Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification

Evgeny Byvatov,[†] Uli Fechner,[†] Jens Sadowski,[‡] and Gisbert Schneider*,[†]

Institut für Organische Chemie und Chemische Biologie, Johann Wolfgang Goethe-Universität, Marie-Curie-Strasse 11, D-60439 Frankfurt, Germany, and AstraZeneca R&D Mölndal, SC 264, S-431 83 Mölndal, Sweden

Received June 13, 2003

Support vector machine (SVM) and artificial neural network (ANN) systems were applied to a drug/nondrug classification problem as an example of binary decision problems in early-phase virtual compound filtering and screening. The results indicate that solutions obtained by SVM training seem to be more robust with a smaller standard error compared to ANN training. Generally, the SVM classifier yielded slightly higher prediction accuracy than ANN, irrespective of the type of descriptors used for molecule encoding, the size of the training data sets, and the algorithm employed for neural network training. The performance was compared using various different descriptor sets and descriptor combinations based on the 120 standard Ghose-Crippen fragment descriptors, a wide range of 180 different properties and physicochemical descriptors from the Molecular Operating Environment (MOE) package, and 225 topological pharmacophore (CATS) descriptors. For the complete set of 525 descriptors cross-validated classification by SVM yielded 82% correct predictions (Matthews $cc = 0.63$), whereas ANN reached 80% correct predictions (Matthews $cc = 0.58$). Although SVM outperformed the ANN classifiers with regard to overall prediction accuracy, both methods were shown to complement each other, as the sets of true positives, false positives (overprediction), true negatives, and false negatives (underprediction) produced by the two classifiers were not identical. The theory of SVM and ANN training is briefly reviewed.

INTRODUCTION

Early-phase virtual screening and compound library design often employs filtering routines which are based on binary classifiers and are meant to eliminate potentially unwanted molecules from a compound library.[1,2] Currently two classifier systems are most often used in these applications: PLS-based classifiers[3,4] and various types of artificial neural networks (ANN).[5−9] Typically, these systems yield an average overall accuracy of 80% correct predictions for binary decision tasks following the "likeness concept" in virtual screening.[2,10] The support vector machine (SVM) approach was first introduced by Vapnik as a potential alternative to conventional artificial neural networks.[11,12] Its popularity has grown ever since in various areas of research, and first applications in molecular informatics and pharmaceutical research have been described.[13−15] Although SVM can be applied to multiclass separation problems, its original implementation solves binary class/nonclass separation problems. Here we describe application of SVM to the drug/nondrug classification problem, which employs a class/nonclass implementation of SVM. Both SVM and ANN algorithms can be formulated in terms of learning machines. The standard scenario for classifier development consists of two stages: training and testing. During first stage the learning machine is presented with labeled samples, which are basically *n*-dimensional vectors with a class membership label attached. The learning machine generates a classifier for prediction of the class label of the input coordinates. During the second stage, the generalization ability of the model is tested.

Currently various sets of molecular descriptors are available.[16] For application to drug/nondrug classification of compounds, the molecules are typically represented by *n*-dimensional vectors.[6,7] In this work, we focused on the fragment-based Ghose-Crippen (GC) descriptors[17−19] which were used in the original work of Sadowski and Kubinyi for drug/nondrug classification,[7] descriptors provided by the MOE software package (Molecular Operating Environment. Chemical Computing Group Inc., Montreal, Canada), and CATS topological pharmacophores.[20] Having defined this molecular representation, the task of the present study was to compare the classification ability of standard SVM and feed-forward ANN on the drug/nondrug data. A www-based interface for calculating the drug-likeness score of a molecule using our SVM solution based on the CATS descriptor was developed and can be found at URL: http://gecco.org.chemie.uni-frankfurt.de/gecco.html.

DATA AND METHODS

Data Sets. For SVM and ANN training we used the sets of "drug" and "nondrug" molecules prepared by Kubinyi and Sadowski.[7] From the original data set 9208 molecules could be processed by our descriptor generation software. The final working set contained 4998 drugs and 4210 nondrug molecules. Three sets of descriptors were calculated: counts of the standard 120 Ghose Crippen descriptors,[17−19] 180

* Corresponding author phone: +49-69 79829821; fax: +49-69 7982-9826; e-mail: gisbert.schneider@modlab.de.
[†] Johann Wolfgang Goethe-Universität.
[‡] AstraZeneca R&D Mölndal.

10.1021/ci0341161 CCC: $25.00
Published on Web 09/27/2003

descriptors from MOE (Molecular Operating Environment. Chemical Computing Group Inc., Montreal, Canada), and 225 topological pharmacophore (CATS) descriptors.[20] MOE descriptors include various 2D and 3D descriptors such as volume and shape desciptors, atom and bonds counts, Kier−Hall connectivity and kappa shape indices, adjacency and distance matrix descriptors, pharmacophore feature descriptors, partial charges, potential energy descriptors, and conformation-dependent charge descriptors. Before calculating MOE descriptors, single 3D conformers were generated by CORINA.[21] 225 CATS descriptors were calculated using our own software taking into consideration pairs of atom types separated by up to 15 bonds (URL: http://gecco.org.chemie.uni-frankfurt.de/gecco.html).[20] All 225 descriptor columns were individually autoscaled. An alternative would have been block-scaling where each descriptor class is autoscaled as a whole, which was not applied here.

