clusters (typically between 1 and 5).

To avoid a bias introduced by obvious similarities, we also conducted a nested leave-cluster-out cross-validation. For this setup, the folds are generated by splitting the data set into k folds, where k is the number of valid clusters. A valid cluster is a cluster of actives with more than four samples. The samples of smaller clusters were removed from the benchmarks. The training is conducted on the folds with the $k - 1$ clusters, without samples from the cluster left out. Using the resulting model, the remaining external fold is predicted. This process is repeated for all clusters, consequently, k models are trained for the evaluation.

Beside the AUC-based performance metrics, we also present the relative enrichment factor performance at 1%. It reflects the percentage of actives retrieved in the first percent of the ranking.

Statistical Significance. The statistical significance for the 10 × 10 cross-validation was evaluated by the corrected resampled t-test.[47] The corrected resampled k-fold cross-validation test uses the following statistic

$$t = \frac{\frac{1}{k \cdot r}\sum_{i=1}^{k}\sum_{j=1}^{r} x_{ij}}{\sqrt{\left(\frac{1}{k \cdot r} + \frac{n_2}{n_1}\right)\hat{\sigma}^2}}$$

Here, for the evaluation of r runs of a k-fold cross-validation, n_1 is the number of instances used for training, n_2 the number of instances used for testing, x_{ij} is the difference observed for fold i and run j, the variance $\hat{\sigma}^2 = 1/(k \cdot r)\sum_{i=1}^{k}\sum_{j=1}^{r}(x_{ij} - m)^2$, and the mean is given by $m = 1/(k \cdot r)\sum_{i=1}^{k}\sum_{j=1}^{r} x_{ij}$. For the comparison of multiple classifiers, the performance was tested at a Bonferroni-corrected p-value $p_{\text{corr}} \leftarrow p/c$, where c is the number of classifiers compared. $p \leq 0.05$ was considered to indicate a statistically significant difference.

Implementations. The LIBLINEAR Java port was modified to optimize computation time and memory usage. To reduce the computation time of the cross-validation on multicore CPUs, the cross-validation runs were threaded to enable a parallelized computation of the folds. The memory usage was minimized by loading a data set once and then creating references only on it for the different cross-validation folds. In addition, the features were stored as float values instead of double values. The memory requirements were further reduced by a compression of the feature vectors. All features that were zero for a complete training and prediction set were dynamically omitted, resulting in a reduced weight vector **w**.

For the direct comparisons in the results section, we chose the WEKA 3.7[48] implementations of the NBC and the random decision forest.

Availability of Source Code, Tools, and Data Sets. The machine learning library applied in this study is a modified version of the LIBLINEAR Java port by B. Waldvogel.[49] It incorporates all screening metrics described in this paper. The LIBLINEAR modification and the molecular fingerprinting tool are freely available as source code and executable Java JAR file from the following link http://www.ra.cs.uni-tuebingen.de/software/ChemLL/. The fingerprint algorithms use the CDK library.[26,27] The implementations of the reference classifiers Naïve Bayes and random decision forest were provided by the open source machine learning library WEKA[48] 3.71.

RESULTS

Direct Comparison with a Nested 10-fold Cross-Validation. Table 3 presents the results of the nested 10-fold cross-validation. The parameters of the parametrized classifiers were optimized according to the AUC performance. The significance of the performance was determined by a paired t-test on the mean outcome on each problem described in Table 1 and Table 2. According to this setup, LIBLINEAR was among the best methods over all data sets. The NBC was outperformed by the remaining methods on all benchmarks. MAX-SIM was competitive on all CHEMDB and MDDR benchmarks but was outperformed on the MUV data sets by LIBLINEAR. The random decision forest

Table 3. Results of Nested 10 × 10 Cross-Validation on CHEMDB, MDDR, and MUV Data Sets Using Different Performance Measures and R2D2 Fingerprints[a]

measure	data set	LIBLINEAR	RF	NBC	MAX-SIM
awAUC[b]					
	CHEMDB	**0.99**	0.67	0.97	**0.98**
	MDDR	**0.99**	**0.99**	0.95	**0.99**
	MUV	**0.77**	0.60	0.63	0.73
REF[c]					
	CHEMDB	**93.71**	34.68	88.98	**92.72**
	MDDR	**89.40**	**91.52**	69.02	**92.83**
	MUV	**27.44**	23.69	23.08	23.18
awROCE[d]					
	CHEMDB	**92.01**	31.77	86.60	**90.80**
	MDDR	**89.72**	**91.32**	68.75	**93.75**
	MUV	**39.32**	32.27	27.62	29.54
BEDROC[e]					
	CHEMDB	**0.92**	0.36	0.86	**0.91**
	MDDR	**0.88**	**0.91**	0.71	**0.90**
	MUV	**0.29**	0.21	0.20	0.24

