Classifying 'Drug-likeness' with Kernel-Based Learning Methods

Klaus-Robert Müller,[†,‡] Gunnar Rätsch,[§] Sören Sonnenburg,*[,†] Sebastian Mika,[||] Michael Grimm,[⊥] and Nikolaus Heinrich[⊥]

Fraunhofer FIRST, Kekuléstrasse 7, 12489 Berlin, Germany, Computer Science, University of Potsdam, August-Bebel-Strasse 89, 14482 Potsdam, Germany, Friedrich Miescher Laboratory of the Max Planck Society, Spemannstrasse 39, 72076 Tübingen, Germany, idalab GmbH, Mohrenstrasse 63, 10117 Berlin, Germany, and Computational Chemistry, Schering AG, Müllerstrasse 178, 13342 Berlin, Germany

Received August 24, 2004

In this article we report about a successful application of modern machine learning technology, namely Support Vector Machines, to the problem of assessing the 'drug-likeness' of a chemical from a given set of descriptors of the substance. We were able to drastically improve the recent result by Byvatov et al. (2003) on this task and achieved an error rate of about 7% on unseen compounds using Support Vector Machines. We see a very high potential of such machine learning techniques for a variety of computational chemistry problems that occur in the drug discovery and drug design process.

1. INTRODUCTION

Neural networks are a common working tool for problem solving in computational chemistry (e.g. refs 1−3) and other industries (e.g. ref 4). Recent advances in machine learning, so-called kernel-based learning methods, like Support Vector Machines[5−8] (SVM), or the more general mathematical programming machines[9,10] have already demonstrated their superiority over neural network approaches for a wide range of applications outside chemistry (cf. refs 7, 11, and 8 and references therein).

Recent work by Byvatov et al.[12] considered a large benchmark data set containing atomic descriptors of compounds that are either drugs (WDI) or nondrugs (ACD). Byvatov et al. proposed a Support Vector Machine (SVM) based method that correctly classifies 80% (error rate 20%) of the compounds based on Ghose-Crippen (GC) descriptors (counting occurrences of atoms according to the 120 types defined in ref 13). In their comparison they find that SVMs are only slightly (and from a statistical point this seems not significantly) better than a rather simple artificial neural network (20.75% error rate).

Independently of Byvatov et al.[12] and simultaneously we analyzed the benchmark data using not only SVMs but also other modern classifiers. Our results are significantly better than those reported in ref 12. We show that (a) using the same amount of training data and after appropriate preprocessing the error rate can be reduced to 18.1% and (b) using more training examples we achieve a surprisingly low error rate of 10.2%. This shows that careful model selection combined with novel large-scale SVM algorithms can indeed help in making better predictions.

Byvatov et al. considered the data set that has also been used in ref 14. However in the experiments by Byvatov et al., the WDI to ACD ratio has been skewed from the original 1:4.4 to almost 1:1. Since it is to be expected that the number of potential compounds in the ACD is much larger than the ones in the WDI, it seems more reasonable to stay with the given empirical priors.

We therefore generated a new data set obeying the empirical class priors and compared different methods on this new data set. For this more realistic data set we show in simulations that the best method achieves error rates at around 7% which we additionally confirmed in a blind-test experiment. We therefore were able to reduce the error rate for the task of classifying drugs (WDI) vs nondrugs (ACD) by more than 60% compared to Byvatov et al.[12] and Sadowski and Kubinyi.[14]

2. METHODS

Machine learning algorithms such as neural networks or SVMs use the statistical differences in features to infer the implicit properties of the classification problem. As most existing algorithms differ in their implicit assumptions, and since there is no single best algorithm for all problems, it is important to test a variety of different techniques on a new problem.

2.1. Model Selection. Typically, machine learning algorithms possess one or two different hyperparameters (e.g. kernel width, number of neighbors, etc.) that need to be selected carefully in order to yield a good generalization performance, i.e., a low error rate on unseen data. This so-called model selection process is an art and mandatory for achieving excellent results. Typically cross-validation techniques are applied here, where an appropriate range of the hyperparameters is scanned (e.g. ref 7). A proper model selection is the key to a successful application of modern learning techniques; however, at the same time it is typically computationally very expensive.

