Automatic Generation of Complementary Descriptors with Molecular Graph Networks

Christian Merkwirth*,†,‡ and Thomas Lengauer*,†

Computational Biology and Applied Algorithmics Group, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, and Department for Information Technology, Faculty of Physics, Astronomy, and Applied Computer Science, Jagiellonian University, Reymonta 4, 30-059 Kraków, Poland

Received December 20, 2004

We describe a method for the automatic generation of weakly correlated descriptors for molecular data sets. The method can be regarded as a statistical learning procedure that turns the molecular graph, representing the 2D formula of the compound, into an adaptive whole molecule composite descriptor. By translating the molecular graph structure into a dynamical system, the algorithm can compute an output value that is highly sensitive to the molecular topology. This system can be trained by gradient descent techniques, which rely on the efficient calculation of the gradient by back-propagation. We present computational experiments concerning the classification of the Developmental Therapeutics Program AIDS antiviral screen data set on which the performance of the method compares with that of approaches based on substructure comparison.

1. INTRODUCTION

The varying number of atoms in a molecule and the varying topology stand in the way of a straightforward representation of the molecule's structure as a fixed length vector that can be subsequently fed into a conventional statistical learning apparatus. When performing statistical learning on molecular data sets, quantitative structure−activity relationship approaches circumvent this problem by proposing a set of *descriptors* that quantify properties or characteristics of the molecule and that can be computed given the molecular structure (see Lengauer et al.[1] and references therein for a review of existing descriptors). However, the huge number of putative descriptors entails the problem of feature selection when using these descriptors as input for a statistical learning procedure. This problem turns out to be ill-posed for very high-dimensional data sets with only a small number of observations.

Descriptors that just rely on the types of the atoms in a molecule cannot correctly account for the influence of the molecular structure on the chemical characteristics. For many chemical applications, the structure of a molecule can be conveniently represented in the form of a graph. The method we propose directly establishes a functional relationship between the molecular graph representing a molecule and the property of interest, for example, therapeutic activity. By attaching a state value to each node in the molecular graph that evolves over a predefined number of iterations, the system is able to compute an output value that is sensitive not only to the initial conditions given but also to the molecular topology. Small changes in the topology can result in profound changes of the network states after several iterations. The resulting output value constitutes the decision variable for discrimination between two or more classes in the case of classification or is real-valued in the case of regression. The key idea is to make the whole system trainable. By strengthening or weakening the parameters of the equation governing the evolution of state values, we can adapt the system to a desired output in the settings of both classification and regression. The adaptation of these parameters is carried out during the training phase in a way similar to the training of neural networks.

Related approaches have been proposed in the literature (see Kireev[2] and Schädler and Wysotzki[3]); however, no successful methods for training these networks have been proposed. In this work, we propose to employ the well-known back-propagation procedure in combination with stochastic gradient descent to train the proposed network structure in order to perform supervised learning on data sets comprised of molecules and corresponding output properties. We proposed a similar approach for training the closely related processing structures known as cellular neural networks[4,5] used in the fields of digit recognition and image processing. The application of nonstandard neural networks in combination with recent developments in computational learning theory (ensemble methods[6] and robust loss functions) enables us to successfully perform classification and regression on real-world chemical data sets.

2. OUTLINE OF THE METHOD

The method introduces a new statistical model called the *molecular graph network* (MGN), which makes use of *feature nets*. In each feature net, the dynamic evolution of node states of the molecular graph is used in order to establish a functional relationship between the molecular graph and a scalar output value. The outputs of several feature nets are fed into a *supervisor neural network*, which computes the final output value. The combination of several feature nets and a supervisor network constitutes the molecular graph network (see Figure 1).

A graph, denoted by G, is a dimensionless mathematical object representing a set of nodes, called vertices, and

* Corresponding authors. E-mail: cmerk@mpi-sb.mpg.de (C.M.), lengauer@mpi-sb.mpg.de (T.L.). Tel.: +49-681-9325-318.
† Max-Planck-Institut für Informatik.
‡ Jagiellonian University.

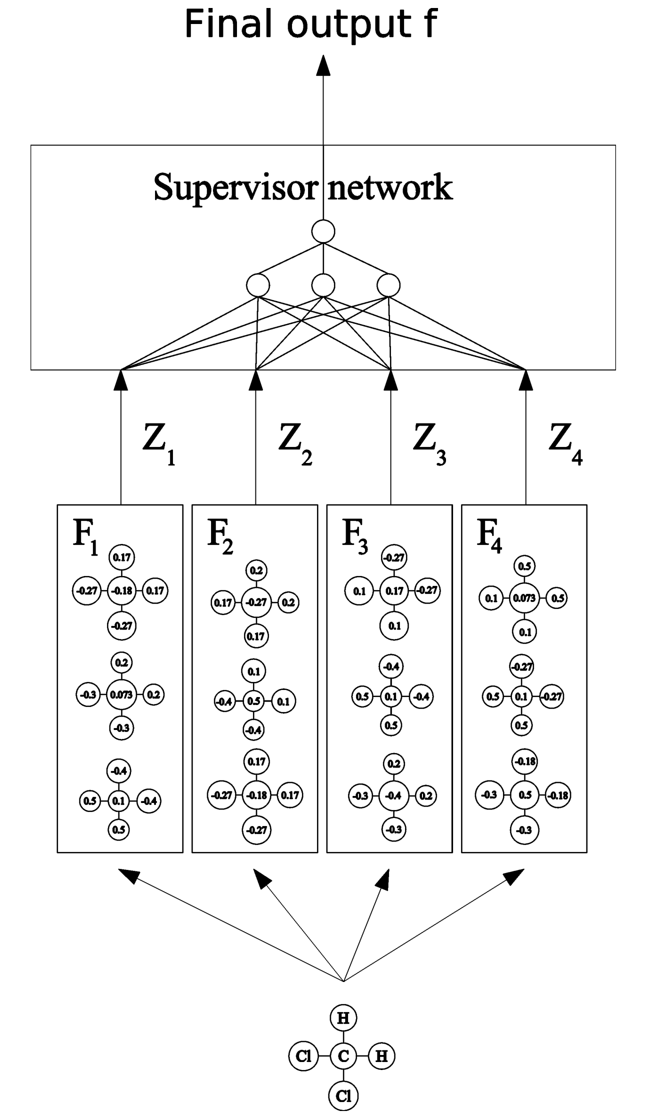

Figure 1. Processing scheme of a molecular graph network. Here, the information describing the molecular graph of dichloromethane, namely, the vector of element types $\vec{e}$ and the adjacency matrix B, is fed in parallel into four individual feature nets F_1–F_4. Each feature net is described by its own parameter vector $\vec{p}_1$–$\vec{p}_4$. The scalar outputs z_1–z_4 of F_1–F_4 are used as input for the supervisor network. The supervisor network is a conventional feed-forward neural network with a single hidden layer that computes the final output value for the given compound. The numerical state values displayed exemplify the processing within the feature nets.

connections between pairs of nodes, called edges. Two vertices connected by an edge are called adjacent. In cheminformatics, graphs are used to represent molecules. In such a molecular graph, the vertices represent atoms and the edges represent bonds. Graph vertices and edges are labeled, vertices by their atom type (C, N, O, etc.) and edges by the bond type (single, double, aromatic, etc.). While hydrogens bound to carbon are usually not taken into account, for this approach, the user can choose either to consistently suppress hydrogens or to consistently include them in the representation of the molecular graph. A multiple bond between adjacent atoms is represented by a single edge with an adequate bond type (e.g., double). The structure of most organic molecules having covalent bonds can be represented as such a molecular graph. Ionic bonds can be included into the range of bond types if necessary. In the version described in this study, the method is not able to account for stereochemistry. By imposing new bond types and making weights dependent on the steric conformation, it should be possible to account for stereochemistry without a major redesign of the algorithm.