[a] The models for the parameterized methods were optimized by the AUC. Significantly best performing methods are indicated by **bold** font. [b] awAUC for the MUV data sets equals the AUC performance. [c] Relative enrichment factor at 1.0%. [d] awROCE at 1.0%, based on the AUC for MUV. [e] $\alpha = 53.6$ for the MUV data set because of the smaller data set size.

classifier showed a convincing performance on the MDDR benchmarks but was clearly outperformed by LIBLINEAR and MAX-SIM on the CHEMDB problems and the MUV data set.

Compared to the previous results, the performance on the MUV data sets was moderate. The main reason for this is that the active samples of the MUV problems were filtered to exclude obvious similarities.[18] LIBLINEAR outperformed the other approaches on this benchmark compilation significantly.

A random classifiers achieves an averaged awAUC of 0.50, a REF of 0.97, a BEDROC of 0.01, and an awROCE of 0.96.

Direct Comparison with a Nested Leave-Cluster-Out Cross-Validation. To assess the generalization performance for unknown scaffolds, we implemented a leave-cluster-out cross-validation as described in the experimental setup. The leave-cluster-out cross-validation protocol splits the data set into k folds according to k structural clusters. Therefore, the problems cannot be solved by regarding obvious similarities, which is reflected in the decreased performance on the data sets in Table 4. The MUV data sets were not regarded in this experiment because of the small number of active compounds and clusters (this is due to the MUV preparation protocol, which removes similar active compounds). Likewise, the awAUC and awROCE were not regarded because the cluster information is already considered by the leave-cluster-out cross-validation.

The AUC performance was decreased for all classifiers. However, the relative performance of the algorithms compared to the results of the standard cross-validation presented in Table 3 was similar. LIBLINEAR outperformed the RF and NBC on the CHEMDB problems and was not significantly worse than RF and MAX-SIM on the MDDR benchmarks.

According to the relative enrichment factor, LIBLINEAR significantly outperformed the other approaches on the CHEMDB benchmark and showed a competitive performance to MAX-SIM and RF on the MDDR benchmark. The NBC was outperformed on every benchmark, and RF was only competitive on the MDDR data set.

Table 4. Results of the Leave Cluster-Out Cross-Validation for CHEMDB and MDDR Data Sets Using R2D2 Fingerprints[a]

measure	data set	LIBLINEAR	RF	NBC	MAX-SIM
AUC					
	CHEMDB	**0.93**	0.77	0.88	**0.91**
	MDDR	**0.89**	**0.89**	0.79	**0.90**
REF[b]					
	CHEMDB	**68.87**	29.89	59.91	58.23
	MDDR	**45.58**	**49.06**	33.26	44.65
BEDROC[c]					
	CHEMDB	**0.70**	0.33	0.62	0.60
	MDDR	**0.50**	**0.52**	0.38	**0.50**

[a] Only clusters with more than five samples were regarded. The models for the parameterized methods were optimized by the AUC. Significant best performing methods are indicated by **bold** font. [b] Relative enrichment factor at 1.0%. [c] $\alpha = 53.6$.

LIBLINEAR exhibits the overall best BEDROC performance. It significantly outperformed the reference methods on the CHEMDB problems and was competitive with RF and MAX-SIM on the MDDR benchmark.

A random classifier achieves an averaged AUC of 0.50, an REF of 1.18, and a BEDROC of 0.02 over all benchmarks with this setup.

