JOURNAL OF CHEMICAL INFORMATION AND MODELING

ARTICLE

pubs.acs.org/jcim

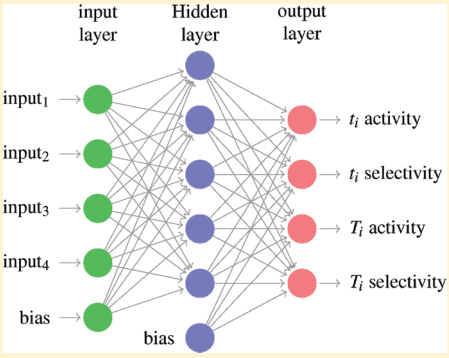
Improved Machine Learning Models for Predicting Selective Compounds

Xia Ning,[†] Michael Walters,[‡] and George Karypisxy*,[†]

[†]Department of Computer Science & Engineering and [‡]College of Pharmacy, University of Minnesota, Twin Cities, Minneapolis, Minnesota 55455, United States

S Supporting Information

ABSTRACT: The identification of small potent compounds that selectively bind to the target under consideration with high affinities is a critical step toward successful drug discovery. However, there is still a lack of efficient and accurate computational methods to predict compound selectivity properties. In this paper, we propose a set of machine learning methods to do compound selectivity prediction. In particular, we propose a novel cascaded learning method and a multitask learning method. The cascaded method decomposes the selectivity prediction into two steps, one model for each step, so as to effectively filter out nonselective compounds. The multitask method incorporates both activity and selectivity models into one multitask model so as to better differentiate compound selectivity properties. We conducted a comprehensive set of experiments and compared the results with those of other conventional selectivity prediction methods, and our results demonstrated that the cascaded and multitask methods significantly improve the selectivity prediction performance.



1. INTRODUCTION

Small molecular drug discovery is a time-consuming and costly process in which the identification of potential drug candidates serves as an initial and critical step. A successful drug needs to exhibit at least two important properties. The first is that the compound has to bind with high affinity to the protein (this protein is referred to as the *target*) that it is designed to affect so as to act efficaciously. The second is that the compound has to bind with high affinity to only that protein so as to minimize the likelihood of undesirable side effects. The latter property is related to compound *selectivity*, which measures how differentially a compound binds to the protein of interest.

Experimental determination of compound selectivity usually takes place during the later stages of the drug discovery process. A selectivity test can include binding assays or clinical trials.[1] The problem with such an approach is that it defers selectivity assessment to the later stages, so if it fails, then significant investments in time and resources get wasted. For this reason, it is highly desirable to have inexpensive and accurate computational methods to predict compound selectivity at earlier stages in the drug discovery process.

The use of computational methods to predict properties of chemical compounds has a long history in chemical informatics. The work pioneered by Hansch et al.[2,3] led to the development of computational methods for predicting structure−activity relationships (SARs). In recent years, researchers have started to develop similar approaches for building models to predict the selectivity properties of compounds. Such models are referred to as structure−selectivity relationship (SSR) models.[4] Existing computational methods for building SSR models fall into two general classes. The first contains methods that determine selectivity by using SSR models, and the second contains methods that build a selectivity model by considering only the target of interest. The disadvantage of the first class of methods is that they rely on models learned by not utilizing information about which of the active compounds are selective and which ones are nonselective. As such, they ignore key information that can potentially lead to an overall better selectivity prediction method. The disadvantage of the second class of methods is that they largely ignore a rich source of information from multiple other proteins, which if properly explored could lead to more realistic and accurate selectivity models.

In this paper, we develop two classes of machine learning methods for building SSR models. The first class of methods, referred to as cascaded SSRs, builds on previously developed techniques and incorporates a pair of models on two levels. Level 1 is a standard SSR model which identifies the compounds that bind to the target regardless of their selectivity. Level 2 is a model that further screens the compounds identified by the level 1 model to identify only the subset that binds selectively to the target and not to the other proteins. Such methods exhibit a cascaded architecture, and by decoupling the requirements of accuracy and selectivity, the respective learning tasks are more focused and easier to learn so as to increase the likelihood of developing accurate models. The second class of methods, referred to as multitask SSRs, incorporates information from

Received: July 25, 2011
Published: November 22, 2011

multiple targets and multiple prediction tasks and builds a multitask SSR model. The key insight is that compound activity/selectivity properties for other proteins can be utilized when building an SSR model for the target of interest. These methods treat activity and selectivity prediction as two different yet related tasks. For the target of interest and multiple other proteins, their SAR and SSR tasks are tied together into one single multitask model. During model training, the SAR and SSR tasks are learned simultaneously with useful information implicitly transferred across one another, and the compound selectivity against multiple proteins is better captured within the model.

We conducted a comprehensive set of experiments to assess the performance of these methods and compare them with other previously proposed state-of-the-art methods. A unique feature of our evaluation is that, unlike previous studies that utilized a very small number of test sets, we constructed data sets derived from publicly available resources (ChEMBL, http://www.ebi.ac.uk/chembl/) that collectively contained 135 individual SSR prediction tasks. Our experimental evaluations show that the proposed methods outperform those developed previously and that the approach based on multitask learning performs substantially better than all the other approaches.