The key idea of the feature nets is to use this molecular graph as a *dynamical system* such that each network node has a scalar state value that is iteratively updated using a nonlinear function. For each node, this function computes a weighted sum of the states of all adjacent nodes and subsequently applies a sigmoidal transform to this sum. Node states are abstract values and are generally not related to the concepts of *chemical* or *quantum* states of the respective atoms. The iterations of the dynamical system should not be mixed up with the epochs of the back-propagation training described in Section 3.4.

The weights governing the dynamic evolution of a feature net do not pertain to a specific position within the network; instead, element and bond type of the node determine which weights are taken from several common *weight tables*. The tables constitute the adjustable parameters of a feature net. They are inferred from the information given by the training data during the training of the network. This provision enables the method to compute output values for arbitrary organic molecules that are not necessarily present in the training data set.

3. MOLECULAR GRAPH NETWORKS

3.1. Feature Nets. A molecular graph m can be coded as a vector of element types $\vec{e}$ of the molecule's atoms and an adjacency matrix B storing information on the covalent or ionic bonds between every pair of atoms within the molecule. Whenever atoms i and j are connected by a bond, the corresponding entry B_{ij} will be a nonzero integer coding for the type of the bond (single, double, aromatic, etc.). Also, element types are coded as positive integers such that they can be used for indexing. Let N denote the number of atoms in the molecule under consideration.

A feature net is a function that maps a molecular graph to a scalar output value. To compute this output z for a given molecule m, the molecular graph is translated into a dynamical system. This implies that the topology of the dynamical system is different for different compounds. Nevertheless, it is possible to calculate node states for every possible molecular structure since feature network weights are located in the weight tables by the element type of the atom that is represented by this node and the types of bonds attached to these atoms. There are no additional input values that are fed into the feature net; instead, the molecular structure m, given by ($\vec{e}$, B), can be considered as the *input* given to the feature net.

A feature net is implemented as a discrete-time spatio-temporal dynamical system that evolves over T iterations. Each node i of the dynamical system starts with an initial state y_i^0 depending on its element type e_i (e.g, 6 for C, 92 for U), taken from the vector of initial states h:

$$y_i^0 = h_{e_i} \qquad i = 1, ..., N \tag{1}$$

To calculate the output for one compound, we compute the dynamic evolution of the node states y_i^t for iterations $t = 0$, ..., $T - 1$ according to the following equations:

$$x_i^{t+1} = \sum_{\text{atom } j \text{ adjacent to } i} A^t_{e_i,B_{ij}} y_j^t + c^t_{e_i}$$

$$y_i^{t+1} = \sigma(x_i^{t+1}) \tag{2}$$

A^t is the weight table, and c^t is the offset vector used for iteration t. The molecule's topology, stored in e_i and B_{ij}, is used in eq 2 to generate the indices into matrices A^t and vector c. For example, when $e_i = 8$ and $B_{ij} = 2$, $A^t_{e_i,B_{ij}}$ refers to element $A^t_{8,2}$. Hence, the terms involved in the sum in eq 2 will be different for each unique molecule processed.

Since a feature net usually contains more than one atom, we have to convert the output states y_i^T into a single, scalar output value z suitable as a decision variable. This is accomplished by computing the weighted average over the state values of all nodes. The scalar output z of a feature net is, thus, independent of the ordering of the atoms in the element table:

$$z = \frac{1}{N}\sum_{i=1}^{N} b_{e_i} y_i^T$$

Individual weight tables A^t and offsets c^t are used for each iteration t. In combination with the vector of initial states h and the vector of output weights b, they constitute the adjustable parameters $\vec{p}$ or so-called weight tables of a single feature net:

$$\vec{p} = \{h, b, A^0, c^0, A^1, c^1, ..., A^{T-1}, c^{T-1}\}$$

During the dynamic evolution, the state information of each node spreads through the network along the bonds. After T iterations, the information on a node is propagated over a maximal distance of T edges. This interaction allows for detection of functional groups without explicitly defining these groups. It also allows for detection of interactions between functional groups when the number of iterations T is sufficiently large.

Figure 2 exemplifies the mode of operation of a feature net with an iteration depth of $T = 2$. It shows the evolution of node states for two molecules that differ only by a single atom, dichloromethane and monochloromethane, in the left and the right columns, respectively. Both molecules consist of five atoms; all bonds are single bonds. For simplicity, the weight tables governing the evolution are the same for every iteration, $A^1 = A^2 = (0.6, 1.1, -0.9)^\dagger$ and $c^1 = c^2 = (-0.05, 0.08, -0.2)$ for C, H, and Cl; the vector of initial states reads $h = (0.1, -0.4, 0.5)$. The weight tables A^1 and A^2 are here matrices of size 3×1 since we have three element types and all bonds are single bonds; thus, $B_{ij} = 1$ for all connected pairs of atoms. Each atom has a state value that evolves as a function of the states of the connected atoms; for example, the state value of the central carbon after the first iteration is computed as $y_C^1 = \sigma(-0.05 + 0.6 \times -0.4 + 0.6 \times -0.4 + 0.6 \times 0.5 + 0.6 \times 0.5) = 0.0734$ for dichloromethane and $y_C^1 = \sigma(-0.05 + 0.6 \times -0.4 + 0.6 \times -0.4 + 0.6 \times -0.4 + 0.6 \times 0.5) = -0.4617$ for monochloromethane. Values depicted in the figure are rounded. While, after the insertion of the initial state values according to eq 1, the difference between both molecules manifests itself only in a difference of the state values of the differing atoms, already, after the first iteration according to eq 2, the state values of the central carbons differ. After the second iteration, the difference in a single atom type resulted in different state values for all five atoms. Despite the local interaction scheme of eq 1, the information about small structural differences in a distant part of the molecule will spread over all its atoms, given the number of iterations is high enough.