* Corresponding author e-mail: soeren.sonnenburg@first.fraunhofer.de.
[†] Fraunhofer FIRST.
[‡] University of Potsdam.
[§] Friedrich Miescher Laboratory of the Max Planck Society.
[||] idalab GmbH.
[⊥] Schering AG.

10.1021/ci049737o CCC: $30.25
Published on Web 02/08/2005

For the first part of the experiments (Section 4.1) we replicated the model selection procedure used in ref 12 as closely as possible in order to guarantee a fair comparison. We first split the data set into a training and a test set. Then we used 4-fold cross-validation to estimate the generalization error and selected the model with the lowest validation error. Finally we measured the accuracy of the selected method on the test set. For the second part (Section 4.2) we opted for a simpler and less computational demanding model selection scheme and split the data set into three parts: training, validation and test set. The validation set is used for parameter tuning and the test set for the final evaluation.

2.2. Learning Techniques. We investigated methods that range from conventional decision trees,[15] Fishers Linear Discriminant[16] and Nearest Neighbor methods[17] to advanced learning techniques such as SVMs[5,7,8] and linear programming machines.[18,19] Contrary to Byvatov et al.[12] we solely base our study on a Ghose-Crippen parametrization of the chemicals.[13] However, before feeding it into any of the learning machines we preprocess the GC descriptors by adding one and then computing the log. This transformation helped to increase the numerical stability of large scale SVM optimization.

In the sequel we outline the different learning techniques that we used in the comparison:

Support Vector Machines work by mapping the training data into a feature space by the aid of a so-called kernel function and then separating the data using a *large margin hyperplane*. Intuitively, the kernel computes a similarity between two given examples. Most commonly used kernel functions are *RBF kernels*

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{||\mathbf{x} - \mathbf{x}'||^2}{\sigma^2}\right)$$

and *polynomial kernels*

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}')^d$$

where one often employs normalization for the latter:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x})k(\mathbf{x}', \mathbf{x}')}}$$

In our experiments we use SVMs with RBF kernels and normalized polynomial kernels. More details on SVMs have been provided in the literature numerous times, for instance in refs 20, 5, 7, and 8.

Linear Programming Machines are similar to SVMs but do not use kernels and compute a hyperplane with large margin using a linear (1-norm) distance metric.[18,19] To allow nonlinear classifications, one may nonlinearly map the examples into another feature space. In our experiments we used quadratic features (all first- and second-order monomials of the 120 input features). Solving the resulting optimization problems can be done very efficiently.

Linear Discriminant Analysis computes a hyperplane in the input space that minimizes the within-class variance and maximizes the between class distance.[16] It can be efficiently computed in the linear case even with large data sets. Nonlinear extensions by using kernels exist,[10] however, make it difficult to apply it to problems with large training sets. We therefore only used the linear version.

Bagged K−Nearest Neighbor Classifiers. We use the standard *K*-Nearest Neighbor algorithm as described in ref 17 together with Bagging:[21] we randomly draw *N* subsets (with replacement) of the training set and then apply the *K*-Nearest Neighbor algorithm to it. The *N* different predictions are averaged. The use of Bagging has the advantage that it usually improves the performance and leads to real-valued outputs. *N* was chosen to be 30 throughout our experiments.

Bagged Decision Trees. We use the C4.5 decision tree implementation[15] with Bagging as above.

3. SETUP

We prepare three different data sets based upon the WDI[22] and ACD[23] databases used in ref 14:

(1) From the WDI and ACD data set used in ref 14, Ghose Crippen (GC) parameters have been extracted (J. Sadowski kindly provided the complete GC descriptor matrices computed for all 207,001 compounds using the 1998 versions of WDI and ACD.), and 10% of the examples have been kept separated for blind testing (cf. Section 4.3). The remaining 90% (186,301 examples) were split into training (166,301 examples), validation (10,000 examples) and test (10,000 examples) sets.