Support Vector Machine. SVM classifiers are generated by a two-step procedure: First, the sample data vectors are mapped ("projected") to a very high-dimensional space. The dimension of this space is significantly larger than dimension of the original data space. Then, the algorithm finds a hyperplane in this space with the largest margin separating classes of data. It was shown that classification accuracy usually depends only weakly on the specific projection, provided that the target space is sufficiently high dimensional.[11] Sometimes it is not possible to find the separating hyperplane even in a very high-dimensional space. In this case a tradeoff is introduced between the size of the separating margin and penalties for every vector which is within the margin.[11] The basic theory of SVM will be briefly reviewed in the following.

The separating hyperplane is defined as

$$D(\mathbf{x}) = (\mathbf{w}\cdot\mathbf{x}) + w_0$$

Here $\boldsymbol{x}$ is a samples vector mapped to a high dimensional space, and $\boldsymbol{w}$ and w_0 are parameters of the hyperplane that SVM will estimate. Then the margin can be expressed as a minimal τ for which holds

$$\frac{y_k D(\mathbf{x_k})}{||\mathrm{w}||} \geq \tau$$

Without loss of generality we can apply a constraint $\tau||w|| = 1$ to $\boldsymbol{w}$. In this case maximizing τ is equivalent to minimizing $||w||$ and SVM training is becoming the problem of finding the minimum of a function with the following constraints:

$$\textit{minimize} \quad \eta(w) = \frac{1}{2}(w\cdot w)$$

$$\textit{subject to constraints} \quad y_i[(w\cdot x_i) + w_0] \geq 1$$

This problem is solved by introduction of Lagrange multipliers and minimization of the function

$$Q(\mathbf{w},w_0,\alpha) = \frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) - \sum_{i=1}^{n}\alpha_i\{y_i[(\mathbf{w}\cdot\mathbf{x_i}) + w_0] - 1\}$$

Here α_i are Lagrange multipliers. Differentiating over $\boldsymbol{w}$ and w_i and substituting we obtain

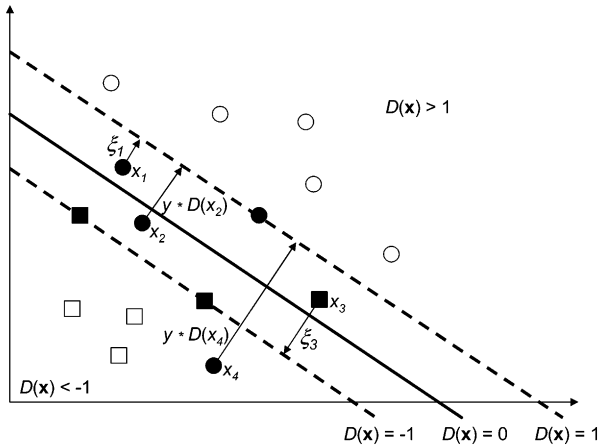

Figure 1. Principle of SVM classification. The task was to separate two classes of objects indicated by squares and circles. Squares represent nonclass samples ("negative examples", e.g. nondrugs) and circles are class members ("positive examples", e.g. drugs). $D(\mathbf{x})$ is the decision function defining class membership according to the SVM classifier which is represented by the separating line ($D(\mathbf{x}) = 0$). The margin is indicated by dotted lines. Support vectors are indicated by filled objects (x_2, x_2, x_3, x_4). ξ_i are slack variables for support vectors that are not lying on the margin border. y_i are label-variables equal to 1 for positive examples (class membership) and −1 for negative examples (nonclass membership). See text for details.

$$\max \quad Q(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j)$$

$$\textit{subject to constraints} \quad \sum_{i=1}^{n} y_i\alpha_i = 0;\ \alpha_i \geq 0, i = 1,\ldots,n$$

When perfect separation is not possible slack variables are introduced for sample vectors which are within the margin, and the optimization problem can be reformulated:

$$\textit{minimize } \eta(w) = \frac{1}{2}(w\cdot w) + C\sum_i \xi_i$$

$$\textit{subject to constraints } y_i[(w\cdot x_i) + w_0] \geq 1 - \xi_i$$

Here ξ_i are slack variables. These variables are not equal to zero only for those vectors which are within the margin. Introducing Lagrange multipliers again we finally obtain

$$\max \quad Q(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j)$$

$$\textit{subject to constraints} \quad \sum_{i=1}^{n} y_i\alpha_i = 0,\ C \geq \alpha_i \geq 0, i = 1,\ldots,n$$

This is a quadratic programming (QP) problem for which several efficient standard methods are known.[22] Due to the very high dimensionality of the QP problem, which typically arises during SVM training, an extension of the algorithm for solving QP is used in SVM applications.[23]

A geometrical illustration of the meaning of slack variables and Lagrange multipliers is given in Figure 1. Points classified by SVM can be divided into two groups, support vectors and nonsupport vectors. Nonsupport vectors are classified correctly by the hyperplane and are located outside

the separating margin. Slack variables and Lagrange multipliers for them are equal to zero. Parameters of the hyperplane do not depend on them, and even if their position is changed the separating hyperplane and margin will remain unchanged, provided that these points will stay outside the margin. Other points are support vectors, and they are the points which determine the exact position of the hyperplane. For all support vectors the absolute values of the slack variables are equal to the distances from these points to the edge of the separating margin. These distances are defined in the units of half of the width of the separating margin. For correctly classified points within the separating margin, slack variable values are between zero and one. For misclassified points within the margin the values of the slack variables are between one and two. For other misclassified points they are greater than two.