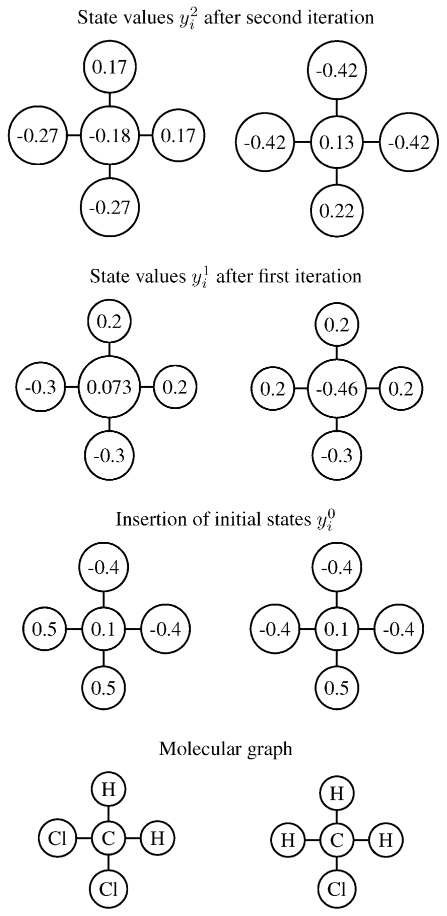

Figure 2. Evolution of feature net states over two iterations exemplified for two molecules that differ only by a single atom. The left column shows the processing for dichloromethane, the right that for monochloromethane. Both molecules consist of five atoms; all bonds are single bonds. The molecular graph is displayed at the bottom; the state values after the last iteration are at the top of the figure. While, after the insertion of the initial state values, the difference between both molecules manifests itself only in a difference of the state values of the differing atoms, already, after the first iteration, the state value of the central carbons differs. After the second iteration, the difference by a single atom resulted in ifferent state values for all five atoms.

The activation function $\sigma(x)$ is a sigmoidal function that does not fully saturate for input values x far from 0 (see Figure 3). This feature enhances the ability of the gradient descent algorithm (see Section 3.3) to escape from solutions for which a node has such a high activation that the derivative of a saturating sigmoidal activation function nearly vanishes and the gradient becomes zero, thus preventing the gradient descent algorithm from improving the involved weights any further.[7]

3.2. Supervisor Network. A single feature network usually does not offer enough capacity to approximate a

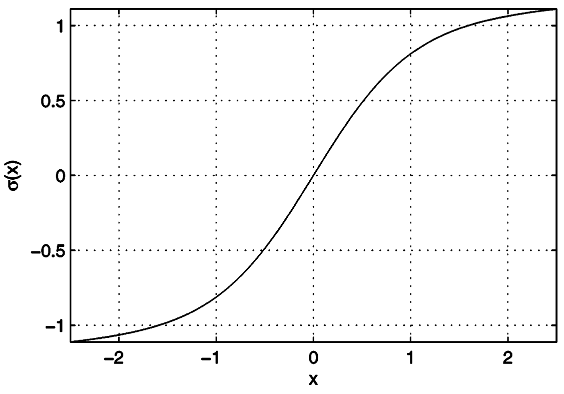

Figure 3. Sigmoidal activation function used as nonlinearity $\sigma(x)$. Note that the function does not saturate far from zero; instead, the slope decreases to 0.05 to prevent the gradient descent algorithm from getting stuck at nodes with activation $|x| \gg 1$.

difficult regression or classification setting. We, therefore, feed the outputs z_q ($q \in 1, ..., Q$) of Q individual feature nets into a conventional fully connected supervisor neural network (see Figure 1). The output f of this supervisor network for a molecule m_n constitutes the final output of the MGN:

$$f(m_n,\vec{P}) = f[z_1(m_n,\vec{p}_1), ..., z_q(m_n,\vec{p}_Q),\vec{p}_0]$$
$$\vec{P} = \{\vec{p}_1, ..., \vec{p}_Q,\vec{p}_0\} \qquad (3)$$

We chose a simple feed-forward neural network with a single hidden layer as the topology of the supervisor network. The number of neurons in this hidden layer can be adjusted to the complexity of the task; common choices range from 4 to 32 neurons. The concept of topology as is used for feed-forward neural networks cannot be applied to the feature nets since the number of nodes and their connectivity varies from sample to sample of the data set. The only remaining parameter controlling the complexity of a feature net is, thus, the number of iterations T_q. All adjustable parameters of the feature nets $\vec{p}_1, ..., \vec{p}_Q$ and of the supervisor network $\vec{p}_0$ are bundled, according to eq 3, into the total parameter vector $\vec{P}$ that exhaustively describes the MGN.

We observed that the outputs of the feature nets that constitute the input of the supervisor network tend to be pairwise uncorrelated after the training algorithm approached a local minimum of the loss function. Thus, feature nets automatically specialize on *complementary* features of the molecules in the training set. Manually devised descriptors are prone to cluster into several mutually correlated groups, which raises the problem of sophisticated feature selection and reduction. We try to foster the tendency of the feature nets to automatically generate complementary outputs by randomly initializing the network's parameters $\vec{P}$ at the beginning of the training. Additionally, we vary the iteration depth of the individual feature nets to induce a forced break of symmetry. The output of the feature nets can, thus, be seen as automatically generated descriptors. Experimental results on the correlation of the feature net outputs are detailed in Section 5.2.

3.3. Training. In the benchmark examples presented later, the number of adjustable weights $\vec{P}$ of the entire molecular graph network consisting of several feature nets and a supervisor network can exceed 50 000, though not all of these will actually be used. While the variable topology of the dynamical system constructed by a feature net appears unsuitable for a training by gradient-based optimization schemes, it turns out to be actually possible to efficiently compute the gradient for the feature nets much in the same way as it is done for neural networks. Thus, we can revert to the whole range of gradient-based optimization techniques developed for neural networks, even with the more complicated structure of a whole MGN making a careful choice necessary. The large number of adjustable weights additionally raises the problem of choosing an efficient training algorithm.

3.4. Gradient Calculation by Back-Propagation. Back-propagation[8] is an elegant method for efficiently computing the gradient of the loss function $L[y_n, z(m_n)]$ for a given molecule m_n with respect to the total parameter vector $\vec{P}$:

$$\vec{g}_n(\vec{P}) = \frac{\partial L[y_n, f(m_n,\vec{P})]}{\partial \vec{P}} = \frac{\partial L_n}{\partial \vec{P}}$$

In the *forward pass*, the output of the MGN $f(m_n,\vec{P})$ is computed and intermediate states have to be stored for the *backward pass*. The backward pass traverses the system in reverse order, starting from the last iteration of the forward pass. When the chain rule of differentiation is applied, it is possible to obtain the backward recursion, eq 6, for the backward pass, which closely resembles the forward equations, eq 2, though the application of the activation function is converted into a multiplication with the derivative of the activation function.

Since the supervisor net is a conventional feed-forward neural network, the gradient $\partial L[y_n, f(m_n,\vec{P})]/\partial \vec{p}_0$ with respect to the supervisor network's parameters $\vec{p}_0$ can be easily computed by back-propagation, as described in various textbooks.[8,9] As a byproduct, we obtain the gradient $\partial L_n/\partial z_q$ of the loss function with respect to the scalar outputs z_q of the feature nets.

