DE GRUYTER OPEN

JAISCR, 2018, Vol. 8, No. 3, pp. 173 – 189
10.1515/jaiscr-2018-0012

SOFT COMPUTING TOOLS FOR VIRTUAL DRUG DISCOVERY

Daniel Hagan[1], Martin Hagan[2]

[1]*Department of Biochemistry, Oklahoma State University, Stillwater, OK, USA*

[2]*Martin Hagan*
School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA

Submitted: 3rd September 2017; accepted: 30th August 2017

Abstract

In this paper, we describe how several soft computing tools can be used to assist in high throughput screening of potential drug candidates. Individual small molecules (ligands) are assessed for their potential to bind to specific proteins (receptors). Committees of multilayer networks are used to classify protein-ligand complexes as good binders or bad binders, based on selected chemical descriptors. The novel aspects of this paper include the use of statistical analyses on the weights of single layer networks to select the appropriate descriptors, the use of Monte Carlo cross-validation to provide confidence measures of network performance (and also to identify problems in the data), the addition of new chemical descriptors to improve network accuracy, and the use of Self Organizing Maps to analyze the performance of the trained network and identify anomalies. We demonstrate the procedures on a large practical data set, and use them to discover a promising characteristic of the data. We also perform virtual screenings with the trained networks on a number of benchmark sets and analyze the results.

Keywords: drug discovery, virtual screening, multilayer network, SOM

1 Introduction

Biochemistry sees drugs as ligands, which are small molecules or ions that bind to specific regions of protein macromolecules. Ligands are used by your cells to control the behavior of proteins, basically like an on-off switch. When ligands bind to their target region (see Figure 1), the protein and ligand change shape to make the tightest fit possible and reach the lowest energy state. If the ligand is an agonist, it induces the active conformation of the protein. If it is an antagonist, it will keep the protein in an inactive state. Protein-ligand interactions are driven by intermolecular forces (ionic bonds, hydrogen bonds, van der Waals (vdW) forces, etc.) as well as entropy and solvent related effects, which create a non-covalent protein-ligand complex. The shape and energy of these complexes are unique for every ligand, and this is what determines their biological functionality. In the lab, biochemists must use an assay to determine the binding affinity of novel ligands. These assays are quite accurate, but also expensive and time consuming.

Computational chemists can generate enormous libraries of drug-like molecules, and need a way to very rapidly screen out those that aren't likely to bind so that any likely binders can be confirmed experimentally. This technique is referred to as high throughput virtual screening (HTVS). Com-

puters can use 3d structures to simulate the physics of protein-ligand interactions and attempt to predict their binding affinities. These programs start with a rigid protein structure and dock ligands into their target region through energy minimization following steepest decent and molecular dynamics calculations. Programs such as Autodock 4 [5] can approach experimental accuracy, but are too computationally intensive for high throughput applications; consider the vast chemical space occupied by drug-like molecules (estimated at 10^{33} unique molecules [6]). Docking programs are now available that take advantage of multicore processors and are able to produce the correct conformation of ligands. Autodock Vina is orders of magnitude faster than its predecessor and yet more accurate in determining binding conformation of candidate ligands [2]. Vina's estimate of affinity, however, is hardly better than a guess.

Neural networks have been used previously in HTVS as a "post-docking" processing step in different ways [1]. Instead of trying to predict affinity, our networks have been trained to classify docked complexes into good binders and bad binders (having greater than 25μM affinity or less). With the rising number and accuracy of crystallographic and NMR complexes in online databases, there is more and more potential for so called knowledge-based functions to accomplish the affinity prediction. These functions are statistical analyses of complexes in the databases. Pairs of atoms that are often found close to each other are considered to be energetically favorable. NNscore [1] is knowledge based and uses atom type pairs and their prevalences as seen in the databases. But it is also empirical, because it characterizes complexes on coulomb energy, vdW forces, number of ligand torsions, and other significant physical descriptors.

In this paper, we extend the work of [1] in several ways. First, we use Monte Carlo cross-validation to provide confidence measures of network performance and to identify problems in the data. In addition, we use Self Organizing Maps to analyze the performance of trained networks and identify anomalies. We were able to use these methods to discover a very interesting characteristic of a commonly used data set that has the potential to significantly increase accuracies of binding predictions. We also added new chemical descriptors to the network inputs, which increased the classification accuracy of the networks.

A general outline of this paper is as follows: In Section 2, we discuss how the training set was compiled and how the inputs are computed. Section 3 covers the architectures of our networks, how irrelevant inputs are removed, how we use the Self Organizing Maps (SOMs), and how we perform our *benchmark* (virtual screen) using the trained networks. (Our networks are trained to classify complexes as good binders or bad binders. In the benchmarking process, we use the same networks to order a new library of compounds from best binder to worst, so that the best binders can be synthesized. To perform the ranking, we use the network output activations. Any activation above 0.5 would be considered a good binder, but a larger activation would indicate that the compound is farther from the decision boundary.) In Section 4, we discuss results from single layer networks and compare with results from multi-layer networks of different sizes. We also show hit histograms of the trained SOM and discuss insights into training data provided by various SOM figures. Finally, we look at the complexes that are difficult to classify and the results from our Benchmarking.

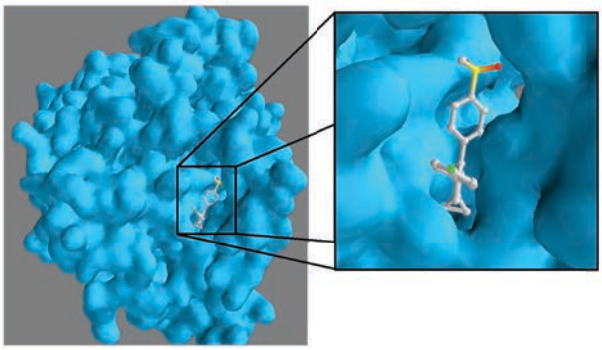

Figure 1. Figure of Protein-ligand Complex [8].

2 Materials

The data we used to develop our neural network classifier was partially comprised of the 3446 protein-ligand complexes of the "refined" set on the PDBbind database [3]. This set was curated from the Protein Data Bank to only include high quality complexes with experimentally determined structures and binding affinity data. Diversity was considered to represent affinities ranging from 10 mM

- 1pM, and excessively redundant proteins and ligands were also removed. Only about 700 complexes had affinities greater than 25μM (weak binders) compared to 2700 less than 25μM (strong binding complexes). This imbalance arises from a general lack of structural data for weak binding complexes. If we train on this data alone, we can expect the networks will reflect the same bias and give more false positives. To provide a more balanced training set, we included the 1431 "weak binding" complexes generated by Durrant et al [1]. These were created by docking the NCI Diversity Set II – a set of 1364 random drug-like ligands – into the 3446 receptors that were already part of the training set, using Autodock Vina. Only those complexes with final energies between 0 and -4 kcal/mol (according to Vina) were used to create this weak binding set. The final training set consisted of 2741 good binders and 2130 bad binders. Structural files for each complex in the PDB refined set were downloaded. For each complex, there is one ligand file in MOL2 format and one receptor file in PDB format. These files need to be in PDBQT format, and some required information must be added. In Figure 2, we have outlined the procedure for preparing the correct PDBQT format from the formats available on the PDBBind database. Starting with the ligands, we used Raccoon [4] and MGLTools v.1.5.4 to add hydrogen atoms, merge non-polar hydrogens with their parent atom, delete lone pairs, assign gasteiger charges, and convert to PDBQT format. For the protein receptors, we used Reduce to add hydrogens, and MGLTools v1.5.4 to merge non-polar hydrogens and assign gasteiger charges. Autodock 4 atom-types were assigned to all atoms.