(2) A subset of 9208 molecules of data set (1), 4998 of them drugs and 4210 nondrugs, are used as data set (2) (as in ref 12). As Byvatov et al. did not make this smaller set publicly available, 9208 examples are chosen randomly from data set (1) which follow the same prior as in ref 12, i.e., the WDI to ACD ratio becomes almost 1:1 instead of 1:4.4. We perform 4-fold cross-validation and thus in each fold use 5250 of the examples for training and 1750 for validation, while the test error was measured on 2208 samples which were kept separate.

Data set (2) and a version of data set (1) following the same WDI to ACD ratio as (2) were used for direct comparison with ref 12 (Section 4.1). We use data set (1) for our performance evaluation of different methods (Section 4.2) and the final blind test evaluation (Section 4.3).

4. RESULTS

In the following sections we present our experimental results on the aforementioned data sets. In particular we will show in Section 4.1 that the results presented in ref 12 were not optimal and that careful model selection and preprocessing leads to improvements. In Section 4.2 we show new results on a larger study avoiding the problems present in ref 12. Finally, in Section 4.3 we present results on a so-called blind-test illustrating the excellent expected performance of our system in the real-world.

4.1. Comparison with Prior Results. First we present our results on the smaller data set (2) as described before − a *randomly* chosen reduced data set with 9208 chemicals described by 120 Ghose Crippen features (cf. Table 1). As motivated before this data set resembles very closely the one used in refs 14 and 12, and we compare our results to these findings, especially the latter.

Byvatov et al.[12] performed model selection for SVMs over the regularization constant *C* and a scaling factor *s* in the

Table 1. An Overview of the Two Data Sets Used

data set	description
1	data set derived from ACD/WDI with 207,001 compounds; only 90%, i.e., 151,752 ACDs and 34,549 WDIs (186,301) were available
2	9208 randomly chosen examples following the same prior as in (1); for comparison with ref 12

Table 2. Estimate of Expected Test Errors on Classifying WDI against ACD Compounds for SVMs Only Using Ghose-Crippen Parameters[a]

method	training examples	accuracy (%)
ANN−BYV (N.P.)	5250	20.8 ± 0.7
SVM-BYV (N.P.)	5250	20.0 ± 0.1
SVM-POLY (N.P.)	5250	20.1 ± 0.5
SVM-RBF (N.P.)	5250	20.0 ± 0.2
SVM-POLY	5250	18.7 ± 0.5
SVM-RBF	5250	18.1 ± 0.6
SVM-POLY large	166,301	10.8
SVM-RBF large	166,301	10.2

[a] The SVMs were trained on 5250 examples (Byvatov et al.) and 166,301 examples (SVM large) and tested on the 2208 remaining examples (Byvatov et al.) and 3454 examples (SVM large), respectively. In the upper part of the table shows results obtained on non preprocessed (N.P.) data for comparison. The ANN−BYV and SVM-BYV results were taken from ref 12.

inhomogeneous polynomial kernel of fixed degree 5, i.e. $k(\mathbf{x}, \mathbf{y}) = ((\mathbf{x}\cdot\mathbf{y})s + 1)^5$. We used homogeneous normalized polynomial and rbf kernels and tuned the degree d of the kernel (rbf-kernel width σ^2 respectively) *and* the regularization constant C of the SVM as outlined in Section 2.2. Our SVMs yields an error rate of 18.7% (RBF−SVM: 18.1%), performing slightly better than the 20.0% presented in ref 12 − a relative reduction of the error rate of 6% (RBF−SVM: 9%). It also turned out that a higher polynomial degree of $d = 6$ was optimal, contrary to the fixed value of $d = 5$.

Table 2 shows the averages and standard deviations of the test errors of the algorithms trained on the four different sets generated in cross-validation (each exluding one-quarter of the data, leading to 5250 examples). Based on the cross-validation errors (not shown) we tuned the model parameters. Usually one retrains the algorithm on all available training data (i.e. 7000). However, this has not been done in ref 12, and we chose to follow the same strategy. Note however, that if we retrain for instance the RBF SVM on all 7000 available training examples, it leads to a further improvement (17.3%).