Next, we describe the computation of the gradient $\partial L_n/\partial \vec{p}_q$ for feature net q. Despite the unusual processing scheme of the feature nets, the gradient for a single feature net can be calculated by back-propagation in much the same way as for conventional neural networks. Care has to be taken of the varying topology of the feature nets. For means of simplicity, we introduce an auxiliary variable s_i^t that represents the gradient of the loss function with respect to the feature net states y_i^t of feature net q:

$$s_i^t = \frac{\partial L_n}{\partial y_i^t} \qquad (4)$$

At this point, the gradient of the loss function with respect to the inputs of the supervisor network $\partial L_n/\partial z_q$ is inserted into the equation for the initial gradient s_i^T:

$$s_i^T = \frac{1}{N}\frac{\partial L_n}{\partial z_q} b_i \qquad (5)$$

Since, for back-propagation, the iteration order is reversed, s_i^T is the initial gradient, and it allows the computation of all auxiliary gradients s_j^t by the following recursion:

$$s_j^{t-1} = \sum_{i \text{ adjacent to } j} A_{e_i,B_{ij}}^{t-1} s_i^t \sigma'(x_i^t) \qquad (6)$$

From s_{ij}^t, all derivatives with respect to the parameters can be obtained:

$$\frac{\partial L_n}{\partial A_{e_i,B_{ij}}^{t-1}} = s_i^t \sigma'(x_i^t) y_j^{t-1} \tag{7}$$

$$\frac{\partial L_n}{\partial c_{e_i}^{t-1}} = s_i^t \sigma'(x_i^t) \tag{8}$$

$$\frac{\partial L_n}{\partial h_{e_i}} = \frac{\partial L_n}{\partial y_i^0} = s_i^0 \tag{9}$$

$$\frac{\partial L_n}{\partial b_{e_i}} = \frac{1}{N} \frac{\partial L_n}{\partial z_q} y_i^T \tag{10}$$

Usually, there are several contributions to the gradient with respect to a certain entry of A^t, c^t, $\vec{h}$, and $\vec{b}$. Care has to be taken in eqs 6−10 when summing up these contributions. From the previous equations, we can see that only state values y_i^t and derivatives of the activation function $\sigma'(x_i^t)$ have to be stored during the forward pass for molecule m_n. This results in a space complexity linear in the number of state variables y_i^t, which is the number of atoms N of the respective compound.

The gradient $\vec{g}_n$ can now be used for minimization of training loss. Since the total loss L is the sum of the individual losses L_n, the gradient of the total loss $\partial L/\partial \vec{P}$ with respect to any parameter $\vec{P}$ of the molecular graph network is the sum of the gradients of the individual losses:

$$\vec{g}_n(\vec{P}) = \frac{\partial L}{\partial \vec{P}} = \sum_{n=1}^{N} \frac{\partial L_n}{\partial \vec{P}} \tag{11}$$

Numerous methods exist for network training, given the gradient of the loss function, from which we found two to perform well in our case, *stochastic gradient descent* and *resilient propagation* (RPROP). Both training procedures start from randomly initialized weight tables and advance iteratively to a local minimum of the training loss.

3.5. Stochastic Gradient Descent. Stochastic gradient descent is an optimization technique that relies on back-propagation for gradient computation. It is also known as online learning. In contrast to batch learning, where gradients are accumulated over all samples of the training set according to eq 11 before a gradient step is performed, a gradient step with very small step size μ is carried out, here, after the gradient for molecule m_n has been calculated:

$$\vec{P}_{n+1} = \vec{P}_n - \mu \vec{g}_n(\vec{P}_n) \tag{12}$$

Since the total parameter vector $\vec{P}$ contains all the parameters describing a MGN exhaustively, eq 12 allows an update of all adjustable parameters of the feature nets within this MGN, including all weight tables A^t and offset vectors c^t. One complete pass of the algorithm through the training set is called an *epoch*. For every epoch, the order in which the samples of the training data set are processed is randomized. The global step size μ is usually decreased exponentially every time a certain number of samples has been processed. Since stochastic gradient descent actually never converges in a strong sense, one has to terminate the algorithm manually. In the simulations below, the number of gradient steps is fixed, not the number of epochs.

We experienced problems in the stability of the convergence of stochastic gradient descent when training molecular graph networks. These could be solved by using the concept of *minibatches*. We sum up the gradients of a fixed, small number of samples (e.g., 10−25) to smooth the gradient before performing a gradient step.

Stochastic gradient descent offers several advantages over batch learning. First of all, it is much faster on large data sets. Solutions for classification problems exhibit a lower generalization error. Also, the absolute value of the network weights tends to be lower, even without applying regularization. A more detailed discussion of the properties of stochastic gradient descent is given in LeCun et al.[10]

3.6. Resilient Propagation. Another optimization technique based on gradient descent is the RPROP algorithm[11] that we employ with the improvements proposed by Igel et al.[12] The batch learning algorithm uses a simple yet powerful scheme for adapting individual step sizes for every network weight and is heuristically one of the fastest algorithms for neural network training. The adaptive step size enables the RPROP algorithm to find its way down, even when confronted with complex error landscapes. It turns out, heuristically, that RPROP is better suited for data sets of only a few hundred samples and that it is better suited for regression problems rather than stochastic gradient descent.

3.7. Class Frequency Normalization. In the case of discrimination of therapeutically active compounds, a molecular graph network has to distinguish a few actives from a much larger number of inactives. When stochastic gradient descent is used, this leads to a large ratio between positive and negative examples of usually much more than 10; that is, on average, each gradient step of a positive example is followed by at least 10 gradient steps of negative examples. We can significantly improve the learning rate and the classification performance if we balance the number of positive and negative examples in the training process. It is, therefore, common practice to enrich the data set with duplicates of positive examples until the ratio becomes approximately 1:1. To avoid an increased memory consumption for storing replicated samples, we implement the class frequency normalization by randomly drawing training samples for the stochastic gradient descent with accordingly adjusted probabilities.

3.8. Building Classifier Ensembles. A common way of improving the performance of statistical learning algorithms is ensemble building.[6] In this study, this is done consistently by training several molecular graph networks on randomly chosen subsets of the training data. The training of each MGN starts with randomly initialized weight tables. To compute the output of the ensemble for one compound, the output decision variables of all MGNs belonging to that ensemble are averaged. Note that when doing multiclass classification with the logistic loss function, we have to construct one such ensemble for each of the three binary classification tasks. Throughout this study, the out-of-train (OOT) technique is employed as described in Merkwirth et al.[13] to assess the extra-sample error (e.g., classification accuracy) of the constructed ensemble models.

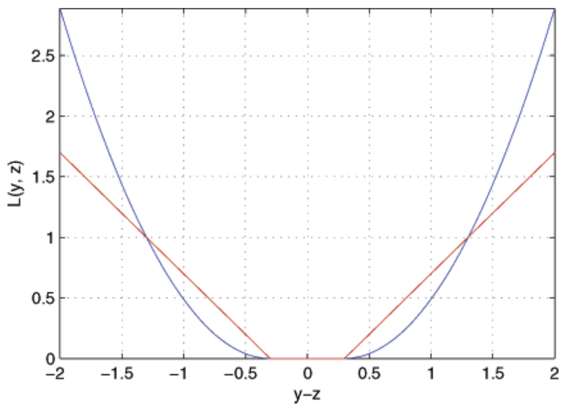

Figure 4. ϵ-insensitive squared (blue line) and absolute loss (red line) used for the training of the molecular graph networks, here with $\epsilon = 0.3$. Training observations that fall into the ϵ margin do not contribute to the total loss and result in a vanishing gradient.