The rest of the paper is organized as follows. In section 2, a brief literature review on the work related to both SSR prediction and multitask learning is provided. In section 3 definitions and notations are given. In section 4, different learning methods for SSR prediction are presented. In section 5, materials used by the study are presented. In section 6, the results for the selectivity study are presented. Finally, in section 7, the conclusions are given.

2. RELATED WORK

2.1. Structure-Selectivity Relationship. Developing computational methods to aid in the identification of selective compounds has recently been recognized as an important step in lead optimization, and several studies have shown the promise of utilizing machine learning approaches toward this goal. Vogt et al.[5] investigated approaches for identifying selective compounds on the basis of how similar they are to known selective compounds (similarity-search-based approach). They tested five widely used 2D fingerprints for compound representation, and their results demonstrated that 2D fingerprints are capable of identifying compounds which have different selectivity properties against closely related target proteins. Stumpfe et al.[6] developed two approaches that they referred to as *single-step* and *dual-step* approaches. The single-step approach builds the SSR model by utilizing only the selective compounds (one class classification). The two-step approach uses a pair of classifiers that are applied in sequence. The first is a binary classifier trained on selective compounds (positive class) and nonselective active compounds (negative class), whereas the second classifier is the one-class classifier as used in the single-step approach. A compound is considered to be selective if both classifiers predicted it as such. For both approaches, they used both *k*-nearest-neighbor (similarity search) and Bayesian methods in building the models and represented the compounds using MACCS and Molprint2D descriptors. Their experimental results demonstrated that both of these approaches are able to identify selective compounds. Wassermann et al.[7,8] built on this work and investigated the use of support vector machines (SVMs)[9] as the underlying machine learning framework for learning SSR models. Specifically, they investigated four types of SSR models. The first is a binary classifier that uses selective compounds as positive instances and inactive compounds as negative instances. The second is a set of three one-vs-rest binary classifiers whose positive classes correspond to the selective, nonselective active, and inactive compounds and whose negative class corresponds to the compounds that did not belong to the positive classes. The third is a two-step approach in which the model of the first step uses active compounds as positive instances and inactive compounds as negative instances (i.e., a standard SAR model) and the model of the second step uses selective compounds as positive instances and nonselective active compounds as negative instances. Finally, the fourth is a preference ranking model that incorporates pairwise constraints that rank the selective compounds higher than the inactive compounds and the inactive compounds higher than the nonselective compounds (i.e., selectives > inactives > nonselectives). Their results showed that SVM-based methods outperformed conventional similarity search methods and that the ranking and one-versus-rest methods performed similarly to each other and outperformed the other SVM-based methods.

2.2. Multitask Learning (MTL). Multitask learning[10,11] is a transfer learning mechanism designed to improve the generalization performance of a given model by leveraging the domain-specific information contained in the training signals of related tasks. In multitask learning, multiple related tasks are represented by a common representation, and then they are learned in parallel, such that information from one task can be transferred to another task through their common representations or shared learning steps so as to boost that task's learning performance. A very intuitive multitask model utilizes back-propagation neural networks (NNs).[12] Input to the back-propagation net is the common representations of all related tasks. For each task to be learned through the net, there is one output from the net. A hidden layer is shared across all the tasks such that by back-propagation all the tasks can learn task-related/target-specific signals from other tasks through the shared hidden layer. Within such a net, all the tasks can be learned simultaneously, and by leveraging knowledge from other related tasks, each task can be better learned than only from its own training instances. In recent years, many sophisticated multitask learning methods have emerged, which include kernel methods,[13] Gaussian processes,[14] task clustering,[10] Bayesian models,[15] matrix regularization,[16] etc. Various studies have reported promising results with the use of multitask learning in diverse areas such as cheminformatics,[17,18] face recognition,[19] and text mining.[20]

3. DEFINITIONS AND NOTATIONS

In this paper, the protein targets and the compounds will be denoted by lower-case t and c characters, respectively, and subscripts will be used to denote specific targets and compounds. Similarly, sets of protein targets or compounds will be denoted by upper-case T and C characters, respectively.

The activity of a compound will be determined by its IC_{50} value (i.e., the concentration of the compound that is required for 50% inhibition of the target under consideration, and lower IC_{50} values indicate higher activity, http://www.ncgc.nih.gov/guidance/section3.html). A compound will be considered to be *active* for a given target if its IC_{50} value for that target is less than 1 μM. For each target t_i, its sets of experimentally determined active and inactive compounds will be denoted by C_i^+ and C_i^-, respectively, whereas the union of the two sets will be denoted by C_i.

A compound c will be *selective* for t_i against a set of targets T_i if the following two conditions are satisfied:

$$\text{(i)} \quad c \text{ is active for } t_i$$
$$\text{(ii)} \quad \min_{\forall t_j \in T_i} \frac{IC_{50}(c, t_j)}{IC_{50}(c, t_i)} \geq 50 \tag{1}$$

This definition follows the common practice of using the ratio of binding affinities in determining the selectivity of compounds.[21] Note that c can be either active or inactive for some or all of the targets in T_i while being selective for t_i.