For points that are lying on the edge of margin, Lagrange multipliers are between zero and *C*, and slack variables for these points are still equal to zero. For all other points, for which the values of slack variables are larger than zero, Lagrange multipliers assume the value of *C*.

Explicit mapping to a very high-dimensional space is not required if calculation of the scalar product in this high dimensional space of every two vectors is feasible. This scalar product can be defined by introducing a kernel function$(\mathbf{x}\cdot\mathbf{x}') = K(\mathbf{x},\mathbf{x}')$,[24] where $\mathbf{x}$ and $\mathbf{x}'$ are vectors in a low-dimensional space for which a kernel function that corresponds to a scalar product in a high dimensional space is defined. Various kernels may be applied.[25] In our case, we used a kernel function of a fifth-order polynomial:

$$K(\mathbf{x},\mathbf{x}') = ((\mathbf{x}\cdot\mathbf{x}')s + r)^5$$

This kernel corresponds to the decision function

$$f(x) = sign(\sum_i \alpha_i K(x_i^{sv}, x) + b)$$

where α_i are Lagrange multipliers determined during training of SVM. The sum is only over support vectors x^{sv}. Lagrange multipliers for all other points are equal to zero. Parameter *b* determines the shift of the hyperplane, and it is also found during SVM training. Simultaneous scaling of *s*, *r*, and *b* parameters does not change the decision function. Thus, we can simplify the kernel by setting *r* equal to one:

$$K(x,x') = ((x\cdot x')s + 1)^5$$

In this case only the kernel parameter *s* and error tradeoff *C* must be tuned. Parameter *C* is not present explicitly in this equation; it is set up as a penalty for the misclassification error before the training of SVM is performed. For tuning parameters *s* and *C*, four-times cross-validation of training data was applied, and values for *s* and *C* that maximize accuracy were then chosen. Accuracy maximization was performed by heuristics based gradient descent.[26] Basically, the following procedure was applied. The data set was divided into two parts, training and validation set. The validation subset was put aside and used only for estimation of the performance of the trained classifier. Training data were divided into four nonoverlapping subsets. The SVM parameters to be determined were set to reasonable initial values. Then, the SVM was trained on the training data excluding one of the four subsets, and the performance of the obtained SVM classifier was estimated with the excluded subset. This procedure was repeated for each subset, and an average performance of the SVM classifier was obtained.

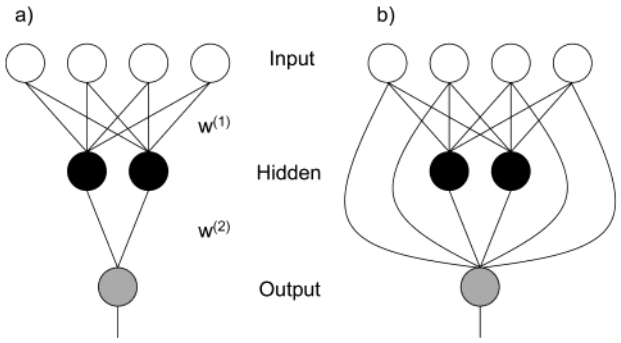

Figure 2. Architecture of artificial neural networks. Formal neurons are drawn as circles, weights are represented by lines connecting the neuron layers. Fan-out neurons are drawn in white, sigmoidal units in black, and linear units in gray. (a) conventional three-layered feed-forward system ("architecture I"); (b) network architecture used by Ajay and co-workers for drug-likeness prediction ("architecture II").[6]

For SVM training we used freely available SVM software (SVM-*Light* package; URL: http://svmlight.joachims.org/).[26,27] A Linux-based LSF (Load Sharing Facility; Platform Computing GmbH, D-40878 Ratingen, Germany) cluster was used for determination of the cross-validation error to reduce calculation time. All calculations were performed using the MATLAB package (MATLAB 2002, The mathematical laboratory. The MathWorks GmbH, D-52064 Aachen, Germany).

ARTIFICIAL NEURAL NETWORK

Conventional two-layered neural networks with a single output neuron were used for ANN model development (Figure 2a).[26] As a result of network training a decision function is chosen from the family of functions represented by the network architecture. This function family is defined by the complexity of the neural network: number of hidden layers, number of neurons in these layers, and topology of the network. The decision function is determined by choosing appropriate weights for the neural network. Optimal weights usually minimize an error function for the particular network architecture. The error function describes the deviation of predicted target values from observed or desired values. For our class/nonclass classification problem the target values were 1 for class (drugs) and -1 for nonclass (nondrugs). Standard two-layered neural network with a single output neuron can be represented by the following equation

$$y = \tilde{g}(\sum_{j=1}^{M} w_{1j}^{(2)} \cdot g(\sum_{i=1}^{d} w_{ji}^{(1)} \cdot x_i + w_{j0}^{(1)}) + w_{11}^{(2)})$$

with the error function $E = \sum_{k=1}^{n}(y(x_k) - y_k)^2$. In this work, $\tilde{g}$ is a linear function and *g* is a tan-sigmoid transfer function.