3.9. Loss Function for Regression. For regression settings, we propose the use of ϵ-insensitive squared loss. Here, the loss and the gradient vanish as soon as the output of the model falls inside the ϵ margin around the desired value (see Figure 4 and eq 13). This behavior mitigates the function estimation problem and makes the solution of the regression problem nonunique, the latter being advantageous for model decorrelation.[6] Though the derivative of this function is not continuous, it can be seamlessly used with gradient descent techniques.

$$L_\epsilon(y, z) = \begin{cases} 0 & : \ |y - z| \le \epsilon \\ (|y - z| - \epsilon)^2 & : \ |y - z| > \epsilon \end{cases} \tag{13}$$

3.10. Loss Function for Classification. *Logistic Loss.* The logistic transform $p = \exp(z)/[1 + \exp(z)]$ is commonly used to transform the possibly unbounded output of a classifier into a quantity that behaves like a probability. This pseudoprobability can then be used to compute a maximum-log likelihood that, with a negative sign to turn the maximization into a minimization, can act as loss function as given by eq 14 (see also Figure 5).

$$L_\epsilon(y, z) = \begin{cases} -\log\left[\dfrac{\exp(z)}{1 + \exp(z)}\right] & : \ y = 1 \\ -\log\left[1 - \dfrac{\exp(z)}{1 + \exp(z)}\right] & : \ y = 0 \end{cases} \tag{14}$$

Classification Loss. Classification loss (see Figure 6 and eq 15) mitigates the function estimation problem since training observations for the output of the molecular graph network that fall on the correct side of the threshold have zero loss and gradient, regardless of the distance to the threshold. This resembles the support vector concept where observations lying on the correct side of the separating hyperplane do not influence the positioning of that hyperplane. In the case of support vector classification, the hyperplane is determined only by observations that lie on the margin or violate it (the so-called support vectors[14]). Additional benefits of the classification loss are, on one hand, that training patterns that cannot be correctly classified contribute only linearly to the loss. This renders the classification loss more robust with respect to outliers in the data set than quadratic loss. Additionally, classification loss leads to a speedup of the gradient descent training since no backward pass and no parameter update have to be performed when the gradient is exactly zero.

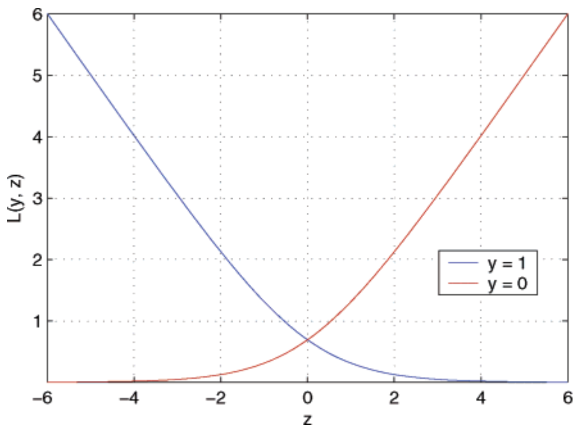

Figure 5. Logistic loss function used for training three ensembles of molecular graph networks on the NCI AIDS antiviral screen data set. The red line depicts the loss for negative training samples, the blue line that for positive training samples.

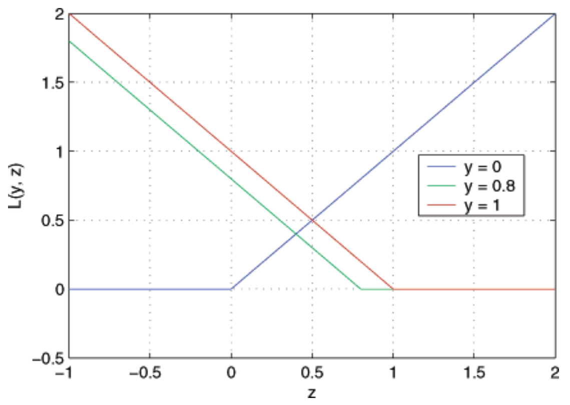

Figure 6. Classification loss function used for training an ensemble of molecular graph networks on the NCI AIDS antiviral screen data set. The blue line depicts the loss for negative training samples, the green line that for positive training samples with a desired output value of 0.8, and the red line that for positive training samples with a desired output value of 1.0.

$$L_\epsilon(y, z) = \begin{cases} 0 & : \ y = 0 \wedge z \le 0 \\ z & : \ y = 0 \wedge z > 0 \\ 0 & : \ y > 0 \wedge y \le z \\ y - z & : \ y > 0 \wedge y > z \end{cases} \tag{15}$$

4. COMPUTATIONAL EXPERIMENTS

4.1. National Cancer Institute AIDS Antiviral Screen Data Set. We considered a data set of more than 42 000 compounds from the Developmental Therapeutics Program AIDS antiviral screen data set of the National Cancer Institute (NCI) Open Database. The antiviral screen utilized a soluble formazan assay to measure the ability of compounds to protect human CEM cells[15] from HIV-1-induced cell death. In the primary screening set of results, the activities of the compounds tested in the assay were described to fall into three classes: confirmed active (CA) for compounds that provided 100% protection, confirmed moderately active (CM) for compounds that provided more than 50% protection, and confirmed inactive (CI) for the remaining compounds or compounds that were toxic to the CEM cells and, therefore, seemed to not provide any protection. The data set was obtained from http://cactus.nci.nih.gov/ncidb2/download.html. The data set originally consisted of 42 689 2D structures with AIDS test data as of October 1999 and was provided in SDF format. Seven compounds could not be parsed and had to be removed. From the total of 42 682 useable compounds, 41 179 compounds were confirmed inactive, 1080 compounds were confirmed moderately active, and 423 compounds were confirmed active. No information on putative targets was used for the computational experiments. Also, stereochemical information, even when it was available in the SDF file, did not enter the computational experiments.

We compared two loss functions on this multiclass classification problem.

When using classification loss, we assigned a training output of 0 to all compounds of class CI, 0.8 to all compounds of class CM, and 1.0 to all compounds of class CA. A value of 0.8 instead of 0.5 was chosen for the confirmed moderately active compounds to indicate to the algorithm that these compounds should be considered rather as being active than as being inactive. Using classification loss simplifies the training since only one ensemble of classifiers has to be trained on all compounds of the training data set. However, converting the continuous output of the classifier ensemble back into class labels necessitates the choice of two thresholds τ_1 and τ_2 in order to discriminate between the three classes. We bypassed this problem by either considering classes CI and CM as inactive and class CA as active or considering only class CI as inactive and classes CM and CA as active, thus converting the problem to a binary classification problem for which we can generate receiver operating characteristic (ROC) curves depending on only one threshold.