From the molecular descriptions of the molecules in the data base, we need to compute a limited number of descriptors that can be input to the neural network. Figure 3 shows 244 descriptors that we considered. We decided to start with the same 194 descriptors that were used in [1]. We will train a single layer network and use the weights to gage the contribution each input has toward the final decision. If our first set of inputs train to high accuracy, we can remove those inputs that correspond to the smallest weights (based on a t-test, as will be described later). This process of removing inputs and retraining can be repeated until only the significant descriptors remain without sacrificing accuracy. In Figure 3, all of the colored atoms/atom pairs were removed by this process. Each color represents one round of removing inputs. Green atoms/atom pairs were removed in the first training, light blue in the second, blue in the third, magenta in the fourth, red in the fifth, and orange in the sixth round of training.

The inputs used in the initial training fall into four main categories: proximity counts, electrostatic forces, van der Waals forces, and ligand characteristics. The proximity list counts pairs of atoms between the receptor and ligand within 2$\AA$ of each other. The number is counted separately for each atom type pair. A similar proximity list counts pairs within 4$\AA$. Electrostatic force is a function of proximity, but also of partial atomic charge. This force is calculated for atoms within 4$\AA$ of each other and then summed for each unique atom type pair. Van der Waals forces were approximated using the Leonard-Jones potential

$$V_{LJ}(r) = \frac{4\varepsilon\sigma^{12}}{r^{12}} - \frac{4\varepsilon\sigma^{6}}{r^{6}}. \qquad (1)$$

This force is calculated for atoms within 3$\AA$ - 5.5$\AA$ of each other and summed for each unique atom type pair. Finally, the ligand is characterized by counting the number and type of atoms it contains along with the number of torsions that remain after binding.

We have developed networks based on the 194 inputs that were proposed by Durrant et al [1] to provide a basis for comparison. We have also developed networks using the additional 50 Van der Waals forces. We will compare the performances of these networks on various testing sets and also on several benchmark virtual screens.

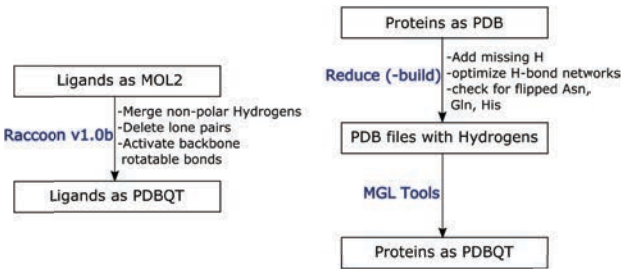

Figure 2. Flowchart showing file processing steps [8].

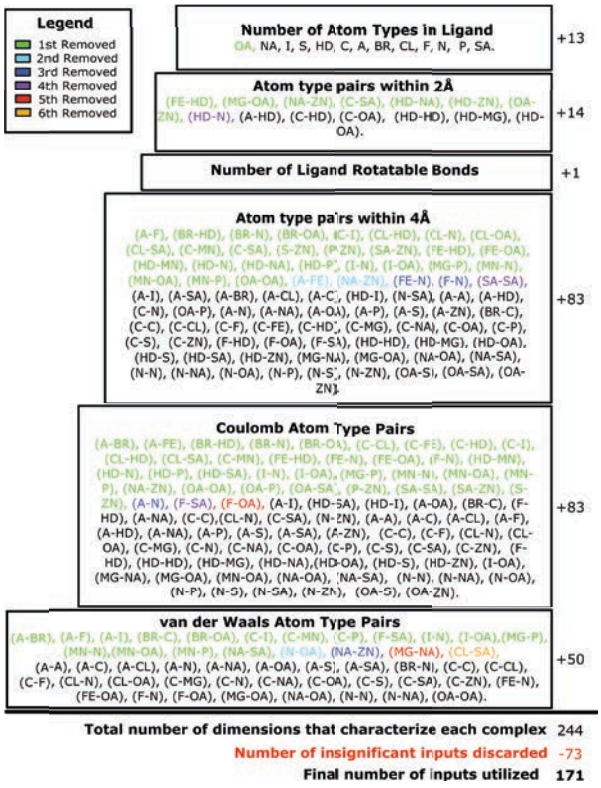

Figure 3. Descriptors used as neural network inputs.

3 Methods

3.1 Feedforward Neural Networks

In order to classify protein-ligand complexes into good binder and bad binder categories, we will use feedforward neural networks. We will use both single layer and two layer networks. As discussed in a later section, the single layer network will be used to determine which inputs (descriptors) are relevant to the classification process. The single layer network also provides a baseline for classification accuracy. After the relevant inputs have been selected, a committee of multilayer networks will be used to make the final classification.

Figure 4 shows the single layer network we will use. It uses a log sigmoid activation function, and is equivalent to logistic regression. The network will be trained to minimize mean square error.

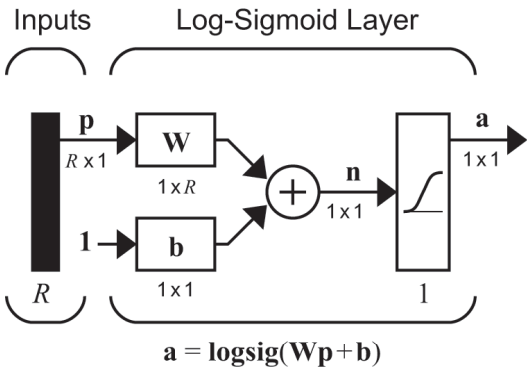

Figure 4. Single Layer Network [8].

Figure 5 shows a two layer network. The hidden layer uses a tan sigmoid activation function, and the output layer uses a softmax activation function. The network will be trained to minimize cross-entropy. The number of neurons in the hidden layer will be adjusted for robust generalization performance. In addition to the two layer network, we also tested networks with three, four and five layers. The hidden layer activations were always of the tan sigmoid type.

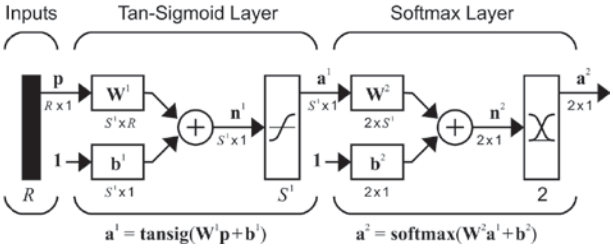

Figure 5. Two Layer Network [8].

For classification purposes, we will use committees of multi-layer networks. Each network in the committee will provide a vote on the classification (good or bad binder), and the class with the most votes will be the final choice. (When we use the networks for benchmarking, we will average the network activations, and use that to rank the complexes. See Section 3.3.)

Both single layer and multi-layer networks will be trained using the scaled conjugate gradient algorithm [9], as implemented in the Neural Network Toolbox for MATLAB [11].

3.1.1 Training, Validation and Test Sets

In order to develop confidence ranges for the performance of the networks, we used Monte Carlo cross-validation (also known as repeated random

subsampling validation). For every Monte Carlo trial, we divided the data into two sets. The first set consisted of a randomly sampled 85% of the data, and the second (test) set consisted of the remaining 15% of the data. For the training of each member of the committee of networks, the first set (85% of the data) was randomly divided again into a training set (70% of the full data set) and a validation set (15% of the full data set). Each committee member is trained to optimize the performance of the network on its assigned training set. The validation set is monitored during training, and training stops when performance on the validation set does not improve for 300 iterations. The weights with the minimum validation error are then used. This early stopping procedure helps avert overfitting. Each member of the committee is thus trained on a somewhat different data set (and with a different set of initial weights). To summarize, on each Monte Carlo trial a committee of multiple feedforward networks are trained. The committee of networks then vote on the test set, and the error is calculated. After all of the Monte Carlo trials are complete, the mean and standard deviation of the test set errors are then computed. Since the test sets are different for each Monte Carlo trial, and since the test sets are not used in any way for training, the distribution of the Monte Carlo errors can be expected to be representative of typical future errors. The procedure is illustrated in Figure 6. The dotted, arrowed lines represent a random sampling process. The solid arrowed lines represent a simple transfer. A different independent test set is used for each Monte Carlo trial. Each test set is used to compute the error for one committee. A different independent validation set is selected for each network in the committee.