A second type network architecture containing additional connections from the input layer to the output layer was trained to reimplement the original drug/nondrug ANN developed by Ajay and co-workers (Figure 2b).[6]

Training of neural network is typically performed on variations of gradient descent based algorithms,[26] trying to

Table 1. Cross-Validated Results of Machine Learning[a]

descriptors	% correct		Matthews *cc*	
	ANN	SVM	ANN	SVM
GC	79.25 ± 0.66	80.01 ± 0.087	0.567 ± 0.012	0.592 ± 0.002
MOE	77.89 ± 0.74	80.19 ± 0.74	0.537 ± 0.013	0.593 ± 0.016
CATS_225	72.13 ± 0.88	73.90 ± 0.51	0.432 ± 0.013	0.485 ± 0.011
all (GC+MOE+CATS)	80.05 ± 1.02	82.24 ± 0.66	0.579 ± 0.018	0.633 ± 0.010

[a] Average values and standard deviations are given. The Levenberg−Marquardt training method was used for ANN training.

minimize an error function. To avoid overfitting cross-validation can be used for finding an earlier point of training.[28] In this work the neural network toolbox from MATLAB was used. Data were preprocessed identically to SVM based learning. We applied the following training algorithms to ANN optimization in their default versions provided by MATLAB: gradient descent with variable learning rate,[29,30] conjugated gradient descent,[30,31] scaled conjugated gradient descent,[32] quasi-Newton algorithm,[33] Levenberg−Marquardt (LM),[34,35] and automated regularization.[36] For each optimization ten-times cross-validation was performed (80+20 splits into training and test data), where the ANN weights and biases were optimized using the training data, and prediction accuracy was measured using test data to determine the number of training epochs, i.e., the endpoint of the training process. This was performed to reduce the risk of overfitting. It should be noted that the validation data were left untouched.

MODEL VALIDATION

The SVM model for drug/nondrug classification of a pattern x was

$$SVM(x) = \sum_i (a_i K(x_i^{SV}, x) + b)$$

Here, i runs only over support vectors (SV). The value of *SVM(x)* is either positive ("drug") or negative ("nondrug").

The ANN model for drug/nondrug classification produced values in]-1,1[, where a positive value meant "drug" and a negative value "nondrug".

Classification accuracy was evaluated based on prediction accuracy, i.e., percent of test compounds correctly classified, and the correlation coefficient according to Matthews:[37]

$$cc = \frac{NP - OU}{\sqrt{(N+O)(N+U)(P+O)(P+U)}}$$

where P, N, O, and U are the number of true positive, true negative, false positive, and false negative predictions, respectively. Drugs were considered as "positive set", the nondrug molecules formed the "negative set". The values of *cc* can range from −1 to 1. Perfect prediction gives a correlation coefficient of 1.

SVM and ANN models were developed using various sizes of training data to measure the influence of the size of the training set on the quality of the classification model. The number of training samples was iteratively diminished: Starting with an 80+20 random split of all available samples into training and validation subsets, at each of the following iterations we diminished the size of the training set to only 80% of the number of samples of the previous iteration. This allowed us to obtain better sampling for small training sets. 10-times cross-validation was performed, and average values of prediction accuracy and $\langle cc \rangle$ were calculated.

RESULTS AND DISCUSSION

The main aim of this study was to compare SVM and ANN classifiers in their ability to distinguish between sets of "drugs" and "nondrugs". We trained different neural network topologies, and performance of the best network was compared to the SVM classifier.

Two types of ANN architecture were considered: standard feed-forward networks with one hidden layer ("architecture I") and a feed-forward network with one hidden layer with additional direct connections from input neurons to the output ("architecture II") (Figure 2). The first type of ANN was used by Sadowski and Kubinyi in their original work on drug-likeness prediction;[7] the second architecture was employed by Ajay and co-workers serving the same purpose.[6] Using these networks and the GC descriptors in combination with the Levenberg−Marquardt training method, classification accuracy was identical to the original results (on average 80% correct) despite the use of a different training technique and different training data (Table 1). This observation substantiates the original findings. Both network types performed identically considering the error margin (approximately 80% correct classification). We observed that for some of the training algorithms a slightly lower standard deviation of the prediction accuracy was observed for architecture I (data not shown). Since the additional connections in network architecture II did not contribute to a greater accuracy of the model, we used only the standard feed-forward network with one hidden layer containing two neurons (architecture I) for further analysis.

For each training method and combination of input variables (descriptors) networks with different numbers of hidden neurons (2−10 neurons) were trained. Overall, we did not observe an overall best training algorithm. The Levenberg−Marquardt method was used for the development of the final ANN model. Also, we did not observe an improved classification result when the number of hidden neurons was larger than two (data not shown). ANN architecture I with two hidden neurons yielded the overall best cross-validated prediction result for all descriptors (GC+MOE+CATS), 80% correct predictions ($\langle cc \rangle = 0.58$). The rank order of descriptor sets with regard to the overall classification accuracy yielded was as follows: All > GC > MOE > CATS (Table 1). It should be stressed that the differences in classification accuracy are minute for the descriptors "All", MOE, and GC and should be regarded as comparable considering a standard deviation of 1%. The CATS descriptor led to approximately 5% lower accuracy.