We randomly partitioned the data set into a training set of 35 000 compounds and a test set of 7682 compounds. We constructed an ensemble of 15 MGNs. Each MGN consisted of 19 individual feature nets with iteration depths ranging from 3 to 10 and a supervisor network with 24 hidden layer neurons. The MGNs were trained by stochastic gradient descent with a fixed number of 10^6 gradient calculations. The global step size μ was decreased by a factor of 0.8 every 70 000 gradient updates. Each MGN was trained on a random 80% of the 35 000 training samples. Thus, the OOT output for every sample of the training set was computed by averaging over three models, and the output for the held-out test set was computed by averaging over all 15 models of the ensemble of MGNs.

In the one-versus-all approach with logistic loss functions, we construct three ensembles of classifiers. Each of these three ensembles is trained to solve the binary classification problem of discriminating one of the three classes against the rest and each consists of six MGNs. Every MGN consisted of 18 individual feature nets with iteration depths ranging from 3 to 10 and a supervisor network with 24 hidden layer neurons. The MGNs were trained by stochastic gradient descent with a fixed number of 10^6 gradient calculations. The global step size μ was decreased every 70 000 gradient updates by a factor of 0.8. Again, we randomly partitioned the data set into a training set of 35 000 compounds and test set of 7682 compounds. Each MGN was trained on a random $^2/_3$ of the 35 000 training samples. Thus, the OOT output for every sample of the training set was computed by averaging over two models, while the output for the held-out test set was computed by averaging over all six models of each ensemble.

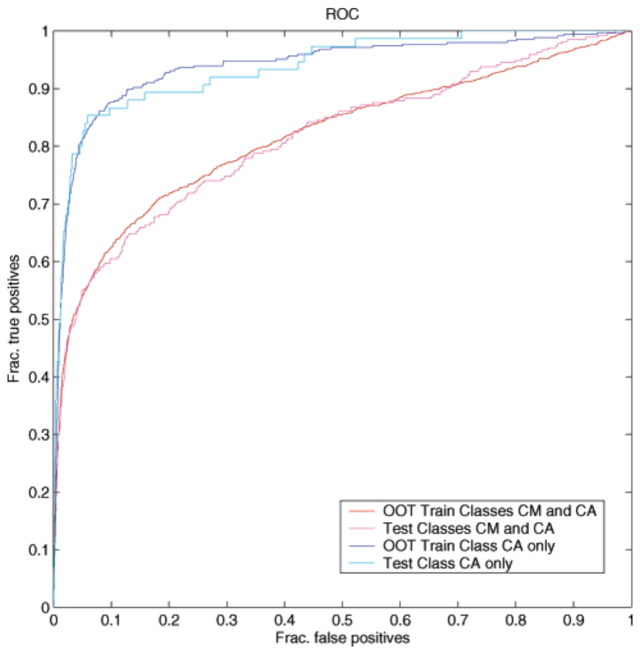

Figure 7. ROC curves for the classifiers constructed on the NCI AIDS antiviral screen data set with ϵ-insensitive absolute loss. The figure displays two pairs of ROC curves. In this computational experiment, we trained an ensemble of molecular graph networks on a data set consisting of three classes of molecules (CI, CM, and CM). To be able to generate ROC curves, we had to reduce the number of classes to two by pooling the molecules of two classes into a single class. The lower pair of ROC curves was obtained by using the ensemble of classifiers to discriminate between CI as one class and CA and CM as the second class, while the upper pair details the ROC curves when using the same ensemble of classifiers to discriminate between CI and CM as one class and the confirmed actives, CA, as the second. The AUCs of the respective pairs of curves are 0.82 and 0.81 for the classification of CI versus CA and CM, respectively, and 0.94 and 0.94 for classification of CI and CM versus CA, respectively.

5. RESULTS AND DISCUSSION

5.1. Classification. Results for the classification experiments on the NCI data set with the classification loss function are given in Figures 7 and 8. Both figures display two pairs of ROC curves. To be able to generate these ROC curves, we had to reduce the number of classes to two by pooling the molecules of two classes into a single class. The lower pair of ROC curves in Figure 7 was obtained by using the ensemble of classifiers to discriminate between CI, on one hand, and CA and CM, on the other, while the upper pair details the ROC curves when using the same ensemble of classifiers to discriminate between CI and CM, on one hand, and the confirmed actives CA, on the other. The remarkable coincidence of the curves obtained by validation of the training part and from the held-out test part of more than 7000 compounds indicates that the validation was performed properly and does not exhibit overfitting. This result is supported by the area under the curves (AUCs) of the respective pairs of curves, which are 0.82 (OOT) and 0.81 (test) for the classification of CI versus CA and CM and

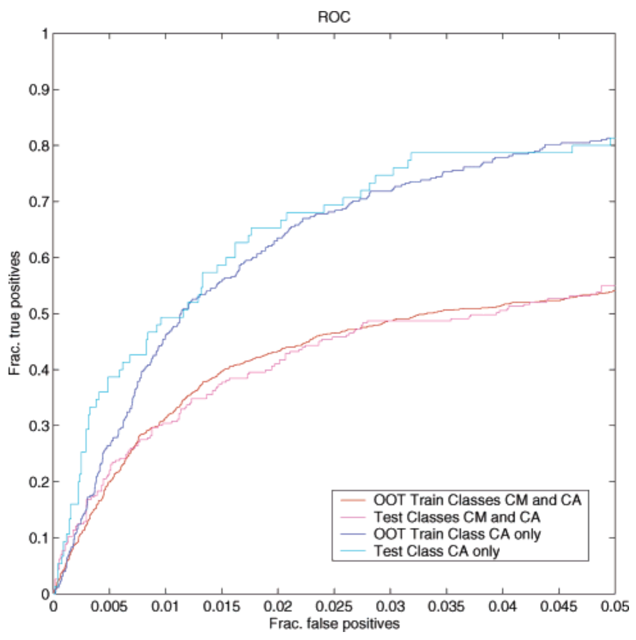

Figure 8. Zoom of the ROC curves given in Figure 7 for false-positive rates up to 5%. Larger false-positive rates usually involve the screening of too many compounds of a large database and are, therefore, of lesser interest in drug discovery or lead optimization. At a false-positive rate of 5%, both validation and test true-positive rates for detecting class CA reach 80%. At a false-positive rate of 1%, both rates exceed 40%, which would result in more than 40% of all actives being found when screening up to this threshold.

0.94 for the classification of both CI and CM versus CA. Figure 8 details the ROC curves for false-positives rates up to 5%. Larger false-positive rates usually involve the screening of too many compounds of a large database and are, therefore, of lesser interest in drug discovery or lead optimization. Again, there is a remarkable coincidence of the curves obtained by validation of the training part and from the held-out test part. At a false-positive rate of 5%, both the validation and test true-positive rates reach 80%. At a false-positive rate of 1%, both rates exceed 40%.