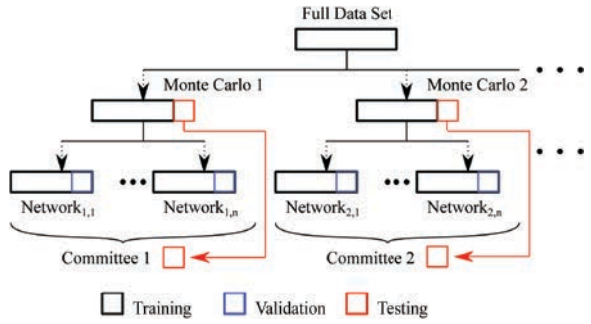

Figure 6. Data Selection Procedure.

3.1.2 Consistently Misclassified Data

We used Monte Carlo cross-validation in a novel way in this work. As described above, each Monte Carlo trial has a committee of networks. Each committee removes a test set that won't be used to train the networks inside that committee. Instead we use the test sets to compute the error for each committee. Then we report the average over all committees and the standard deviation. We are able to show that there is significant variation in network performance over a committee. Furthermore, at the completion of each Monte Carlo trial, we also applied the full data set to the committee of networks, in order to classify every protein-ligand complex (whether in training, validation or testing sets). We then kept track of the number of times each protein-ligand complex is misclassified. We would expect that a given complex would be more likely to be classified correctly when it is included in the training or validation set, and more likely to be misclassified when it is included in the test set. However, if some complexes were misclassified much more often than others, this could indicate problems. If a complex is frequently misclassified, this could indicate that there is a problem with the data, or it could indicate that the complex is in a region of the input space that is poorly sampled, with few neighboring complexes. It could also indicate that the descriptors used in the network input are insufficient to characterize the binding quality of the complex. Later in this paper, we will investigate consistently misclassified data.

3.1.3 Removing Irrelevant Inputs

In addition to the use of Monte Carlo cross-validation to assess the generalization capabilities of the neural network classifiers, we also used it in a novel way to remove irrelevant descriptors from the network inputs. We trained single layer networks, using Monte Carlo cross-validation, as described above. We then considered the distribution of weights across the Monte Carlo trials. Since in a single layer network each weight corresponds to a single input, the distribution of a specific weight can indicate the importance of the corresponding input. If an input (descriptor) is not informative about the binding strength of protein-ligand complexes, then we would expect the weights for that input to vary randomly over the Monte Carlo trials, with a mean

value near zero. Using the average value and sample standard deviation of each weight, we can perform a one-sample t-test to test the hypothesis that the mean is different than zero. After computing the t-statistic, a p-value can be found. If the p-value is below a statistical significance level (we used 0.01), then we would reject the hypothesis that the mean is zero. If it is above the significance level, we cannot reject the hypothesis that the mean is zero, and we will then remove that input. After the inputs are removed, the networks are retrained, and the process is repeated until no more inputs have p-values above 0.01.

3.2 Self Organizing Map

In addition to feedforward networks, we also use the Self Organizing Map (SOM) [12], which is an unsupervised clustering network. The SOM is a topology preserving network, in that neurons within the network have neighbor relationships that are preserved by the training process. We will use the SOM to cluster the training data. This will show us which complexes have similar inputs. We will also be able to visualize the distribution of misclassified complexes, and improve the training of the feedforward networks. (We use the standard SOM described in [12], using Euclidean distance, and train with the batch SOM algorithm.)

3.3 Benchmarking

Virtual screening, known as VS, refers to the computational methodology of evaluating large compound libraries to identify novel ligands or inhibitors. VS can be separated into ligand and structure-based screening. Ligand-based virtual screening finds novel drugs by comparing compounds to known inhibitors or ligands, while structure-based methods dock compounds into a 3-dimensional structure of the target before evaluating their affinity. NNScore [1] and our networks are structure-based scoring functions that evaluate potential ligands after they have been docked into a target receptor.

As described in Section 3.1.1, we can evaluate a network's performance in correctly identifying strong and weak binders by the percent error on an independent test set. In practice, however, the goal of a VS is not to minimize percent error over a large test set, but rather to order a compound library from best binder to worst. This is because only the top scoring ligands will be tested experimentally.

Receiver operating characteristic (ROC) curves can be generated when performing a VS on libraries with known ligands, and the area under these curves (AUROC) provides a better assessment of a virtual screen's performance [15] than percent error. The AUROC corresponds to the chance that a random known ligand will be ranked above a random inactive, and it is independent of the ratio of ligands to inactives in the library being screened. Random selection of strong and weak binders should give an AUROC of 0.5. A perfect screen with no mistakes would have an AUROC of 1, and all true ligands would be ranked above all inactives.

We need to say something about how the AUROC will be computed for a committee of multilayer networks. When computing percent error, the committee decision is obtained by a vote of the individual networks. The target for a good binding complex would be (1,0), but the actual network outputs can range continuously between 0 to 1, with the sum of the two activations always equal to 1 (because of the softmax activation function). A member network vote is considered positive, if the first output neuron activation is larger than the second ($a_1 > a_2$).

In order to rank the protein-ligand complexes, which is needed to compute the AUROC, we will define the score to be the average of the differences between the two activations ($a_1 - a_2$) across all of the committee members. Complexes with a higher score will be considered better binders, since we would expect these binders to be farther from the decision boundary. (This same procedure is used in [1].)

To show how our networks might perform in a real VS, we need a larger set of known active and decoy ligands that have been docked into a diverse set of protein receptors, so that we can calculate area under ROC curves. The diverse set of receptors is required because scoring functions are notoriously system dependent, and performance can vary greatly from receptor to receptor. Following Durrant et al, we decided to use the original Directory of Useful Decoys set of 40 diverse receptors and 2950 ligands with known affinity [14]. For every ligand, the DUD also provides 36 de-

coys. These decoys are chemically similar to the active ligands, but presumed to be inactive. In a real virtual screen, we wouldn't expect compounds to be similar to each other or to known ligands in any way. Instead of using the decoys as inactives, we will use 1528 molecules of the NCI Diversity Set III as presumed inactives. This diverse set of drug-like molecules will be more representative of a typical VS, and is made publicly available at `http://dtp.nci.nih.gov` . Furthermore, since this is the same Benchmarking set used by Durrant, we will be able to compare results and verify that our networks are performing as expected.

All ligands, decoys, and receptors were processed as described in the materials section to add appropriate information. Ligands will be docked into their respective receptors along with the Diversity Set III using AutoDock Vina with default parameters and exhaustiveness 8. The dimensions and centers for the docking boxes were taken from the DUD. Only the top scoring Vina docking was re-scored with our networks.

4 Results and Discussion

4.1 Single Layer Results

We developed two different sets of networks for predicting binding quality. In one set, we used the 194 descriptors originally proposed by Durrant et al [1]. In the second set, we added an additional 50 descriptors based on Van der Waals forces (see Eq. 1). The additional descriptors enabled the networks to achieve lower test set errors, so in the following sections we will describe model development principally for the larger input case. We will be comparing the two sets at various points in the paper. The 194 descriptor case will be referred to as the Coulomb networks, since only coulomb forces were considered, and the 244 input case as the vdW networks, since van der Waals forces were added.