Comparison against Literature Results. We also evaluated the approach on the HIVA data set according to the IJCNN 2007 competition rules. An important issue of this data set is that the problem size was not adjusted for large-scale approaches. The training set consists of 3845 samples, 128 of them active or moderately active. The results according to the HIVA challenge competition rules were: On the external test set LIBLINEAR achieves an AUC of 0.777 (DFS8), 0.764 (R2D2), and 0.766 (R3D3). The balanced error rate (BER) was 0.282 (DFS8), 0.288 (R2D2), and 0.297 (R3D3). Swamidass et al. published the following results for the IRV: AUC of 0.762 on the test set and a BER of 0.271. On the complete HIVA data (10 × 10 cross-validation), they achieve an AUC of 0.845 within a 10 × 10 cross-validation; using the same setup (nested 10 × 10 cross-validation with 2-fold cross-validation for parameter selection), LIBLINEAR achieves an AUC of 0.836 ± 0.022 (DFS8), 0.830 ± 0.023 (R2D2), and 0.838 ± 0.021 (R3D3). The winner of the challenge used a nonlinear SVM, achieving a AUC of 0.93 on the training set and an AUC of 0.88 on the test set; the BER was 0.264. The median BER of the IJCNN 2007 submissions was 0.31; thus, LIBLINEAR would have been ranked in the midtable of this challenge.

Another large benchmark data set was published by Hansen et al.[17] containing 6512 compounds with measured Ames toxicity. This data set differs from the previous benchmarks because it is balanced, and it provides defined folds for a 5-fold nested cross-validation. Furthermore, Hansen et al. published the AUC values for several nonlinear machine learning approaches using dragonX 1.2 descriptors.[50] The best performing approaches in the study were Gaussian Processes (AUC of 0.84 ± 0.01) and a standard SVM in combination with a radial basis function kernel (AUC of 0.86 ± 0.01). LIBLINEAR achieves an AUC of 0.78 ± 0.01 (DFS8), 0.85 ± 0.01 (R2D2), and 0.83 ± 0.01 (R3D3) on the defined splits.

Table 5. Computation Times of Complete LIBLINEAR Model Selection for Parameters C and W^- on Different Data Sets Using Different Fingerprints[a]

data set	DFS8	R2D2	R3D3
HIVA	5 h 03 min	1 min 26 s	1 min 49 s
MUV[b]	30 min	4 min 21 s	3 min 32 s
ChemDB[b]	—[c]	10 min 07 s	7 min 21 s
MDDR[b]	1 h 21 min	3 min 03 s	3 min 08 s

[a] The experiments were performed on cluster nodes configured with two Dual Core AMD Opteron(tm) Processor 280 (2.4 GHz), 2 GByte RAM, Java Runtime Environment 1.6, and Linux kernel 2.6.18. The modified LIBLINEAR library is threaded to exploit multiple cores (4 in the given configuration). [b] Averaged empirical computation times over all data sets of this type. [c] DFS fingerprint was not used for these data sets.

Empirical Computation Times. The computation time for a complete model selection of LIBLINEAR using different encodings is summarized in Table 5. During the experiments, each model selection included $48 \times 10 = 480$ temporary models by applying an extended parameter grid compared to the previous experiments within a 10-fold cross-validation. The goal of the model selection was to fine-tune C and W^- like for a practical application.

We observed significant differences between the computation times for the parameter selection for different encodings. For the Molprint-like encodings, the model selection could be computed within 1 min 26 s and 1 min 49 on HIVA. Using the DFS8 encoding, this process took 5 h and 3 min. This is caused by a complexity of $O(n \cdot \alpha^d)$ for the DFS8 encoding, where α is the branching factor. α depends on the average degree of the vertices and is slightly above 1 for hydrogen-depleted graphs of small molecules.[23] n is the number of heavy atoms, and d is the search depth. In contrast, Molprint-like fingerprints grow linearly with the number of heavy atoms. As the training time for LIBLINEAR relates to the average number of non-zero features (caused by the update step for $\mathbf{w}$, see the section about the linear support vector classification), the Molprint-like features decrease the computation time.

In this paragraph, we assess the empirical training times in comparison to the random decision forest. The computation time in practical applications depends on multiple variables, like the parameter grid of the model selection for the parametrized classifiers or the dimensionality of the input vectors. To get an impression of the performance on a standard cluster node (single core on an AMD Opteron 280, 2.4 GHz, Scientific Linux with JRE 1.6), we present the averaged computation time on the CHEMDB benchmarks. The encoding for this experiment was a 5 kbit R2D2 fingerprint. The random decision forest was configured with 100 trees with a 128 feature subspace; the parameters for LIBLINEAR were determined in a model selection within a 2-fold cross-validation with $\log_2 C \in \{-5,...,1\}$ and $\log_2 W^{-1} \in \{-5,...,1\}$. Using a single CPU core, the average computation time for training a model on a stratified 90% split was 702 s for the random decision forest and 40 s for LIBLINEAR with parameter tuning. The computation times for the unparameterized Naïve Bayes and MAXSIM classifiers are negligible.