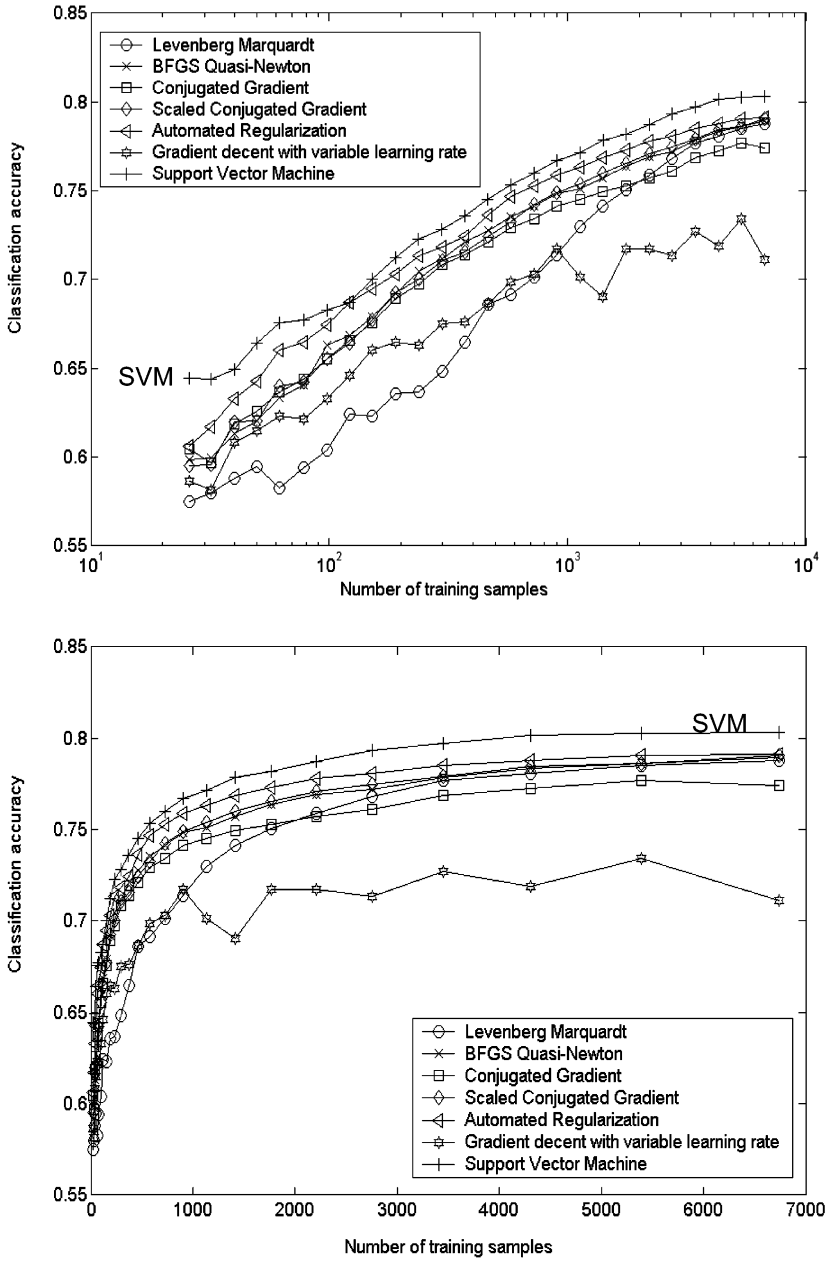

Figure 3. Average cross-validated prediction accuracy (fraction correct) of SVM and ANN classifiers optimized by various training schemes for GC descriptors (upper graph: logarithmic scale; lower graph: linear scale).

SVM training resulted in models showing slightly higher prediction accuracy than the ANN systems (Table 1). A 1−2% gain was observed, independent of the number of training samples and method used for neural network training. Figures 3 and 4 illustrate the dependency of the classification accuracy on the number of sample molecules used for training. In one experiment only GC descriptors were used (Figure 3), in a second study the combination of GC, MOE, and CATS descriptors was employed (Figure 4). With the GC descriptor the SVM estimator only slightly outperforms the neural networks (Figure 3). Similar results were obtained if only MOE or CATS descriptors were used for training (data not shown). The situation changed when all descriptors were used. With the complete descriptor set (525-dimensional) SVM clearly outperforms the neural network system (Figure 4). These results substantiate earlier findings that SVM performs better than ANN when large numbers of features or descriptors are used.[12]

A general observation was the fact that classification accuracy significantly improved with an increasing number of training samples, reaching a plateau in performance between 2000 and 3000 samples (Figures 3 and 4). The accuracy curves represent almost ideal learning behavior. It should be mentioned that the performance plateau observed does not reflect an inherent clustering of the data set, as training data subsets were randomly selected from the pool. The fraction correctly predicted grows from approximately 65% to 80% when the training set is increased by a factor of ∼250. The combination of MOE, GC, and CATS descriptors improved classification accuracy by approximately two percent for SVM and by one percent for ANN compared to models based on individual descriptors. These results demonstrate that an optimal ANN training to a large extent depends on the number of training patterns available and the type of molecular descriptors used. For instance, for GC descriptors the best learning algorithm was training with