Table 4.1 shows the results of training the single layer network as the descriptors in the input are removed for the vdW case. The average percent error over 100 Monte Carlo trials, on the independent test sets, is provided, along with the sample standard deviation. (We want to emphasize that all of the errors reported in this paper are on test sets that were not used in any way to train the networks that are being tested.) In addition, the table shows the average sensitivity and specificity for each case, along with their corresponding sample standard deviations. The original input contains 244 descriptors. The descriptors with p-values greater than 0.01 were then removed, to produce 195 descriptors. After the simplified network was trained, new p-values were computed, and additional descriptors were removed. This continued until 149 descriptors remained, at which time subsequent p-values remained less than 0.01. To further investigate the importance of the remaining descriptors, we removed additional inputs, approximately 20 at a time, using p-value ranking. It was possible to remove a total of 195 of the original inputs (leaving 49 descriptors) without any significant decrease in performance. (When starting with the 194 descriptors from the Coulomb case, we found that we could reduce to 72 descriptors without degrading performance, although the overall error was smaller for the vdW case, as we will see later.)

Table 1. Single Layer Network Performance

Inputs	Error	STD	Sens	STD	Spec	STD
244	17.1%	1.63	89.8%	1.43	75.5%	3.19
195	17.6%	1.30	89.5%	1.3	75.2%	3.08
174	16.9%	1.55	89.8%	1.29	76.4%	2.70
149	16.8%	1.47	89.8%	1.16	76.0%	2.58
129	16.7%	1.72	89.6%	1.06	77.0%	2.73
109	16.2%	1.63	89.9%	1.04	77.2%	2.10
89	15.8%	1.36	90.0%	0.90	77.9%	1.77
69	15.6%	1.45	89.9%	0.72	78.6%	1.60
49	15.4%	1.30	89.7%	0.87	79.1%	1.07
29	15.7%	1.44	89.2%	0.46	79.0%	0.82
19	15.6%	1.24	89.1%	0.34	79.1%	0.59
14	15.7%	1.49	89.0%	0.40	78.6%	0.49
9	19.1%	1.38	87.7%	0.65	72.6%	0.59
4	22.1%	1.27	86.4%	0.56	66.7%	1.09
2	28.2%	1.56	82.5%	0.81	58.5%	1.25

4.2 Multilayer Results

Table 2 shows the average performance of the 10 member committee of two layer networks with 10 neurons in the hidden layer, using the number of inputs that were found using the single layer network. As we can see, the pattern is very similar to the single layer case.

It should also be noted that in each case there is a significant variation among the 50 Monte Carlo trials. For example, the lowest average percent er-

ror of 14.3% occurs with 49 descriptors in the input vector, but the sample standard deviation of the 50 trials is 1.05, which means we could expect a range of percent errors on any one trail to go from 12.2% to 16.14%. This suggests that with data of the type we have used, it would not be appropriate to report results on a single test set, since performance can vary significantly with small variations in data selection.

Table 2. Two Layer Network Performance

Inputs	Error	STD	Spec	STD	Sens	STD
244	14.4%	1.22	92.6%	0.39	79.9%	0.67
195	14.4%	1.25	92.8%	0.35	79.6%	0.83
174	13.9%	1.38	92.8%	0.38	79.8%	0.78
149	14.4%	1.23	92.8%	0.42	79.5%	0.82
129	14.4%	1.30	92.8%	0.32	79.4%	0.79
109	14.3%	1.35	92.9%	0.28	78.7%	0.80
89	14.3%	1.22	92.9%	0.27	78.4%	0.65
69	14.3%	1.11	93.0%	0.28	78.4%	0.48
49	14.3%	1.05	92.6%	0.27	78.0%	0.45
29	15.3%	1.24	91.5%	0.31	77.9%	0.55
19	14.9%	1.13	91.3%	0.27	77.7%	0.51
14	15.4%	1.23	91.5%	0.27	77.0%	0.62
9	17.2%	1.02	92.2%	0.42	72.0%	0.45
4	21.0%	1.53	86.5%	0.49	69.1%	0.85
2	28.1%	1.56	80.4%	0.8	61.6%	1.05

Figure 7 shows a comparison among the average percent errors of the one layer and two layer networks for both the Couloumb and vdW descriptor sets, as the number of inputs is changed. (Inputs were removed in the order of their p-values.) There are several things that we can take from this figure. First, we can see that the two layer networks do have consistently smaller errors for every set of inputs than the one layer networks, although the amount of reduction is not extremely large. Second, the patterns of the curves are very similar. Even though the p-values that were used to remove the inputs were determined using the single layer network, the results on the two layer network follow the same pattern. This gives us some confidence in the technique we have proposed for removing irrelevant inputs. In addition, the use of the vdW descriptors provides a noticeable reduction in percent error. It is worth emphasizing again the errors reported here are on independent test sets that were not used in any way in training the networks.

At a significance level of 1%, 95 inputs were removed (resulting in 149 remaining inputs). Another 100 inputs were removed (with the next highest p values) without any significant increase in the percent error. This pattern is demonstrated in both the single layer and multi-layer results (and a similar pattern is seen with both Coulomb and vdW descriptor sets), which again provides corroboration for the use of single layer p-values to assist in input removal.

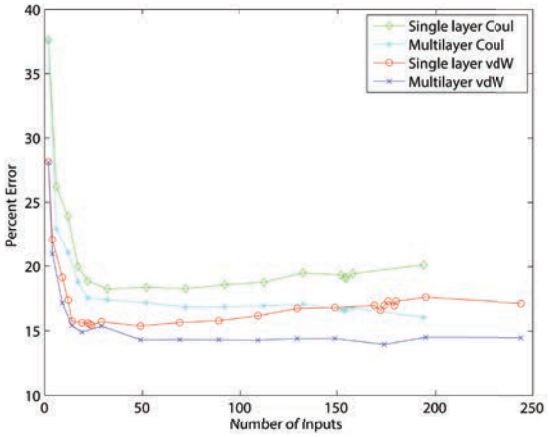

Figure 7. Percent Error Comparison.

Table 3 shows network performance as the number of hidden layer neurons HN) are varied (with 49 inputs to the network). In addition to the two layer network, we tested deep networks with 2, 3 and 4 hidden layers (including many that are not shown in the table). There is a consistent, if not large, difference between the multilayer performance and the single layer performance, but having more than ten hidden neurons provides little additional improvement, and no improvement is seen when using deeper networks. (These results are for the vdW descriptors, but the same patterns were seen for the Coulomb case. The pattern does not depend on the number of inputs to the networks.)

It might be possible that, with a much larger data set, improvements could be seen with deeper networks. With only 4871 data points, it is not possible to make full use of a large network. Most of the data is experimentally measured, using X-ray crystallography, NMR spectroscopy, or cryo-electron microscopy. These techniques are time consuming and expensive, therefore, it is not practical in the near future to obtain an experimental data set of vastly larger size. Perhaps, with a much larger data set, we might find a deeper network that would have improved performance, but we don't

have enough data to demonstrate whether or not that is possible. On the other hand, we might find, with additional data (or additional features), that a better shallow network would be sufficient. Unfortunately, the current data doesn't enable us to say.