Results for the classification with the logistic loss function are depicted in Figures 9 and 10. The AUCs are 0.80 versus an OOT value of 0.81 for class CI, 0.75 versus an OOT value of 0.75 for class CM, and 0.94 versus an OOT value of 0.91 for class CA, and they fall into the same range as those for the previous approach. Figure 10 depicts the ROC curves for the three ensembles for false-positive rates up to 5%. Larger false-positive rates usually involve the screening of too many compounds of a large database and are, therefore, of lesser interest in drug discovery or lead optimization. At a false-positive rate of 5%, both true-positive rates for detecting active compounds (class CA) approach 75%. At a false-positive rate of 1%, the OOT true-positive rate for class CA exceeds 40%, while the rate on the held-out test set approaches 60%, which corresponds to an enrichment factor of about 38. When interpreting the output of the three classifiers as pseudoprobabilities and assigning the class label of the classifier with the highest output value to each sample, we are able to compute confusion matrices for the OOT validation of the training set and for the held-out test set, given in Tables 1 and 2. While classes CI and CA can be correctly classified in a majority of the cases, samples of class CM are recognized correctly in less than 40% of all cases.

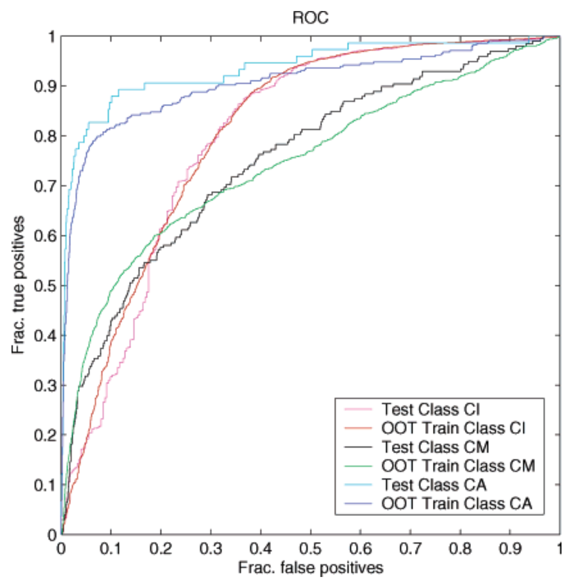

Figure 9. ROC curves for the three classifiers constructed on the NCI AIDS antiviral screen data set using the logistic loss and one-versus-all approaches. The figure displays three pairs of ROC curves. In this computational experiment, three ensembles of molecular graph networks were trained on a data set consisting of three classes of molecules (CI, CM, and CM). The green/black pair of ROC curves corresponds to the ensemble classifier discriminating class CM from the two other classes, the red/magenta pair to class CI against the others. The blue/cyan pair details the ROC curves resulting from the ensemble classifier trained to discriminate class CA against the two remaining classes CI and CM. AUCs are 0.80 versus an OOT value of 0.81 for class CI, 0.75 versus an OOT value of 0.75 for class CM, and 0.94 versus an OOT value of 0.91 for class CA.

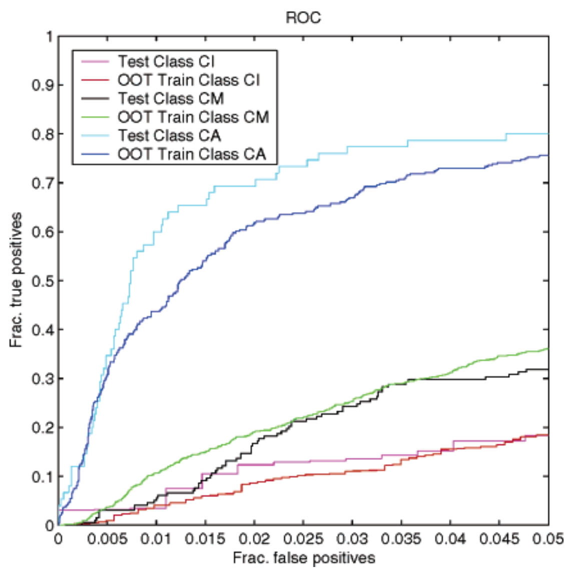

Figure 10. Zoom of the ROC curves given in Figure 9 for false-positive rates up to 5%. At a false-positive rate of 5%, both true-positive rates for detecting class CA approach 75%. At a false-positive rate of 1%, the OOT true-positive rate for class CA exceeds 40%, while the true-positive rate on the held-out test set approaches 60%.

AUC values of the approach employing classification loss and those of the one-versus-all approach are similar to the best results of several variants of a recent classification method based on finding frequent subgraphs[16] (experiments H2 and H3 when omitting class CM from the test set for the ensemble constructed to discriminate CA versus the two other

Table 1. Confusion Matrix for the OOT Validation of the Training Set Obtained by the System of Three Classifiers on the NCI AIDS Antiviral Screen Data Set Using the Logistic Loss and One-Versus-All Approaches[a]

	predicted class		
actual class	CI	CM	CA
CI	0.835	0.126	0.038
CM	0.408	0.380	0.212
CA	0.124	0.187	0.690

[a] The values displayed indicate what fraction of the samples of each class are classified into the respective classes by the constructed classifier. Here, 84% of the samples of class CI are classified correctly, 13% of the samples of class CI are classified wrongly as belonging to class CM, and the remaining 4% are wrongly classified to fall into class CA. While samples of classes CI and CA are mostly classified correctly, class CM (confirmed moderate) are recognized correctly in only 38% of the cases.

Table 2. Confusion Matrix for the Held-Out Test Set Obtained by the System of Three Classifiers on the NCI AIDS Antiviral Screen Data Set Using the Logistic Loss and One-Versus-All Approaches[a]

	predicted class		
actual class	CI	CM	CA
CI	0.852	0.121	0.027
CM	0.444	0.369	0.187
CA	0.093	0.160	0.747

[a] Here, 85% of the samples of class CI are classified correctly, 12% of the samples of class CI are classified wrongly as belonging to class CM, and the remaining 3% are wrongly classified to fall into class CA.

classes). Wilton et al.[17] compare several ranking methods for virtual screening on an older version of the NCI data set. The best performing method there, binary kernel discrimination, is able to locate 12% of all actives (CM and CA pooled) in the first 1% and 35% of all actives in the first 5% of the ranked NCI data set. MGNs trained with logistic loss are able to find 36% and 74% of all actives in the first 1% and 5%, respectively, of the NCI data set ranked according to the output of the ensemble of classifiers. Altogether, both approaches of building ensembles of MGNs exhibit highly competitive classification rates in the held-out test set. Results of the one-against-all approach tend to be slightly better than those of the first approach, which might be caused by the more straightforward way in which the multiclass problem is converted into more easily solved binary classification problems. The universally better results for the held-out part of the NCI data set could be a result of the ensembling. While, for the OOT validation, the output for every sample is computed as an average of only two independent MGNs, for every sample of the test fraction, the average is computed over six MGNs, resulting in an increased ensemble gain and, thus, in an improved prediction.

5.2. Correlation of Feature Net Outputs. We used the three ensembles of classifiers trained by means of the logistic loss function to verify to what extent the outputs z_q of individual feature nets are correlated. Figure 11 depicts a histogram of the absolute value of the pairwise correlation coefficient of the output of two feature nets belonging to the same molecular graph network. We extracted all MGNs from the three ensembles trained with the logistic loss function and computed the output on the held-out test fraction of the NCI data set. We only computed correlation coefficients for pairs of feature nets that belonged to the same MGN and that, thus, were trained on the same output property. The relative frequency seems to be peaked around 0.05 and drops almost linearly from 0.1 to 0.5. Absolute correlation coefficients larger than 0.5 occur very infrequently; those larger than 0.84 occur never. The average absolute value of the 5508 pairwise correlation coefficients is 0.175, and the standard deviation is 0.134. These results indicate that the output of the individual feature nets belonging to the same MGN are generally weakly correlated and can, thus, be seen as complementary descriptors.