Feature Space and Regularization. To assess the quality of the different fingerprints used in this study, we counted the set features in the encoding using a hash space of size 2.[18] For this experiment, we downloaded the lead-like subset of ZINC[51] version 8, computed the described features and counted the percentage coverage by the OR connection of each bit position. The unset bits were for DFS8 0.00%, for R2D2 48.51%, and for R3D3 0.02%. Yet, the coverage for the R2D2 encoding could be increased by extending the neighborhood.

It is also interesting to know how the models were regularized because of the size of the data sets and the linear separation of the dot product kernel. For example, on the MDDR drug data set subsets AT2 (Angiotensin II Antagonists) and D2 (D2 Antagonists) the percentage of support vectors in an optimized model was between 2.1% (AT2, R2D2, model selection for BEDROC) and 13.9% (D2, R3D3, model selection for awAUC). The percentage of support vectors for the HIVA data set was in between 39% (DFS8, AUC) and 51% (R3D3, AUC).

In the extreme case of the HIVA data set, the problem was obviously poorly regularized. In general, the regularization depends on the diversity of the input samples, weights for C, encoding of the samples, and overall difficulty of the learning task (which also depends on the kernel function and noise of the data). However, the predictive performance of the R3D3 and DFS8 on this data set was comparable, although the number of support vectors for R3D3 was increased.

Model Selection by Optimizing the vHTS Performance Measures. As a consequence of optimizing the weight W^- for the inactive class, LIBLINEAR intuitively enables a model selection of the vHTS measures that should make the model more sensitive for the active class. The goal of the following experiments was to investigate whether a model selection of the vHTS measure improves the vHTS performance.

Accordingly, we optimized the awROCE at 1%, awAUC, and BEDROC with $\alpha = 100.0$ in the model selection for LIBLINEAR on the MDDR and CHEMDB data sets and compared the result for the respective performance measures with those obtained by using the remaining measures. Again, we conducted a nested 10-fold cross-validation to ensure that these results were not influenced by overfitting. For tuning the AUC, we observed in one case that tuning the BEDROC lead to a significantly worse AUC on MDDR 5HT1A (0.992 ± 0.004 vs 0.990 ± 0.005). For tuning the awAUC, there were no significant differences. There was a significant worse result with respect to the awROCE by selecting the best parameters for the BEDROC (95.059 ± 3.848 vs 97.160 ± 2.848) on MDDR AT2. On six of the MDDR data sets the BEDROC parameter selection lead to significant better BEDROC scores than at least one of the other metrics: On MDDR 5HT1A, it performed significantly better than awAUC and AUC; on MDDR 5HT3, better than the AUC; on AT2, better than all other metrics; on MDDR HIV, better than AUC; and finally, on MDDR SP and MDDR Thrombin, better than the other measures. The most significant difference was observed on the MDDR 5HT1A with a BEDROC of 0.762 ± 0.038 for a model selection with the AUC and 0.803 ± 0.030 for a model selection with the BEDROC.

In summary, there was only one benchmark where the AUC performance was decreased by a model selection for a vHTS performance measure. There were six benchmarks sets, where tuning the BEDROC lead to a significantly increased BEDROC performance. On one data set, the model selection for the BEDROC score significantly decreased the awROCE performance in comparison to tuning the awROCE directly.

DISCUSSION AND CONCLUSION

The focus of this study was to evaluate the performance of a linear large-scale support vector machine on large virtual

screening benchmark problems. Large-scale problems become increasingly important because of the growing number of measured compounds available in computer-readable formats. Many machine learning algorithms are limited in the size of the training set. Hence, there is a need for large-scale approaches which can be trained on such problems.

On the HIVA literature data set, LIBLINEAR performance was comparable to the AUC performances of a nonlinear SVM and the IRV. However, the training data of the HIVA data set does not favor LIBLINEAR because the number of training samples was adjusted to be used by a variety of learning approaches. The performance of LIBLINEAR boils down to a standard linear SVM on small data sets, which is reflected by the performance on HIVA. Nevertheless, the performance shows that LIBLINEAR is competitive, even in this situation. A further problem of the HIVA benchmark is that this data set has only one defined external test set, and the training set contains comparatively few samples, which may be lucky for some machine learning approaches. In fact, this is an evaluation on a single-sampled external fold. This is overcome by the second benchmark set, the large Ames toxicity benchmark. The data set was split into five defined subsets and a static training set, which allows for a nested 5-fold cross-validation. On this benchmark, LIBLINEAR had a comparable performance with Gaussian Processes and with a nonlinear SVM with a radial basis function kernel. All the same, the parameter tuning, training, and prediction can be conducted for LIBLINEAR within about a minute for all five models. Summing up the comparison against the literature benchmarks, we may conclude that LIBLINEAR performs comparable on the data sets but without achieving the top-ranked results of the nonlinear machine learning algorithms. Obviously, the performance is a trade-off between the training time and prediction quality.