Table 3. Adjusting the Number of Hidden Neurons

HN	Error STD	Spec STD	Sens STD
0	15.4% 1.30	89.7% 0.87	79.1% 1.07
2	15.4% 4.34	91.6% 1.53	76.5% 11.0
3	14.9% 1.24	91.6% 0.83	78.1% 1.00
4	14.9% 1.23	91.7% 0.82	77.9% 1.23
5	15.2% 2.19	91.8% 0.93	77.7% 5.64
10	14.7% 1.32	91.6% 0.60	78.7% 1.28
20	14.9% 1.14	91.5% 0.53	79.4% 1.09
40	14.8% 1.36	91.3% 0.55	80.1% 1.04
10,10	14.9% 1.15	91.6% 0.57	78.8% 1.17
10,20	14.8% 1.23	91.5% 0.55	78.9% 1.01
20,10	14.8% 1.20	91.6% 0.52	79.2% 1.16
10,20,10	14.8% 1.24	91.8% 0.58	78.6% 1.16
10,20,10,10	14.9% 1.15	91.8% 0.58	78.5% 1.08

Table 4 shows network performance as the number of committee members are adjusted. Each individual network had one hidden layer with 10 neurons. To compute the error, a vote is taken of the committee members. If 50% or more of the members indicate that the ligand is a good binder, that is the committee decision. As with previous results, the error is always computed on an independent test set that is never used in training any of the committee members. The results show that little improvement is obtained when using more than 10 networks in the committee. Therefore, in the remainder of the paper we will be using committees of 10 two layer networks, with 10 neurons in the hidden layer.

Table 4. Adjusting the Number of Networks in a Commitee

Nets	Error STD	Spec STD	Sens STD
1	14.7% 1.32	91.6% 0.60	78.7% 1.28
5	14.7% 1.27	92.1% 0.30	78.6% 0.61
10	14.3% 1.05	92.6% 0.27	78.0% 0.45
20	14.3% 1.20	92.5% 0.24	78.3% 0.42
30	14.8% 0.98	92.5% 0.23	78.3% 0.32

4.3 Self-Organizing Map Analysis

Approximately 600 of the 4872 data points were always misclassified, whether they were included in the training, validation or testing data sets. We would generally expect that data points would be less likely to be misclassified when they were included in the training set, and more likely to be misclassified when they were in the testing set. This would not be the case, however, if the inputs from different classes were so similar to each other as to be indistinguishable. In order to investigate whether or not this is the case, we will cluster the input vectors to determine their similarities. The clustering method we will use is the Self-Organizing Map (SOM). The SOM is a topology preserving network, in that neurons within the network have neighbor relationships that are preserved by the training process. After training, the inputs from the training set are characterized by a small set of prototype vectors - one for each neuron in the SOM.

To illustrate the process, we will consider the 147 most important input elements from the original 244, as determined by p-values less than 0.01. We will train a 10x10 SOM (100 neurons) with an hexagonal grid to cluster the input vectors. The neuron numbers of the SOM are indicated in Figure 8 inside the circles at the corners of the figure. (Neuron 1 is at the lower left, and numbers increase along each row from left to right, with Neuron 100 at the top right.) After training, we expect that the cluster represented by Neuron 1 will be near the clusters represented by Neurons 2 and 11. This will better enable us to judge the distribution of inputs in the training set. (To validate the SOM operation, we performed a chemical analysis of the clusters and found that the structures of the complexes within a given cluster are indeed similar, and that the chemical structures differ from cluster to cluster.)

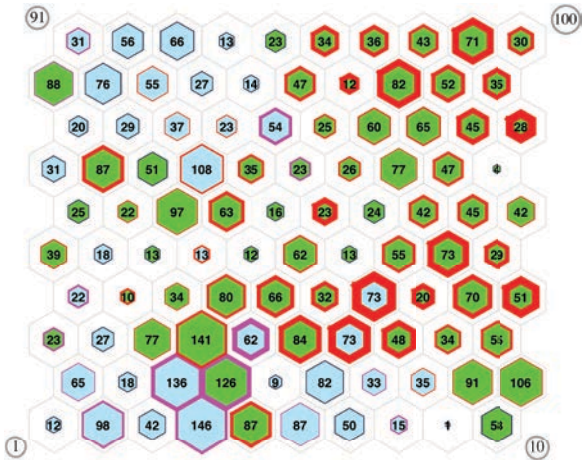

Figure 8. SOM Hits.

Figure 8 is a hit histogram of the trained SOM. The sizes of the hexagons represent the number of input vectors that are in each cluster. Here we can see that Neuron 4 has the largest cluster, with 146 input vectors. The green hexagons represent clusters in which more than 50% of the input vectors are good binders. The light blue hexagons represent clusters in which more than 50% of the input vectors are bad binders.

This figure also illustrates where the consistently misclassified vectors are located. A red edge on a hexagon indicates that there are consistently misclassified vectors in the cluster, and the majority of those vectors are False Positives (bad binders classified as good). A magenta edge on a hexagon indicates False Negatives (good binders that are classified as bad). The thickness of the edge indicates the number of misclassified vectors in the cluster. For example, Neuron 26 has 84 vectors in its cluster; the majority are good binders. It also has a large number of misclassified vectors, and the majority are False Positives (bad binders that are classified as good binders). Comparing the colors of the hexagons with the colors of the edges, we can see that most misclassified inputs are bad binders that are classified as good, and they are on the right side of the feature map, where most good binders are located.

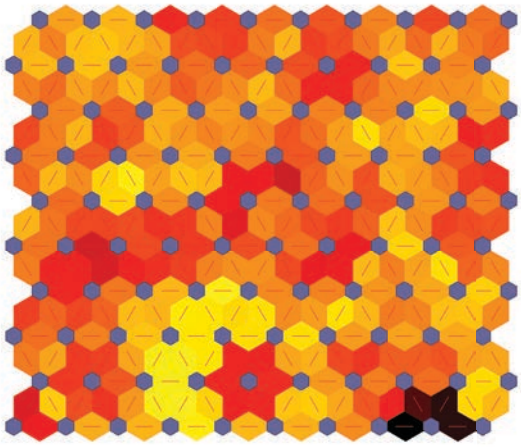

Figure 9. SOM Neighbor Distances.

Some additional information on the SOM can be obtained from the distance matrix, which is shown in Figure 9. The figure indicates distances between cluster centers using a color map. Darker colors indicate larger distances. For example, clusters 4, 5, 13, 14 and 24 have lightly colored connections. That means that these clusters must contain similar complexes. We will have more to say about these clusters in a later section.

There are several things we can say based on an initial reading of this figure. The good binders generally fall on the right side of the SOM, and the bad binders fall on the left side. This tells us that the 147 inputs that we are using are able to characterize the binding properties of the protein-ligand complex, at least in a general way, and this is what enables an accuracy of approximately 80%. However, there are a number of clusters in which there are many input vectors that are consistently misclassified. Clusters 27, 37, 40, 49 and 80, in particular, have a high percentage of misclassified vectors. The highest percentages of misclassified vectors occur in clusters that contain primarily good binders, and the errors are generally False Postives (bad binders that are misclassified as good binders). Does this mean that the existing inputs are not sufficient to distinguish the binding ability of certain types of protein-ligand complexes?

As discussed earlier, there are two data sets that were combined to produce our training set. The first set consists of the 3446 protein-ligand complexes of the “refined” set of the PDBbind database [3], which contains experimentally measured affinities. Because this set contains a disproportionate number of good binders, it was suplemented with the 1431 “weak binding” complexes generated by Durrant et al [1]. This data was not developed experimentally, but through docking software. In order to understand how the networks performed on these two components of the training set, we analyzed the SOM clusters to determine if the two components of the data set occupied similar regions of the input space. (For reference, we will refer to data from the first component as PDB, and data from the second component as DOCKED.) Figure 10 indicates where the two components of the data set were located. The sizes of the hexagons are the same as those in Figure 8 and indicate the number of inputs in each cluster. The grey scale in Figure 10 indicates the percentage of DOCKED data in the cluster. Black hexagons contain 100% DOCKED data, and white hexagons contain 100% PDB data.

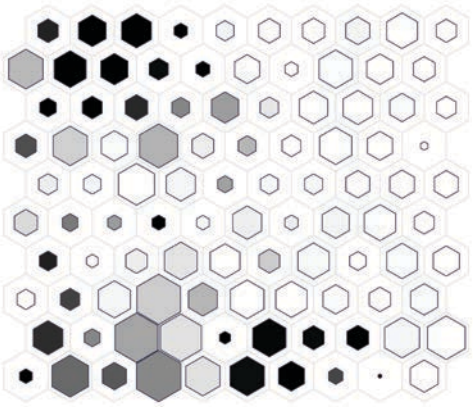

Figure 10. PDB (white) vs DOCKED (black).