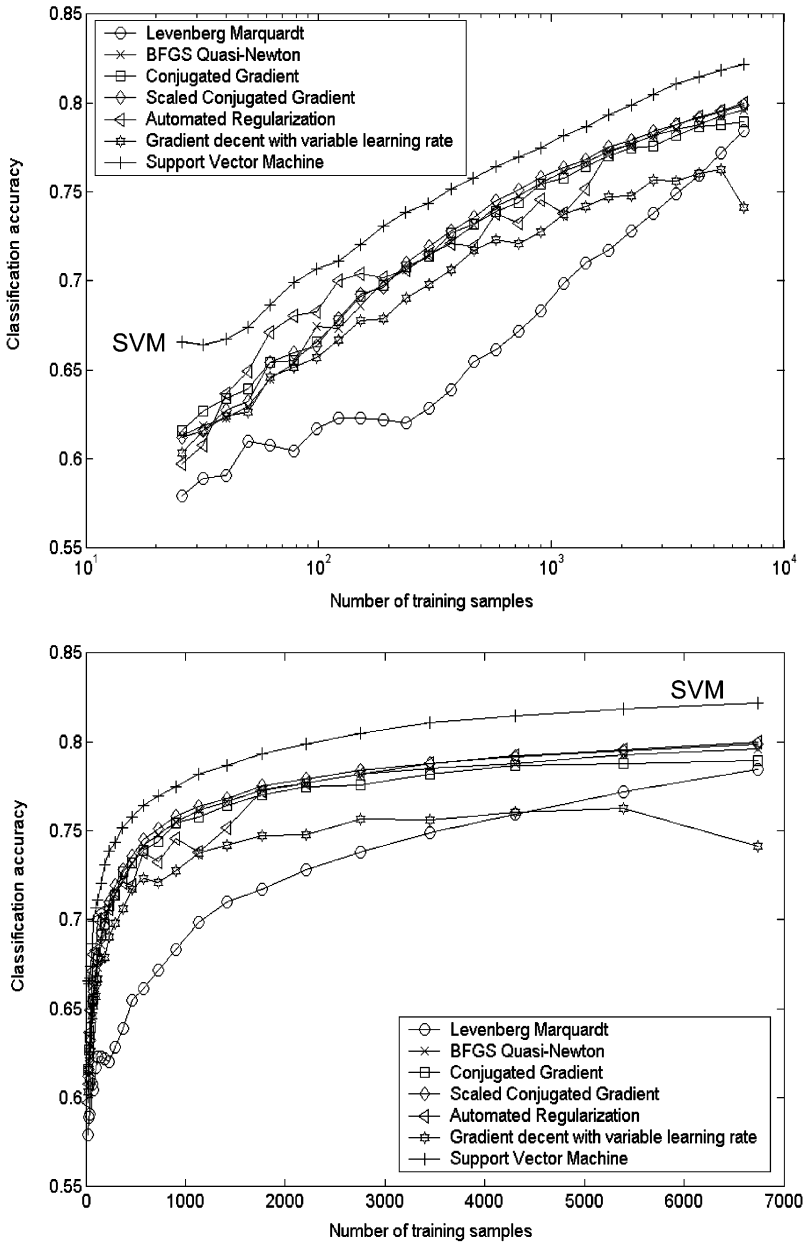

Figure 4. Average cross-validated prediction accuracy (fraction correct) of SVM and ANN classifiers optimized by various training schemes for the combination of GC, MOE, and CATS descriptors (upper graph: logarithmic scale; lower graph: linear scale).

automated regularization, but for the combination of GC, MOE, and CATS descriptors this algorithm was extremely slow and converged relatively unstable. In contrast, SVM generally performed more stably compared to ANN, with only a small increase in computation time for both sets of descriptors (Figures 3 and 4).

In a previous comparison of SVM to several machine learning methods by Holden and co-workers it was shown that an SVM classifier outperformed other standard methods, but a specially designed and structurally optimized neural network was again superior to the SVM model in a benchmark test.[13] This observation is supported by the observation that in the present study the set of molecules which were correctly classified by both SVM and ANN (mutual true positives) was 72% on average, and the fraction incorrectly classified by both systems (mutual false negatives) was 11%. 10% of the test data were correctly predicted by SVM but failed by ANN, and 6% were correctly classified by ANN but not by SVM using the full set of descriptors (GC+MOE+CATS). Examples of the latter two sets of molecules are shown in Figure 5. Clearly, the ANN classifier and the SVM classifier complement each other, and both methods could be further optimized, for example, by changing the SVM kernel or by exploring more sophisticated ANN architectures and concepts.