There are also some trade-offs between LIBLINEAR and the IRV. A major point for discussion compared to the IRV is related to the computation of the k nearest neighbors, which has complexity of $O(n^{1.6} \log k)$. Consequently, for a fast cross-validation the k-nearest neighbors have to be precomputed. There is no need for LIBLINEAR to determine the k-nearest neighbors for the prediction of an unknown compound. Moreover, the prediction time is linear in the number of non-zero bits and independent from the number of support vectors. As a major drawback compared to the IRV, LIBLINEAR suffers from an inherent issue of kernel machines: the results are hardly interpretable, whereas the IRV addresses this problem.

The results of the direct comparison showed that the linear SVM had the overall best performance regarding the different performance measures. Naïve Bayes was clearly outperformed. MAX-SIM produced comparable results on the CHEMDB and MDDR benchmarks regarding the results from the nested cross-validation. MAX-SIM achieved significantly worse results on the more challenging MUV data sets. Similar results were obtained using the leave-cluster-out setup. As an alternative parametrized classifier, random decision forest showed convincing results on the MDDR data set using cross-validation and the leave-cluster-out validation protocol. It was outperformed on the MUV data set and the CHEMDB benchmarks. Both data sets are more diverse than the MDDR, which could be a reason for the performance of random decision forest. Another problem was observed by a closer look at the values of the consensus vote: The class of the active samples could hardly be separated by the consensus vote of the trees for these problems. In contrast to the random decision forest, the SVM formulation allows to adjust weights on unbalanced problems. Another problem of random decision forests stems from the dimensionality of the given data. With an increasing dimensionality of the fingerprints, the random forest classifier needs more trees and an increased size of the random feature space. Each parameter is a multiplier with respect to the complexity of the random forest classifier. Configured with the maximum number of trees and feature space of our parameter grid (both set to 2^7), the random decision forest classifier was considerably slower than the linear support vector machine. Therefore, we think that this linear large-scale SVM is an approach that could be increasingly important for solving large-scale learning tasks such as training models on complete assay outcomes.

In some cases, the early enrichment could be improved by optimizing the BEDROC score. In general, the performance was not influenced by a model selection with a specific performance measure. In spite of that, we suggest to optimize the BEDROC because it increased the BEDROC performance significantly on six benchmark sets without decreasing the performance of the other AUC measures significantly. Furthermore, the BEDROC score is independent from structural cluster information.

LIBLINEAR has several major advantages and disadvantages. To start with the disadvantages, it sacrifices the possibility to use other kernels than the dot product kernel, which assumes that the problem can be linearly separated. In comparison to the AUC performance of nonlinear machines, this is a drawback. Furthermore, the models are hard to interpret; however, it is possible to regard the weight for a feature stored in the optimized weight vector as an indicator of the importance of the corresponding feature. Finally, it shares another problem of support vector machines, namely that a model has to be properly parametrized by the class weights and the C parameter. The advantages of large-scale linear SVMs become obvious if the size of a training set renders a training of nonlarge-scale approaches computationally infeasible. The complexity of the LIBLINEAR solver grows linearly with the number of samples and is therefore suited for training on large data sets without removing samples. Support vector machines have solid theoretical foundations and have shown to have an excellent performance on many applications in cheminformatics. The algorithm solves a defined convex optimization problem, which implies that the algorithm converges to an optimal solution. This was corroborated by the performance of LIBLINEAR on the benchmark problems where it outperformed the remaining state-of-the-art approaches. It also seems to work well on training sets that have a highly imbalanced class distribution. Finally, the solver is especially suited for large-scale learning tasks with high-dimensional and sparse encodings because of the definition of the optimization problem and its kernel function. This property favors the use of sparse hashed fingerprints.