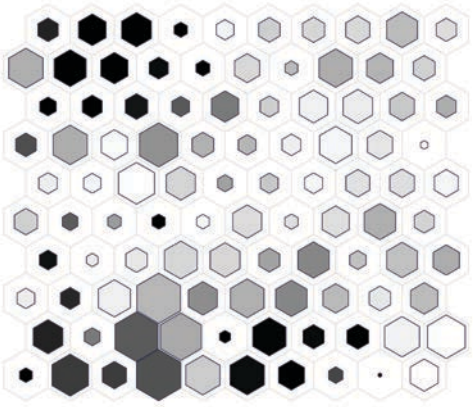

Figure 11. Good binder (white) vs bad binder (black).

Figure 11 is a companion to Figure 10. It indicates the percentage of bad binders in each cluster. Black hexagons contain 100% bad binders, and white hexagons contain 100% good binders. By comparing these two figures with Figure 8, we can make several observations. First, the SOM clustering is better able to separate PDB and DOCKED data than it can separate good binders and bad binders, since the upper right side of Figure 10 is almost all white, which indicates PDB data. From Figure 11, we can see that there are bad binders within the PDB data, but they occur in different parts of the input space than the bind binders of the DOCKED set. This is indicated by the fact that the shading in the left side of both figures is very similar, indicating that the bad binders in this part of the SOM come primarily from the DOCKED set.

If we compare Figures 11 and 10 with Figure 8, we can see that most of the errors occur on the right side of the SOM. These errors are False Positives (bad binders classified as good) that are in the PDB set. The bad binders from the DOCKED set are generally classified correctly. Based on the SOM analysis, the DOCKED data that were added to the PDB set, in order to fortify the number of bad binders there, do not appear to have the same type of bad binders that are in the PDB set. For this reason, they are not as helpful in improving the performance on the bad binders in the PDB set. It would improve performance to have additional bad binders in the DOCKED set that had more similarity to the bad binders in the PDB set.

There are some bad binders in the PDB set that were correctly classified. For example, consider cluster 30. From Figure 10 we can see that cluster 30 consists almost entirely of PDB data. From Figure 11 we can see that it does have a significant number of bad binders, and Figure 8 indicates that most of the inputs in that cluster were correctly classified.

To summarize, most of the errors are bad binders in the PDB set that are classified as good binders. These errors occur mainly in the upper right quadrant of the SOM. Based on the SOM clustering, those bad binders look more like the good binders in the PDB set than the bad binders in the DOCKED set. This suggests that for future development we should attempt to develop additional bad binders that look more like the bad binders in the PDB set. We can use the SOM to assist in that process – as we generate additional bad binders, we can check them against existing SOM clusters to see if they are close enough to be helpful.

Most of the complexes that are being misclassified are misclassified whether they are included within the training set or not. Based on this result, and the SOM analysis, it appears that we need to use additional descriptors in order to improve the classification abilities of the network. It is not a matter of increasing the numbers of neurons or layers in the networks.

4.4 Analysis of Misclassified Data

As described earlier, a Monte Carlo analysis of the training process was performed. For example, the best results were obtained for a committee of 20 networks, where each network had two layers with

five neurons in the hidden layer. (For this analysis we are using the Coulomb data, but the vdW data showed the same pattern.) During the training process, the network committees were generated 100 times, with different training/validation and test sets, and the mean and standard deviations of errors on the 100 different committees were computed. In addition, we counted the number of times that each ligand-protein complex in the entire data set (training, validation and testing) was misclassified over the 100 Monte Carlo trials. One would expect that a typical complex would be less likely to be misclassified, if it was included in the training (or validation) set, and more likely to be misclassified, if it was included in the test set (which is not used to update weights or stop training). At each Monte Carlo trial, 15% of the data is included in the test set. If a given complex was always misclassified when it was included in the test set, and never misclassified when it was included in the training or validation sets, then we would expect it to be misclassified, on average, 15 times out of the 100 Monte Carlo Trials. Of course, this would be an extreme result. If the network is being trained effectively, and if the descriptors in the network inputs are sufficiently explanatory, we would expect the number of misclassifications to be much less.

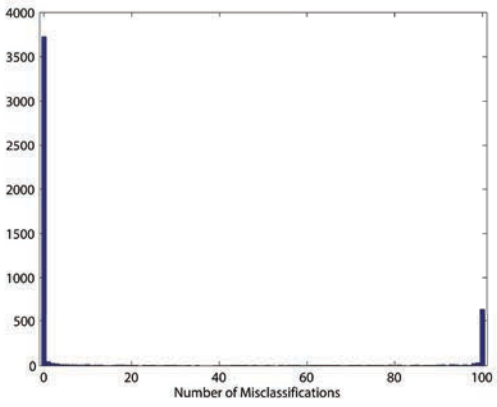

Figure 12. Histogram of the Number of Misclassifications of Each Complex [8].

Figure 12 is a histogram of the number of misclassifications that occur for each of the 4871 complexes. Notice that over 600 of the complexes are misclassified in every Monte Carlo trial (100 times). This result was very unexpected. This means that even when these complexes are in the training set, they are misclassified every time. As illustrated in Figure 8, the consistently misclassified complexes consist of both good and bad binders. In addition, they must be distributed throughout the input space, since they appear in almost every cluster in the SOM network. This suggests that there must be good binders that look like bad binders and bad binders that look like good binders. This raises the question as to whether these consistently misclassified complexes have anything in common.

To test this, we divided the data into two sets. One set contained all complexes that were misclassified more than 50% of the time, and the other set contained all complexes that were misclassified less than 50% of the time. The two data sets were then trained separately. We would, of course, expect that the complexes that were usually correctly classified in the full data set would also be correctly classified in the smaller data set, where the original misclassified complexes were removed. We would not, however, expect that the complexes that were always misclassified should now be accurately classified, when the correctly classified complexes were removed. There is no reason, a priori, to expect that the misclassified complexes have anything in particular in common with each other. If the consistently misclassified complexes were simply very difficult to classify, then they should still be difficult to classify when the correctly classified complexes are removed. (For example, if the difficult data consisted of random labels, it would produce percent errors of 50%, which could not be improved.)

There were 4871 complexes in the original data set. Of these, 2741 were good binders and 2130 were bad. We found 854 complexes that were misclassified more than 50% of the time. Of these, 246 were good binders and 608 were bad. There were 4017 complexes that were correctly classified more than 50% of the time. Of these, 2495 were good, and 1522 were bad. This means that bad binders were more likely to be misclassified than good binders, but significant numbers of both bad and good binders were contained in both the misclassified data set and the correctly classified data set.

Using just the single layer network, we trained both the well-classified complexes and the misclassified complexes. For the well-classified complexes, we found, as expected, that the accuracy improved. The mean percent error over 100

Monte Carlo trials was 6.23%. This is much less than the approximately 20% error on the full data set (Coulomb data). The average sensitivity was 94.5%, and the average specificity was 93.9%. Both of these numbers are significantly higher than the results on the full data set.

For the misclassified data set, the results were unexpected. We found that the average percent error was 16.3%. This is less than the approximately 20% error on the full data set. The average sensitivity was 92.6%, and the average specificity was 71.6%. The specificity is approximately the same as the results for the full data set and the sensitivity is higher. Remember that the error on these complexes was close to 100% when they were included in the full data set.