Fast classifier systems are mainly developed for first-pass virtual screening, in particular for identification ("flagging") of potentially undesired molecules in very large compound collections.[2] Due to robust convergence behavior SVM seems to be well-suited for solving binary decision problems in molecular informatics, especially when a large number of descriptors is available for characterization of molecules. In this study we have shown that two drug-likeness estimators can produce complementary predictions. We recommend the parallel application of both predictive systems for virtual screening applications. One possibility to combine several estimators for "drug-likeness" or any other classification task is to employ a "jury decision", e.g. calculate an ensemble

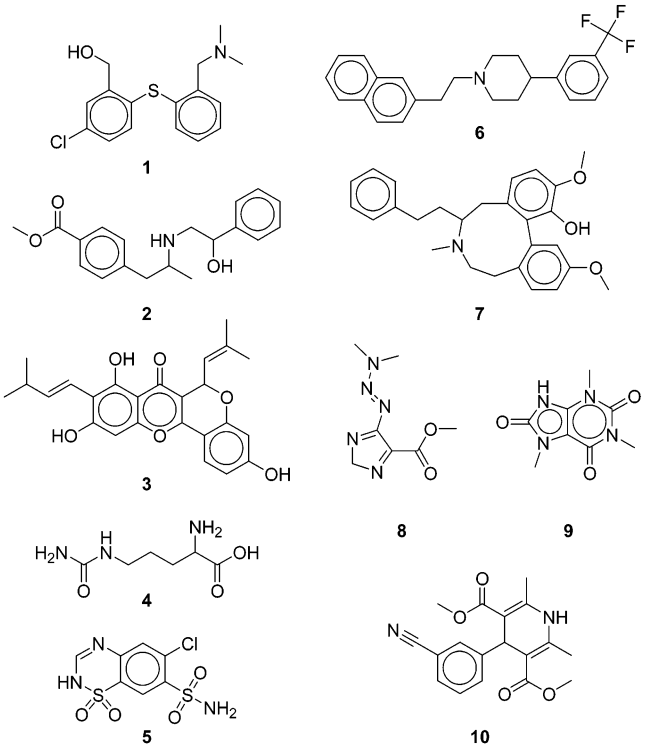

Figure 5. Examples of drugs correctly classified by ANN but not by SVM (structures 1−5), and drugs correctly classified by SVM but not by ANN (structures 6−10).

average.[38,39] As more and more different predictors become available for virtual screening a meaningful combination of prediction systems that exploits the individual strengths of the different methods will be pivotal for reliable compound library filtering.

CONCLUSION

It was demonstrated that the SVM system used in this study has the capacity to produce higher overall prediction accuracy than a particular ANN architecture. Based on this observation we conclude that SVM represents a useful method for classification tasks in QSAR modeling and virtual screening, especially when large numbers of input variables are used. The SVM classifier was shown to complement the predictions obtained by ANN. The SVM and ANN classifiers obtained for drug-likeness prediction are comparable in overall accuracy and produce overlapping, yet not identical sets of correctly and misclassified compounds. A similar observation can be made when two ANN models are compared. Different ANN architectures and training algorithms were shown to lead to different classification results. Therefore, it might be wise to apply several predictive models in parallel, irrespective of their nature, i.e., being SVM- or ANN-based. We wish to stress that our study does not justify the conclusion that SVM outperforms ANN in general. In the present work only a standard feed-forward network with a fixed number of hidden neurons was compared to a standard SVM implementation. Nevertheless, our results indicate that solutions obtained by SVM training seem to be more robust with a smaller standard error compared to standard ANN training. Irrespective of the outcome of this study, it is the appropriate choice of training data and descriptors, and reasonable scaling of input variables that determines the success or failure of machine learning systems. Both methods are suited to assess the usefulness of different descriptor sets for a given classification task, and they are methods of choice for rapid first-pass filtering of compound libraries.[40] A particular advantage of SVM is "sparseness of the solution". This means that an SVM classifier depends only on the support vectors, and the classifier function is not influenced by the whole data set, as it is the case for many neural network systems. Another characteristic of SVM is the possibility to efficiently deal with a very large number of features due to the exploitation of kernel functions, which makes it an attractive technique, e.g., for gene chip analysis or high-dimensional chemical spaces. The combination of SVM with a feature selection routine might provide an efficient tool for extracting chemically relevant information.

ACKNOWLEDGMENT

The authors are grateful to Norbert Dichter and Ralf Tomczak for setting up the LSF Linux cluster. Alireza Givehchi is thanked for assistance in installing the gecco! Web interface. This work was supported by the Beilstein-Institut zur Förderung der Chemischen Wissenschaften, Frankfurt.