For comparison we also used the same data set without the log-transformation (as in ref 12). In this case the resulting error rates were almost identical with the SVM result by Byvatov et al. (cf. Table 2). We originally introduced the log-transformation to increase the numerical stability of the SVM optimization (Note that atom counts can be quite big and raising it to the power of for instance five can lead to quite large numbers.). However, it turned out that the transformation actually leads to a considerable improvement. Moreover, while the optimal degree for the polynomial kernel with log-transformation is $d = 6$, it is only $d = 2$ for the unprocessed data. The log-transformation seems to represent the data at more appropriate scales and allows better generalization.

Moreover, in ref 12 it was shown that the error rate "leveled off" between 2000 and 3000 examples. Since the number of available data is much larger than just 9,208 samples, we repeated the model selection with a much larger training set (1) (cf. Table 1). To keep results comparable we also skewed the class priors in the validation and testing sets to be roughly equal instead of being close to the original ratio and estimated the SVMs bias term on this validation set (leading to fewer validation and test examples − 3454 each, instead of 10,000). Using more of the abundantly available training data, one obtains considerable lower error rates. Using 166,301 training examples we achieved an error of 10.8% for the SVM using a polynomial kernel and for the RBF−SVM 10.6% (cf. Table 2) − a relative improvement of 46% (RBF−SVM: 47%). This finding clearly contradicts the interpretation of a plateau in ref 12.

Table 3. Estimate of Expected Test Errors on WDI against ACD Data for Various Machines Using Only Ghose-Crippen Parameters[a]

method	total error (%)	AUC (%)	WDI found at 83% ACD found (%)
Sadowski and Kubinyi (1998)	18.1	n/a	77
LPM linear	13.7	86.2	76.9
Fishers Discrim.	13.6	86.9	77.6
LPM quadratic	9.3	91.9	86.0
KNN	8.2	93.4	90.9
C4.5	8.2	94.7	92.0
SVM-POLY	7.0	95.0	91.4
SVM-RBF	6.8	95.2	91.7

[a] Machines were trained on 166,301 examples from data set (1). Parameter tuning was performed on 10,000 examples, and it was tested on the remaining 10,000 examples. For comparison to Sadowksi and Kubinyi (1998) also the true positive rate of WDI at a false negative rate of 17% for ACD (i.e. 83% ACD are found) is shown. The AUC denotes the area under the ROC Curve.[25]

4.2. Evaluation. As we have already seen in the previous Section, the rate of 20% error as presented in ref 12 can significantly be improved upon by performing a more careful model selection.

Note furthermore that it seems unclear why it should make sense for a practical application to change the class priors to be roughly equal. Since it is to be expected that the number of potential compounds in the ACD is much larger than the ones in the WDI it seems more reasonable to stay with the given empirical priors. The results we present in the following are hence all based on the data set (1) described before (cf. Table 1), i.e., the data for training, validation and testing have the original class prior of 1:4.4.

Yet, the 120 GC-parameters do not sufficiently describe the compounds as for the 34,549 drugs there are only 32,932 unique descriptors. Similarly for the 151,752 nondrugs there are only 131,464 unique descriptors. Even worse, there are also 824 descriptors which code for both drug *and* nondrug resulting in an WDI/ACD overlap of 2368 compounds. However intuitively, training with "inconsistent" data can be understood as some extra regularization by noise.[24]

The results on data set (1) reported in Table 3 vary considerably, with error rates of 13.6% for the Linear Programming Machine, which is our worst result, but is still a relative improvement of 24% over the result reported in ref 14 and of 31% over the result from ref 12 (note, however, that the class priors are different in ref 12). Our best result is 6.8% error from a RBF−SVM of kernel width $\sigma^2 = 5$, a relative improvement of 62% and 66% compared to the errors reported in refs 14 and 12, respectively.