It appears that there is some consistent difference between the complexes that were consistently misclassified and those that were not. To investigate this further, we inspected the weights of the single layer network to see how they differed between the two data sets. First, consider Figure 13. This is a plot of the weights for the single layer network, with 144 inputs, that was trained on the complexes that were misclassified less than 50% of the time when trained on the full data set. There are three curves in the figure. The blue curve is the mean of the weights for the 100 Monte Carlo runs. The red line is two standard deviations above the mean, and the green line is two standard deviations below the mean. Almost all of the Monte Carlo runs fell between the red and green lines. The data was very consistent when these 144 inputs were used. (For space reasons, the plot of the weights for the complete data set is not shown, but the shape of that plot was almost identical to the originally well classified case, although the weights were much smaller – with a maximum of about 3, instead of 80.)

Now, consider Figure 14. These are the weights for the single layer network that was trained only on the complexes that were misclassified more than 50% of the time when trained on the full data set. These weights follow a very different pattern than the other case. The weight pattern shown in Figure 13 is able to correctly classify the majority of the complexes in the original data set. However, there is a set of 600 to 800 complexes that seem to require a different type of classifier, represented by the weights in Figure 14. In certain regions of the weight matrix, the weights for the originally misclassified data are approximately the negative of the weights for the well classified data, but this does not hold completely.

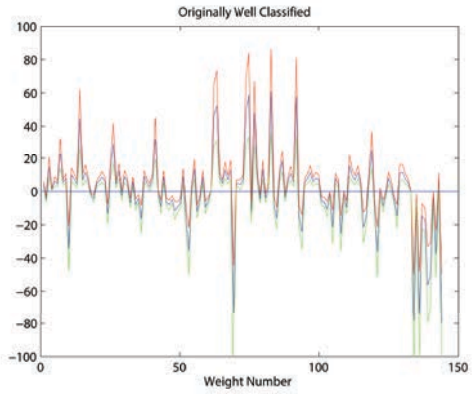

Figure 13. Weights when trained only on well-classified data [8].

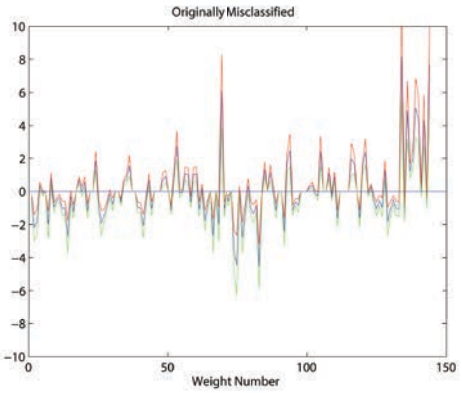

Figure 14. Weights when trained only on misclassified data [8].

Figure 15 shows a scatter plot of the weights from the full set versus the weights from the misclassified set. The best fit linear equation is $-0.44x + 0.34$, with an R value of 0.8. There is a generally negative relationship, but there is a significant amount of variation. There are some inputs that have the largest weights in the complete data set (also the well classified), but small weights in the originally misclassified. In the future, we will investigate those descriptors that are important for the originally well classified data, but apparently not consequential for the originally misclassified data.

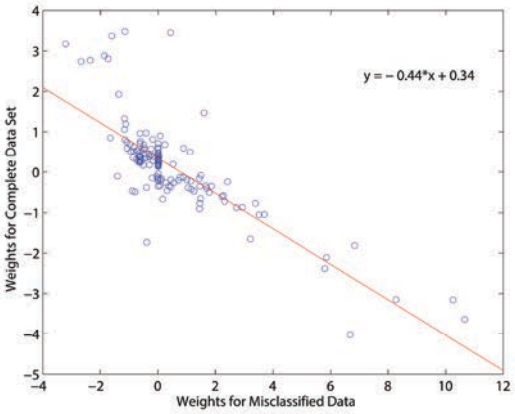

Figure 15. Weights from full set versus weights from misclassified set [8].

4.5 Benchmarking

In Section 3.3, we noted that the objective of developing the neural network models is not necessarily to minimize the percent classification error on a test set of potential ligand-protein complexes, but rather to order a library of compounds from best binder to worst binder, so that the best binders can be synthesized and tested experimentally. This process is called a Virtual Screen (VS). To see how our networks might perform in a VS, we used the 40 benchmark sets from Durrant et al [1], as described in Section 3.3.

Since the objective of a VS is only to find the strongest binding ligands, using percent error over the entire benchmark set as a measure of performance is not particularly informative. It was demonstrated in [15] that the Area Under the Receiver Operating Characteristic Curve (AUROC) provides a better assessment of the performance of a VS than percent error. An AUROC value of 0.5 would represent a scoring method that was no better than guessing. An AUROC value of 1.0 would represent a perfect scoring method that always ranked the strongest binding ligand first, and ranked all true ligands higher than all inactives.

Table 5 provides a comparison of AUROC values for several different screening methods on the 40 benchmark sets described in Durrant et al [1]. The first column is the abbreviation for the receptor involved in the screen; the second column gives the AUROC value for the scoring function described in [1] (NNScore); the third column gives the value for the single layer network trained with the vdW descriptors (SL vdW); the fourth column gives the value for the committee of multilayer networks trained with only the Coulomb descriptors (ML Coul); and the final column shows the results for the committee of multilayer networks trained with the vdW descriptors (ML vdW). The last row gives the average AUROC values over all 40 benchmark sets.

Table 5. AUROC on benchmarks

Protein	NNScore	SL vdW	ML Coul	ML vdW
ACE	0.6190	0.6297	0.7132	0.6202
ACHE	0.9135	0.7986	0.8088	0.8088
ALR2	0.7269	0.6092	0.6148	0.6148
AMPC	0.5955	0.8561	0.8743	0.8743
AR	0.7821	0.4589	0.6235	0.6235
CDK2	0.5922	0.7138	0.6486	0.6773
COMT	0.5021	0.4753	0.4794	0.4620
COX1	0.6826	0.6077	0.6907	0.6282
COX2	0.7784	0.8591	0.7996	0.8736
DHFR	0.6161	0.7916	0.7522	0.7477
EGFR	0.7592	0.8080	0.7742	0.7644
ERAG	0.8460	0.5823	0.7977	0.6319
ERAN	0.9401	0.9273	0.9647	0.9444
FGFR1	0.8226	0.9331	0.9344	0.9213
FXA	0.9110	0.9632	0.9471	0.9622
GART	0.6173	0.8920	0.9199	0.8499
GPB	0.3845	0.8010	0.8752	0.7071
GR	0.8825	0.5174	0.7151	0.5663
HIVPR	0.9541	0.9866	0.9878	0.9889
HIVRT	0.5825	0.4526	0.4668	0.4496
HGMA	0.5136	0.4907	0.5051	0.5099
HSP90	0.8156	0.8367	0.7585	0.7879
INHA	0.8156	0.8308	0.8099	0.8253
MR	0.8428	0.3606	0.7110	0.4120
NA	0.7170	0.7464	0.8204	0.7243
P38	0.8372	0.8630	0.7981	0.8652
PARP	0.4504	0.3435	0.2956	0.3794
PDB	0.7349	0.8986	0.7263	0.8805
PDE5	0.8675	0.8375	0.8617	0.8545
PDGFR	0.7102	0.6526	0.5518	0.6629
PNP	0.6506	0.6232	0.5724	0.5604
PPAR	0.8872	0.9555	0.9122	0.9619
PR	0.7524	0.4010	0.5393	0.4215
RXR	0.8581	0.7492	0.8617	0.7764
SAHH	0.6822	0.7861	0.7621	0.7299
SRC	0.7521	0.8753	0.8400	0.8593
THROM	0.8546	0.9510	0.9423	0.9442
TK	0.4587	0.4951	0.5192	0.4582
TRYPS	0.9095	0.9744	0.9825	0.9712
VEGFR	0.7206	0.8321	0.7769	0.8140
ADA	0.4350	0.5933	0.6463	0.5354
Average	0.7262	0.7259	0.7459	0.7232

Figure 16. SOM Showing clustering of all benchmarks.