The computation time required for training a model depends on the average number of non-zero features of the samples: The more bits are set in a binary fingerprint, the higher becomes the computation time for LIBLINEAR (Yet, this is also true for any approach employing a similarity measure). The cost for the update step of the weight vector is the reason for this behavior. The effect can be observed on the benchmarks for the DFS8 encodings. Consequently, the best computation times are achieved on input vectors with few features. Nevertheless, the training time with radial fingerprints was convincing.

The approach suffers from minor problems regarding the representation of the samples and the linear learning approach itself. The compounds have to be encoded by sparse molecular fingerprints to apply LIBLINEAR. The feature mappings by the

hash function lead to an inherent loss of information. However, the probability of a collision can be reduced by choosing a suitably large feature space. Therefore, a suitable combination of feature representation and hash function is needed to cover this space. Another disadvantage is inherent in the assumption that a learning problem is linearly separable. This restriction can be compensated by using a sufficiently high dimension for the input vectors. The results indicate that in practice it does not seem to be a substantial problem for the fingerprint encodings of chemical compounds.

The basic LIBLINEAR solver has been published by Fan et al.[1] as open source in C and has been ported to Java.[49] We modified the Java version by extending it by the performance measures used in this study. For example, with our version, it is possible to train a model with parameters optimized by the BEDROC score, evaluate the model by the alternative AUC measures, store the model, and predict an external test file. Recently, a LIBLINEAR wrapper has been added to WEKA,[48] starting with the version 3.6.

To conclude, the described linear large-scale support vector machine is a supervised machine learning approach suitable for training models on large classification problems. The optimization of the problem-specific metrics from the vHTS literature was useful to adapt the parameter selection for this general machine learning algorithm to unbalanced large-scale problems.

■ ASSOCIATED CONTENT

Supporting Information. Lists of compound identifiers for the HIVA data set, EXTREG numbers for the MDDR database as well as the cluster files for the MDDR and CHEMDB problems. This information is available free of charge via the Internet at http://pubs.acs.org/.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: georg.hinselmann@uni-tuebingen.de.

Funding Sources

This work has been partially funded by Nycomed GmbH, Konstanz, Germany.

■ ACKNOWLEDGMENT

The authors thank the developers of the LIBLINEAR library for sharing their source code with the community. We thank B. Waldvogel for porting the LIBLINEAR to Java, where we could integrate it into our existing libraries. We thank Dr. C. Hinselmann and S. Zitzer for carefully proofreading the manuscript. We thank three anonymous reviewers, whose reviews greatly improved the manuscript.

■ REFERENCES

(1) Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; Lin, C.-J. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* **2008,** *9*, 1871–1874.

(2) Chen, B.; Wild, D. J. PubChem BioAssays as a data source for predictive models. *J. Mol. Graphics Modell.* **2010,** *28*, 420–426.

(3) Wild, D. J.; Hur, J. PubChemSR: A search and retrieval tool for PubChem. *Chem. Cent. J.* **2008,** *2*, 11.

(4) Wang, Y.; Xiao, J.; Suzek, T. O.; Zhang, J.; Wang, J.; Bryant, S. H. PubChem: A public information system for analyzing bioactivities of small molecules. *Nucleic Acids Res.* **2009,** *37*, 1–11.

(5) Chen, J.; Swamidass, S. J.; Dou, Y.; Bruand, J.; Baldi, P. ChemDB: A public database of small molecules and related chemoinformatics resources. *Bioinformatics* **2005,** *21*, 4133–4139.

(6) Whittle, M.; Gillet, V. J.; Willett, P.; Loesel, J. Analysis of data fusion methods in virtual screening: Similarity and group fusion. *J. Chem. Inf. Model.* **2006,** *46*, 2206–2219.

(7) Melville, J. L.; Burke, E. K.; Hirst, J. D. Machine learning in virtual screening. *Comb. Chem. High Throughput Screening* **2009,** *12*, 332–343.

(8) Bender, A.; Mussa, H. Y.; Glen, R. C.; Reiling, S. Similarity searching of chemical databases using atom environment descriptors (MOLPRINT 2D): Evaluation of performance. *J. Chem. Inf. Comput. Sci.* **2004,** *44*, 1708–1718.

(9) Han, L.; Wang, Y.; Bryant, S. Developing and validating predictive decision tree models from mining chemical structural fingerprints and high-throughput screening data in PubChem. *BMC Bioinf.* **2008,** *9*, 401.