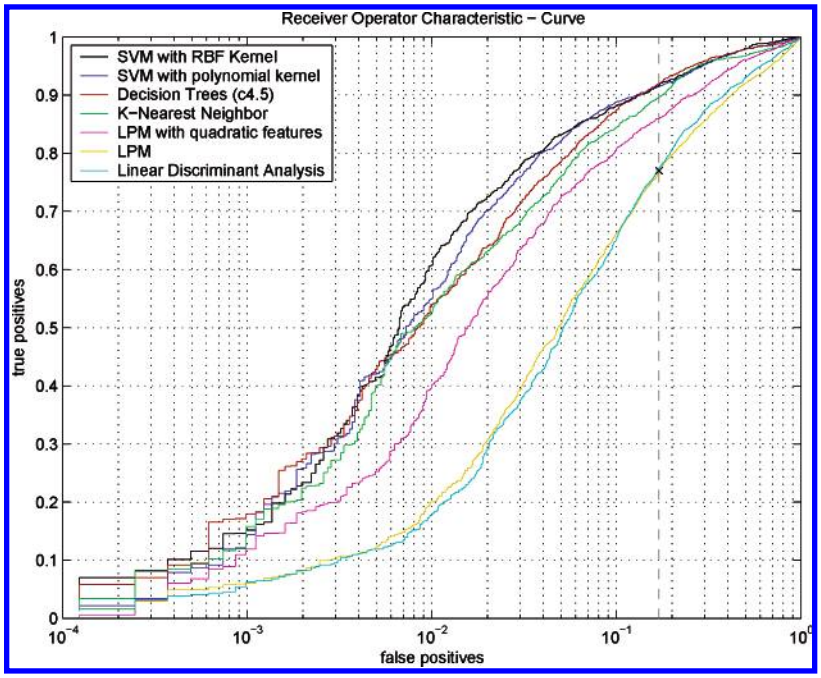

Figure 1. Receiver-operator-curves for the tested methods including the result of Sadowski and Kubinyi (1998) (marked by 'x').

It seems noteworthy that the well-established techniques such as C4.5 decision trees or even k-nearest neighbor achieved very good results. We noticed quite large improvements for these techniques when used with Bagging.

Since for practical applications such as e.g. high throughput screening the minimal possible error rate is much less important than the ratio of false positive predictions (i.e. nondrug like compounds that need to be tested in vain) to true positives (i.e. compounds that are at least potential candidates) we show in Figure 1, the Receiver Operating Characteristic (ROC) Curve for each of the classifiers. To obtain such a curve one changes an internal parameter (the decision threshold) and measures the number of false positives and true positives (on the test set). In ref 14 a true positive rate of 77% is reported at a false positive rate of 83% (marked by a 'x' in Figure 1). Support Vector Machines with polynomial and RBF kernel obtain true positive rates of 91.4% and 91.7% at the same false positive rate, respectively. This is a relative improvement of the false negative rate of 63% (RBF−SVM: 64%) (cf. Table 3 for the other methods).

An interesting observation is that $d = 11$ is the optimal degree for polynomial kernels. This finding is an indication for a highly fragmented classification space that requires such a highly complex learning machine and also a large number of examples for learning.

4.3. Blind Test. After careful model selection and the comparison of the methods one can be confident that the chosen learning machines also perform well on unseen data. However to provide a fully realistic practice test, 10% of the data points were retained by the Schering group as described before in Section 3 and thus were from the beginning unavailable for the training and model selection procedure done by Fraunhofer's IDA group. After finishing the training and model selection procedure this truly unseen data was supplied *without a label* to the Fraunhofer IDA group, i.e., in this manner a blind-test was executed. Using the best (most trusted in) models for data set (1), i.e., a polynomial SVM using a polynomial kernel of degree 11 and the RBF−SVM with width $\sigma^2 = 5$, IDA computed the labels and sent the results to the Schering group, where the correct labels were matched with the inferred ones. The blind test result yielded an error of 7.1% for the polynomial SVM and 6.9% for the RBF−SVM in good agreement with our predictions of the generalization error.

5. CONCLUSION

We would like to conclude by stating that machine learning technology where a careful model selection procedure is used can improve dramatically upon existing results and ultimately provide more powerful software tools for computational chemistry. Note furthermore that the current results were achieved under a realistic blind test scenario, unlike most existing studies.