There are several conclusions we can make from this study. First, the highlighted row represents AUROC values for the training data set, which consisted of complexes from the Protein Data Bank, along with selected well-docked bad binders. For this data, the addition of the vdW descriptors produced significantly higher AUROC values. This is reasonable, since the vdW descriptors also produced smaller percent error on the PDB/DOCKED testing sets, as shown in Figure 7. However, if we look at the overall average in the final row, all of the methods produced very similar results. Even the single layer network did as well as the multilayer networks. In fact, although it was left out of the table for reasons of space, the single layer Coulomb network performed as well as any of the others. This result is consistent with Durrant's benchmark [1]. Each scoring method performs best on different receptors, but the average over all receptors is around 70% - 75%.

What explains why even the simplest network performs as well as the more complex methods on these 40 benchmarks? By adding additional descriptors, and increasing the number of layers and the number of networks in the committees, we were able to produce improved test set errors on the combined PDB/DOCKED data sets. But this did not translate into improved AUROC values on the benchmarks.

To help understand this result, we returned to the SOM analysis from Section 4.3. Using the SOM that was trained on the PDB/DOCKED data, we located the data from all of the benchmark sets in the original clusters. The results are shown in Figure 16, where the sizes of the blue hexagons indicate the number of benchmark complexes in each cluster. We can see that the data from all of the benchmark sets fit almost entirely in 6 clusters. Interestingly, many of them fall in the neighboring clusters 4, 13, 14 and 24. From Figures 8, 10 and 11, we know that 4 and 13 are mainly bad binders from the DOCKED set, while 14 and 24 have a few more good binders from the PDB set. These clusters seem to straddle a boundary between good and bad binders and also between the PDB and DOCKED sets. The other two clusters (62 and 64) where the benchmark data are also located seem to fall on a similar boundary, with both cells containing a mixture of PDB/DOCKED and good/bad binders. It is worth noting that while many clusters contained very few complexes, there were none that contained zero.

It is clear that the majority of benchmarks only sample a very small region of the space spanned by the training/testing set. Developing improved networks on the training/testing set will not necessarily result in improved virtual screens on the benchmark set.

5 Conclusions

In this paper, we have described how soft computing can be used in the virtual screening of potential drug candidates. There are several novel aspects of our research. We have used a statistical analysis of the weights of single layer networks to select relevant inputs, we have used Monte Carlo cross-validation to provide confidence measures of network performance and to identify problems in network training, and we have used Self Organizing Maps to investigate the training results and identify anomalies. We applied these techniques to a large practical data set of protein-ligand complexes. We were able to classify the complexes into good binding and bad binding categories with average accuracies of more than 85%, which is better than previously reported results. More importantly, by using an analysis of the Monte Carlo cross-validation and Self Organizing Map results, we were able to discover a characteristic of the data set that has promise for producing even more accurate results. By dividing the data into two separate categories,

and training different network committees on each category, we were able to achieve average accuracies of more than 90%. This division of the data into the two categories cannot be achieved using the current descriptors. Future research will look for biochemical differences between the complexes in the two categories.

We also used the trained neural network committees to compute virtual screens on 40 standard benchmark sets. Although our network committees, using new chemical descriptors, achieved improved accuracies on the test sets, they did not produce significantly better results on the virtual screens. Using an SOM analysis, we were able to find that all of the benchmark data were located in a small portion of the full input space that was sampled by the training/testing sets. It will be important in the future to either concentrate training data in regions where the virtual screens are to be performed, or to have benchmark sets that are more representative of the range of complexes for which the networks will be used.

This work has also demonstrated the value of using SOM analysis as part of any virtual screening process. Because there are a limited amount of experimental data available for ligand-protein interactions, additional training data must be obtained with docking software. The SOM can be used to determine if data obtained from docking are located in a useful region of the input space and, therefore, increase the efficiency of the training process.

6 Acknowledgements

The authors would like to thank Dr. Jacob Durrant of the University of California, San Diego, for supplying us with data on well docked bad binders and for helpful suggestions on this work.

References

[1] J. D. Durrant and J. A. McCammon, NNScore: A Neural-Network-Based Scoring Function for the Characterization of Protein/Ligand Complexes, J. Chem. Inf. Model, 50, 2010, 1865-1871.

[2] Oleg Trott and Arthur J. Olson, AutoDock Vina: Improving the speed and accuracy of docking wih a new scoring function, efficient optimization, and multithreading, J. Computational Chemistry, 31, 2009, 455-461.

[3] R. Wang and X. Fang and Y. Lu and S. Wang, The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures, J. Med. Chem, 47, 2004, 2977-2980.

[4] Stefano Forli, Raccoon—AutoDock VS: an automated tool for preparing AutoDock virtual screenings, http://autodock.scripps.edu/resources/raccoon, Accessed: 2016-01-10.

[5] G. M. Morris and R. Huey and W. Lindstrom and M. F. Sanner and R. K. Belew and D. S. Goodsell and A. J. Olson, Autodock4 and AutoDockTools4: automated docking with selective receptor flexibility, J. Computational Chemistry, 16, 2009, 2785-2791.

[6] P. G. Polishchuk and T. I. Madzhidov and A. Varnek, Estimation of the size of drug-like chemical space based on GDB-17 data, J. Computer Aided Molecule Design, 8, 2013, 675-679.

[7] Guo-Bo Li and Ling-Ling Yang and Wen-Jing Wang and Lin-Li Li and Sheng-Yong Yang, ID-Score: A New Empirical Scoring Function Based on a Comprehensive Set of Descriptors Related to ProteinLigand Interactions, J. Chem. Inf. Modeling, 53, 2013, 592-600.

[8] Daniel M. Hagan, and Martin T. Hagan, Virtual drug screening using neural networks, International Joint Conference on Neural Networks (IJCNN), pp. 579-587. IEEE, 2016.

[9] Martin Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks, 6, 1993, 525-533.

[10] M. T. Hagan and H. B. Demuth and M. H. Beale, Neural Network Design, PWS, 1996.

[11] H. B. Demuth and M. H. Beale and M. T. Hagan, The Neural Network Toolbox for MATLAB, The MathWorks, 2014.

[12] Kohonen, T., The self-organizing map, Proceedings of the IEEE, 78, 1990, 1464-1480.

[13] T. J. Cheng, and X. Li, and Y. Li, and Z. H. Liu, and R. X. Wang, Comparative assessment of scoring functions on a diverse test set, J. Chem. Inf. Modeling, 49, 2009, 1079-1093.

[14] Shoichet Huang and J. Irwin, Benchmarking Sets for Molecular Docking, Journal of Med. Chemistry, 49, 2006, 6789-7801.

[15] N. Triballeau, F. Archer, I. Brabet, J. P. Pin and H. O. Bertrand, Virtual screening workflow development guided by the receiver operating characteristic curve approach. Application to high-throughput docking on metabotropic glutamate receptor subtype 4, Journal of Med. Chemistry, 48, 2005, 2534-2547.

Daniel Hagan is currently with the department of Biochemistry at Oklahoma State University. His research interests include pharmacology, drug discovery, neural networks, machine learning, and cognitive science. He also enjoys playing tennis and plays violin with the OSU Symphony Orchestra.

Martin Hagan received the PhD degree in electrical engineering from the University of Kansas.
Currently, he is Professor of Electrical and Computer Engineering at Oklahoma State University. He is the author, with H. Demuth and M. Beale, of the textbook Neural Network Design. He is also a coauthor of the Neural Network Toolbox for MATLAB. His research interests focus on the theory and application of neural networks and other soft computing technologies.