(10) Swamidass, S. J.; Azencott, C.-A.; Lin, T.-W.; Gramajo, H.; Tsai, S.-C.; Baldi, P. Influence relevance voting: An accurate and interpretable virtual high throughput screening method. *J. Chem. Inf. Model.* **2009,** *49*, 756–766.

(11) Hsieh, J.-H.; Wang, X. S.; Teotico, D.; Golbraikh, A.; Tropsha, A. Differentiation of AmpC beta-lactamase binders vs. decoys using classification k NN QSAR modeling and application of the QSAR classifier to virtual screening. *J. Comput.-Aided Mol. Des.* **2008,** *22*, 593–609.

(12) Durant, J. L.; Leland, B. A.; Henry, D. R.; Nourse, J. G. Reoptimization of MDL keys for use in drug discovery. *J. Chem. Inf. Comput. Sci.* **2002,** *42*, 1273–1280.

(13) Mackey, M. D.; Melville, J. L. Better than random? The Chemotype enrichment problem. *J. Chem. Inf. Model.* **2009,** *49*, 1154–1162.

(14) Good, A. C.; Oprea, T. I. Optimization of CAMD techniques 3. Virtual screening enrichment studies: A help or hindrance in tool selection? *J Comput.-Aided Mol. Des.* **2008,** *22*, 169–178.

(15) Truchon, J.-F.; Bayly, C. I. Evaluating virtual screening methods: Good and bad metrics for the "early recognition" problem. *J. Chem. Inf. Model* **2007,** *47*, 488–508.

(16) NIH, DTP AIDS Antiviral Screen, 1999. http://dtp.nci.nih.gov/ (accessed August 25, 2010).

(17) Hansen, K.; Mika, S.; Schroeter, T.; Sutter, A.; ter Laak, A.; Steger-Hartmann, T.; Heinrich, N.; Müller, K.-R. Benchmark data set for in silico prediction of ames mutagenicity. *J. Chem. Inf. Model.* **2009,** *49*, 2077–2081.

(18) Rohrer, S. G.; Baumann, K. Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data. *J. Chem. Inf. Model.* **2009,** *49*, 169–184.

(19) Nasr, R.; Swamidass, S. J.; Baldi, P. Large scale study of multiple-molecule queries. *J. Cheminf.* **2009,** *1*, 7.

(20) MDL Drug Data Report 2006; Symyx Technologies, Inc.: Sunnyvale, CA, 2006.

(21) Kashima, H.; Tsuda, K.; Inokuchi, A. In *20th International Conference on Machine Learning*; Fawcett, T., Mishra, N., Eds.; Proceedings of the 20th International Conference on Machine Learning; AAAI Press: Menlo Park, CA, 2003; pp 321−328.

(22) Fröhlich, H.; Wegner, J. K.; Sieker, F.; Zell, A. Kernel functions for attributed molecular graphs: A new similarity-based approach to ADME prediction in classification and regression. *QSAR Comb. Sci.* **2006,** *25*, 317–326.

(23) Ralaivola, L.; Swamidass, S. J.; Saigo, H.; Baldi, P. Graph kernels for chemical informatics. *Neural Networks* **2005,** *18*, 1093–1110.

(24) Mahé, P.; Ralaivola, L.; Stoven, V.; Vert, J.-P. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model.* **2006,** *46*, 2003–2014.

(25) Bender, A.; Mussa, H. Y.; Gill, G. S.; Glen, R. C. Molecular surface point environments for virtual screening and the elucidation of binding patterns (MOLPRINT 3D). *J. Med. Chem.* **2004,** *47*, 6569–6583.

(26) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The chemistry development kit (CDK): An open-source

Java library for chemo- and bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003,** *43,* 493–500.

(27) Steinbeck, C.; Hoppe, C.; Kuhn, S.; Floris, M.; Guha, R.; Willighagen, E. Recent developments of the chemistry development kit (CDK): An open-source java library for chemo- and bioinformatics. *Curr. Pharm. Des.* **2006,** *12,* 2111–2120.

(28) Kirchmair, J.; Markt, P.; Distinto, S.; Wolber, G.; Langer, T. Evaluation of the performance of 3D virtual screening protocols: RMSD comparisons, enrichment assessments, and decoy selection. What can we learn from earlier mistakes? *J. Comput.-Aided Mol. Des.* **2008,** *22,* 213–228.

(29) Jain, A. N.; Nicholls, A. Recommendations for evaluation of computational methods. *J. Comput.-Aided Mol. Des.* **2008,** *22,* 133–139.