It is interesting to observe that among the best classifiers − an SVM of polynomial degree $d = 11$ − for the drug-likeness analysis allows for a very complex decision surface. This indicates that the Ghose-Crippen features make the problem rather local and fine granular. We conjecture that more powerful chemical descriptors could ultimately allow classification with simpler global models that would at the same time be easier to interpret. Future research will follow this idea.

ACKNOWLEDGMENT

This work was partly funded by the PASCAL Network of Excellence project (EU #506778). The authors thank H. Briem for valuable discussions and especially J. Sadowski for supplying the original data set making a direct comparison with the other authors work possible. A large part of this work has been done while G.R. was at Fraunhofer FIRST in Berlin and at the Max Planck Institute for Biological Cybernetics in Tübingen.

REFERENCES AND NOTES

(1) Zupan, J.; Gasteiger, J. *Neural Networks in Chemistry and Drug Design*; Wiley: 1999.
(2) Peterson, K. *Rev. Comput. Chem.* **2000**, *16*, 53−140.

(3) Devillers, J. *Principles QSAR Drug Design* **1996**, *2*.
(4) Orr, G., Müller, K.-R., Eds.; *Neural Networks: Tricks of the Trade*; Springer LNCS: 1998; Vol. 1524.
(5) Vapnik, V. *The nature of statistical learning theory;* Springer-Verlag: New York, 1995.
(6) Schölkopf, B.; Smola, A.; Müller, K.-R. *Neural Computation* **1998**, *10*, 1299−1319.
(7) Müller, K.-R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B. *IEEE Trans. Neural Networks* **2001**, *12*, 181−201.
(8) Schölkopf, B.; Smola, A. *Learning with Kernels*; MIT Press: Cambridge, MA, 2002.
(9) Mika, S.; Rätsch, G.; Müller, K.-R. A mathematical programming approach to the kernel Fisher algorithm. In *Advances in Neural Information Processing Systems*; Leen, T., Dietterich, T., Tresp, V., Eds.; MIT Press: 2001; Vol. 13.
(10) Mika, S.; Rätsch, G.; Weston, J.; Schölkopf, B.; Smola, A.; Müller, K.-R. *IEEE Trans. Patterns Analysis Machine Intelligence* **2003**, *25*, 623−627.
(11) Zien, A.; Rätsch, G.; Mika, S.; Schölkopf, B.; Lengauer, T.; Müller, K.-R. *BioInformatics* **2000**, *16*, 799−807.
(12) Byvatov, E.; Fechner, U.; Sadowski, J.; Schneider, G. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1882−1889.
(13) Ghose, A.; Crippen, G. *J. Comput. Chem.* **1986**, *7*, 565−577.
(14) Sadowski, J.; Kubinyi, H. *J. Med. Chem.* **1998**, *41*, 3325−3329.
(15) Quinlan, J. *C4.5: Programs for Machine Learning*; Morgan Kaufmann: 1992.
(16) Fisher, R. *Annals Eugenics* **1936**, *7*, 179−188.
(17) Cover, T.; Hart, P. *IEEE Trans. Inf. Theory* **1967**, *13*, 21−27.
(18) Bennett, K.; Mangasarian, O. *Optimization Methods Software* **1992**, *1*, 23−34.
(19) Graepel, T.; Herbrich, R.; Schölkopf, B.; Smola, A.; Bartlett, P.; Müller, K.-R.; Obermayer, K.; Williamson, R. Classification on Proximity Data with LP-Machines. In *Proceedings of ICANN'99*; Willshaw, D., Murray, A., Eds.; IEE Press: 1999; Vol. 1, 304−309.
(20) Boser, B.; Guyon, I.; Vapnik, V. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*; Haussler, D., Ed.; 1992.
(21) Breiman, L. *Machine Learning* **1996**, *26*, 123−140.
(22) "World drug index WDI", 1996 version 2/96.
(23) "Available Chemicals Directory ACD", 1996 Version 2/96.
(24) Bishop, C. *Neural Comput.* **1995**, *7*, 108−116.
(25) Metz, C. *Seminars Nuclear Medicine* **1978**, *VIII*.

CI049737O