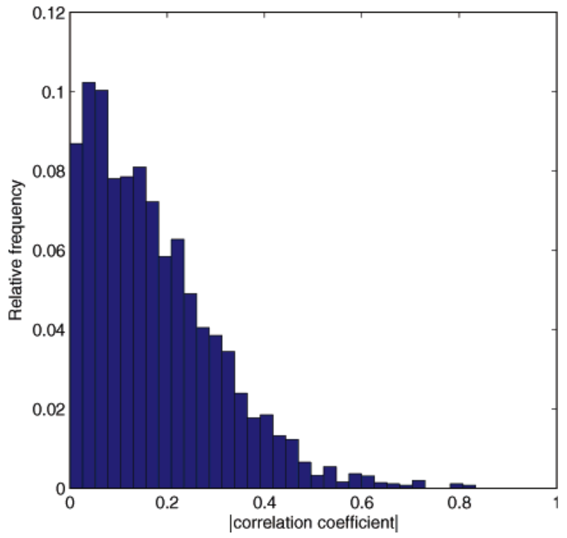

Figure 11. Histogram of the absolute value of the correlation coefficient of the output of two feature nets belonging to the same molecular graph network. The average absolute value of the 5508 correlation coefficients is 0.175; the standard deviation is 0.134.

5.3. Computation Times. All benchmarks were conducted on an standard PC equipped with 512 MB of RAM and an Athlon XP processor running at 1.47 GHz clock speed. The computation time for training and evaluating a MGN depends strongly on the number and iteration depths of its feature nets as well as on the average number of atoms in a molecule and, to a smaller extent, on the loss function employed and the separability of the classification problem. In the two classification settings, the time for training a MGN with 18 or 19 feature nets falls in the range of 0.1−0.3 s per sample of the training set, resulting in a total training time of 1−3 h for a single MGN on the NCI data set. Fortunately, the training of an ensemble of MGN classifiers on different subsets of the training data set can be efficiently parallelized. Evaluating a previously trained MGN takes about 1−3 ms, thus allowing for the performance of a virtual screening of databases of millions of compounds with an ensemble of MGNs within a few hours on a single PC.

6. CONCLUSIONS AND OUTLOOK

In this paper, we presented a method that employs spatio-temporal dynamical systems with local interactions for statistical learning on molecular graphs. The supervised learning problem was formulated as an optimization problem that can be solved by gradient descent techniques. We showed that the gradient can be efficiently computed even for a MGN with a very high number of adjustable parameters. In particular, the stochastic gradient descent algorithm accelerates the training on large data sets considerably. Since the approach is not engineered to perform well on only a certain class of output properties, it can be universally used for predicting biological activity, toxicity, mutagenicity, and physicochemical and other properties. In contrast to approaches that define a distance to known inhibitors in chemical space, the proposed method is flexible and can be retrained if experimental evidence does not coincide with a previously predicted outcome. The approach overcomes the problem of generating putative descriptors as an intermediate step when performing classification and regression on molecular data sets. The applicability to both classification and regression problems is a major strength of the proposed method. Ensemble methods, first used for statistical learning on vector-valued input, are as well able to improve the generalization error of MGNs on unseen data. A disadvantage is that the algorithm is not able to adjust weights for elements or bond types that are not present in the training data set. It would be desirable to induce reasonable default values for those weights either by an expert's knowledge or by using the structure of the periodic table of elements. Another problem for the first-time user might be properly adjusting the network's complexity to the given learning task. A drawback the method shares with conventional neural network approaches is the lack of interpretability and visualizability of the inferred network weights. Nevertheless, the high classification accuracy in combination with the moderate computational demands render the method preferable for the analysis of high-throughput screening data sets and for the virtual screening of large databases.

ACKNOWLEDGMENT

This work is supported by the Deutsche Forschungsgemeinschaft (DFG), Grant LE 491/11-1. We would like to thank all colleagues at the MPI in Saarbrücken and at the Jagiellonian University in Kraków for their support. We especially thank Prof. Maciej Ogorzalek for stimulating discussions and support.

REFERENCES AND NOTES

(1) Lengauer, T.; Lemmen, C.; Rarey, M.; Zimmermann, M. *Drug Discovery Today* **2004**, *9*, 27−34.

(2) Kireev, D. B. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 175−180.

(3) Schädler, K.; Wysotzki, F. *Appl. Intelligence* **1999**, *11*, 15−30.

(4) Merkwirth, C.; Ogorzalek, M.; Wichard, J. Stochastic Gradient Descent Training of Ensembles of DT-CNN Classifiers for Digit Recognition. In *Proceedings of the European Conference on Circuit Theory and Design ECCTD'03*; Kraków, Poland, 2003; Vol. 2.

(5) Merkwirth, C.; Ogorzalek, M.; Wichard, J. Finite Iteration DT-CNN− New Design and Operation Principles. In *Proceedings of the IEEE Int. Symposium on Circuits and Systems*; IEEE: Vancouver, Canada, 2004.

(6) Krogh, A.; Vedelsby, J. Neural Network Ensembles, Cross Validation, and Active Learning. In *Advances in Neural Information Processing Systems*; Tesauro, G., Touretzky, D., Leen, T., Eds.; The MIT Press: Cambridge, MA, 1995; Vol. 7.

(7) LeCun, Y.; Bottou, L.; Orr, G.; Müller, K. Efficient BackProp. In *Neural Networks: Tricks of the Trade*; Orr, G., Müller, K., Eds.; Springer-Verlag: New York, 1998; Vol. 1524.

(8) Bishop, C. M. *Neural Networks for Pattern Recognition*; Oxford University Press: New York, 1995.

(9) Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer-Verlag: New York, 2001.

(10) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. *Proc. IEEE* **1998**, *86*, 2278−2324.

(11) Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In *Proc. IEEE Int. Conf. Neural Networks*; San Francisco, CA, 1993.

(12) Igel, C.; Hüsken, M. Improving the Rprop Learning Algorithm. In *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*; Bothe, H., Rojas, R., Eds.; ICSC Academic Press: Alberta, Canada, 2000.

(13) Merkwirth, C.; Mauser, H.; Schulz-Gasch, T.; Roche, O.; Stahl, M.; Lengauer, T. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1971−1978.

(14) Vapnik, V. *The Nature of Statistical Learning Theory*; Springer-Verlag: New York, 1999.

(15) Weislow, O.; Kiser, R.; Fine, D.; Bader, J.; Shoemaker, R.; Boyd, M. *J. Natl. Cancer Inst.* **1989**, *81*, 577−586.

(16) Deshpande, M.; Kuramochi, M.; Karypis, G. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proceedings of the Third IEEE International Conference on Data Mining ICDM 2003*; Melbourne, Florida, 2003.

(17) Wilton, D.; Willett, P.; Lawson, K.; Mullier, G. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 469−474.

CI049613B