(30) Nicholls, A. What do we know and when do we know it? *J. Comput.-Aided Mol. Des.* **2008,** *22,* 239–255.

(31) Rarey, M.; Dixon, J. S. Feature trees: A new molecular similarity measure based on tree matching. *J. Comput.-Aided Mol. Des.* **1998,** *12,* 471–490.

(32) Clark, R. D.; Webster-Clark, D. J. Managing bias in ROC curves. *J. Comput.-Aided Mol. Des.* **2008,** *22,* 141–146.

(33) Swamidass, S. J.; Azencott, C.-A.; Daily, K.; Baldi, P. A CROC stronger than ROC: Measuring, visualizing and optimizing early retrieval. *Bioinformatics* **2010,** *26,* 1348–1356.

(34) Schölkopf, B.; Smola, A. *Learning with Kernels*; MIT Press: Cambridge, MA, 2002.

(35) Gower, J. C. A general coefficient of similarity and some of its properties. *Biometrics* **1971,** *27,* 857–871.

(36) Swamidass, S. J.; Baldi, P. Bounds and algorithms for fast exact searches of chemical fingerprints in linear and sublinear time. *J. Chem. Inf. Model.* **2007,** *47,* 302–317.

(37) Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J. C.; Sheridan, R. P.; Feuston, B. P. Random forest: A classification and regression tool for compound classification and QSAR modeling. *J. Chem. Inf. Comput. Sci.* **2003,** *43,* 1947–1958.

(38) Hsieh, C.-J.; Chang, K.-W.; Lin, C.-J.; Keerthi, S. S.; Sundararajan, S. *Proceedings of the 25th International Conference on Machine Learning*; ICML '08; ACM: New York, 2008; pp 408−415.

(39) Bottou, L.; Bousquet, O. *Mining Massive Data Sets for Security*; NATO ASI Workshop Series; IOS Press: Amsterdam, 2008.

(40) Sutherland, J. J.; O'Brien, L. A.; Weaver, D. F. Spline-fitting with a genetic algorithm: A method for developing classification structure−activity relationships. *J. Chem. Inf. Comput. Sci.* **2003,** *43,* 1906–1915.

(41) Oprea, T. I.; Davis, A. M.; Teague, S. J.; Leeson, P. D. Is there a difference between leads and drugs? A historical perspective. *J. Chem. Inf. Comput. Sci.* **2001,** *41,* 1308–1315.

(42) Stahl, M.; Rarey, M. Detailed analysis of scoring functions for virtual screening. *J. Med. Chem.* **2001,** *44,* 1035–1042.

(43) Chen, J. H.; Linstead, E.; Swamidass, S. J.; Wang, D.; Baldi, P. ChemDB update: Full-text search and virtual chemical space. *Bioinformatics* **2007,** *23,* 2348–2351.

(44) Gasteiger, J.; Rudolph, C.; Sadowski, J. Automatic generation of 3D-atomic coordinates for organic molecules. *Tetrahedron Comput. Methodol.* **1992,** *3,* 537–547.

(45) *Schrödinger MacroModel,* version 9.6; Schrödinger LLC: New York, 2008.

(46) Hert, J.; Willett, P.; Wilton, D. J.; Acklin, P.; Azzaoui, K.; Jacoby, E.; Schuffenhauer, A. Comparison of topological descriptors for similarity-based virtual screening using multiple bioactive reference structures. *Org. Biomol. Chem.* **2004,** *2,* 3256–3266.

(47) Bouckaert, R. R.; Frank, E. In *Advances in Knowledge Discovery and Data Mining, Proceedings of 8th Pacific−Asia Conference, PAKDD 2004*; Dai, H., Srikant, R., Zhang, C., Eds.; Springer: Heidelberg, 2004; Vol. 3056; pp 3−12.

(48) Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. The WEKA data mining software: An update. *SIGKDD Explorations* **2009,** *11,* 10–18.

(49) Waldvogel, B. LIBLINEAR Java Port, 2010. http://github.com/bwaldvogel/liblinear-java/commits/master (accessed August 25, 2010).

(50) *dragonX,* version 1.2; Talete srl: Milano, Italy, 2008.

(51) Irwin, J. J.; Shoichet, B. K. ZINC-A free database of commercially available compounds for virtual screening. *J. Chem. Inf. Model.* **2005,** *45,* 177–182.