REFERENCES AND NOTES

(1) Clark, D. E.; Pickett, S, D. Computational methods for the prediction of 'drug-likeness'. *Drug Discov. Today* **2000**, *5*, 49−58.
(2) Schneider, G.; Böhm, H.-J. Virtual screening and fast automated docking methods. *Drug Discov. Today* **2002**, *7*, 64−70.
(3) Wold, S. Exponentially weighted moving principal component analysis and projections to latent structures. *Chemomet. Intell. Lab. Syst.* **1994**, *23*, 149−161.
(4) Forina, M.; Casolino, M. C.; de la Pezuela Martinez, C. Multivariate calibration: applications to pharmaceutical analysis. *J. Pharm. Biomed. Anal.* **1998**, *18*, 21−33.
(5) *Neural Networks in QSAR and Drug Design*; Devillers, J., Ed.; Academic Press: London, 1996.
(6) Ajay; Walters, W. P.; Murcko, M. A. Can we learn to distinguish between "drug-like" and "nondrug-like" molecules? *J. Med. Chem.* **1998**, *41*, 3314−3324.
(7) Sadowski, J.; Kubinyi, H. A scoring scheme for discriminating between drugs and nondrugs. *J. Med. Chem.* **1998**, *41*, 3325−3329.
(8) Sadowski, J. Optimization of chemical libraries by neural networks. *Curr. Opin. Chem. Biol.* **2000**, *4*, 280−282.
(9) Schneider, G. Neural networks are useful tools for drug design. *Neural Networks* **2000**, *13*, 15−16.
(10) Sadowski, J. In *Virtual Screening for Bioactive Molecules*; Böhm, H.-J., Schneider, G., Eds.; Weinheim: Wiley-VCH: 2000; pp 117−129.
(11) Cortes, C.; Vapnik, V. Support-vector networks. *Machine Learning* **1995**, *20*, 273−297.
(12) Vapnik, V. *The Nature of Statistical Learning Theory*; Berlin: Springer, 1995.
(13) Burbidge, R.; Trotter, M.; Buxton, B.; Holden, S. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comput. Chem.* **2001**, *26*, 5−14.
(14) Warmuth, M. K.; Liao, J.; Ratsch, G.; Mathieson, M.; Putta, S.; Lemmen, C. Active learning with Support Vector Machines in the drug discovery process. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 667−673.
(15) Wilton, D.; Willett, P.; Lawson, K.; Mullier, G. Comparison of ranking methods for virtual screening in lead-discovery programs. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 469−474.
(16) Todeschini, R.; Consonni, V. *Handbook of Molecular Descriptors*; Weinheim: Wiley-VCH: 2000.
(17) Ghose, A. K.; Crippen, G. M. Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure−activity relationships 1. Partition coefficients as a Measure of hydrophobicity. *J. Comput. Chem.* **1986**, *7*, 565−577.
(18) Ghose, A. K.; Crippen, G. M. Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure−activity

relationships 2. Modeling dispersive and hydrophobic interactions. *J. Comput. Chem.* **1987**, *27*, 21−35.
(19) Ghose, A. K.; Pritchett, A.; Crippen, G. M. Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure−activity relationships 3. *J. Comput. Chem.* **1988**, *9*, 80−90.
(20) Schneider, G.; Neidhart, W.; Giller, T.; Schmid, G. "Scaffold-hopping" by topological pharmacophore search: a contribution to virtual screening. *Angew. Chem., Int. Ed. Engl.* **1999**, *38*, 2894−2896.
(21) Gasteiger, J.; Rudolph, C.; Sadowski, J. Automatic generation of 3D-atomic coordinates for organic molecules. *Tetrahedron Comput. Methods* **1990**, *3*, 537−547.
(22) Coleman, T. F.; Li, Y. A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM J. Optimization* **1996**, *6*, 1040−1058.
(23) Joachims, T. In Making large-scale SVM learning practical. *Advances in Kernel Methods − Support Vector Learning;* Schölkopf, B., Burges, C., Smola, A., Eds.; MIT-Press: Cambridge, MA, 1999; pp 41−56.
(24) Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*; Cambridge University Press: Cambridge, 2000.
(25) Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining Knowledge Discovery* **1998**, *2*, 121−167.
(26) Bishop, C. M. *Neural Networks for Pattern Recognition*; Oxford: Oxford University Press: 1995.
(27) Joachims, T. Learning to classify text using Support Vector Machines. Kluwer *International Series in Engineering and Computer Science 668*; Kluwer Academic Publishers: Boston, 2002.
(28) Duda, R. O.; Hart, P. E.; Stork, D. G. *Pattern Classification*; Wiley-Interscience: New York, 2000.
(29) Rumelhart, D. E.; McClelland, J. L.; The PDB Research Group. *Parallel Distributed Processing*; MIT Press: Cambridge, MA, 1986.
(30) Hagan, M. T.; Demuth, H. B.; Beale, M. H. *Neural Network Design*; PWS Publishing: Boston, 1996.
(31) Fletcher, R.; Reeves, C. M. Function minimization by conjugate gradients. *Comput. J.* **1964**, *7*, 149−154.
(32) Moller, M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks* **1993**, *6*, 525−533.
(33) Dennis, J. E.; Schnabel, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Prentice-Hall: Englewood Cliffs, 1983.
(34) Hagan, M. T.; Menhaj, M. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Networks* **1994**, *5*, 989−993.
(35) Foresee, F. D.; Hagan, M. T. Gauss−Newton approximation to Bayesian regularization. *Proceedings of the 1997 International Joint Conference on Neural Networks*; pp 1930−1935.
(36) MacKay, D. J. C. Bayesian interpolation. *Neural Comput.* **1992**, *4*, 415−447.
(37) Matthews, B. W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta* **1975**, *405*, 442−451.
(38) Krogh, A.; Sollich, P. Statistical mechanics of ensemble learning. *Phys. Rev. E* **1997**, *55*, 811−825.
(39) Baldi, P.; Brunak, S. *Bioinformatics − The Machine Learning Approach*; MIT Press: Cambridge, 1998.
(40) Byvatov, E.; Schneider, G. Support vector machine applications in bioinformatics. *Appl. Bioinf.* **2003**, *2*, 67−77.

CI0341161