An important aspect of the selectivity definition is that it is done by taking into account both the target under consideration (t_i) and also another set of targets (T_i) against which a compound's selectivity for t_i is defined. We will refer to T_i as the *challenge set*. Depending on the problem at hand, each target may have multiple challenge sets, and they will be denoted using subscripts, i.e., $T_{i,1}$, $T_{i,2}$, ..., $T_{i,n}$. In such cases, a compound's selectivity properties for a target can be different against different challenge sets. Given a target t_i and a challenge set T_i, t_i's selective compounds against T_i will be denoted by $S_i^+(T_i)$, whereas the remaining nonselective active compounds will be denoted by $S_i^-(T_i)$. This notation will be simplified to S_i^+ and S_i^- when a single challenge set is considered.

Given a target t_i and a challenge set T_i, the goal of the SSR model is to predict whether a compound is selective for t_i against all the targets in T_i. We will refer to target t_i as the *target of interest*.

4. METHODS

The methods that we developed for building SSR models are based on machine learning techniques. Within the context of these methods, there are two approaches that can be used to build SSR models. The first approach is for target t_i and each target $t_j \in T_i$ to build a regression model for predicting the binding affinity of a compound (e.g., IC_{50}) for that target. Then a compound c will be predicted as selective if the two conditions of eq 1 selectivity are satisfied by the predicted binding affinities. The second approach is to build a classification model that is designed to directly predict whether a compound is selective for t_i without first predicting the compound's binding affinities.

Even though the available training data (i.e., compounds with known binding affinities and their labels according to eq 1) can support both of these approaches, the methods developed and presented in this work are based on the second approach. Specifically, we developed methods that employ neural networks as the underlying machine learning mechanism and determine the selectivity of a compound by building different types of binary or multiclass classification models.

4.1. Baseline SSR Models. Given a target t_i and a challenge set T_i, the compounds for which the activity information with respect to t_i is known belong to one of three sets: S_i^+, S_i^-, and C_i^-. From these sets, three different SSR classification models can potentially be learned using (i) S_i^+ vs C_i^-, (ii) S_i^+ vs S_i^-, and (iii) S_i^+ vs $S_i^- \cup C_i^-$. These models share the same positive class (first set of compounds, i.e., S_i^+) but differ in the compounds that they use to define the negative class (second set of compounds).

The first model (i.e., built using S_i^+ as positive training instances and C_i^- as negative training instances), due to ignoring the nonselective active compounds (S_i^-) during training, can potentially learn a model that differentiates between actives and inactives (i.e., C_i^+ vs C_i^-) since C_i^- may dominate during training, irrespective of whether the active compounds are selective. The second model (i.e., built using S_i^+ as positive training instances and S_i^- as negative training instances), due to ignoring the inactive compounds (C_i^-) during training, can potentially learn a model that predicts as selective compounds that may not even be active against the target under consideration. For these reasons, we did not investigate these models any further but instead used the third model to define a baseline SSR model that will be denoted by SSR_{base}.

The SSR_{base} method constructs the SSR model by treating both the inactive and nonselective active compounds as negative training instances, thus allowing it to focus on the selective active compounds while taking into account the other two groups of compounds. A potential limitation of this model is that, depending on the relative size of the S_i^- and C_i^- sets, the model learned may be more influenced by one set of compounds. In particular, since in most cases $|C_i^-| > |S_i^-|$, the resulting model may have characteristics similar to those of the model learned using only C_i^- as the negative class. To overcome this problem, we applied an undersampling technique; that is, while constructing the negative class, an equal number of compounds from S_i^- and C_i^- were randomly selected. The total number of compounds that are selected to form the negative class was set to be equal to the number of compounds in the positive class ($|S_i^+|$).

4.2. Cascaded SSR Models. SSR_{base} described in section 4.1 tries to build a model that can achieve two things at the same time: learn which compounds are both active and selective. This is significantly harder than trying to learn a single thing at a time, and as such, it may lead to poor classification performance. To address this shortcoming, we developed a *cascaded* SSR model that takes into account all the compounds (selectives, nonselectives, and inactives) and builds models such that each model is designed to learn one single task.

For a target t_i and a challenge set T_i, the cascaded SSR model consists of two levels. The model on level 1 is a normal SAR model that tries to differentiate between active and inactive compounds, and the model on level 2 is a model that tries to differentiate between selective and nonselective compounds. The level 1 model serves as a filter for the level 2 model so as to filter out those compounds that are not likely to be even active. During prediction, compounds are first classified by the level 1 model, and only those compounds whose prediction values are above a certain threshold, referred to as the "minactivity" threshold, go through the level 2 SSR model. Only compounds classified as positive by the level 2 SSR model will be considered as selective. This two-level cascaded SSR model is refereed to as SSR_c.

The level 1 model is trained using C_i^+ and C_i^- as positive and negative training instances, respectively, and is identical to t_i's SAR model. The level 2 model can be trained using S_i^+ and S_i^- as positive and negative training instances, respectively, as it will be used to classify compounds that were predicted as active by the level 1 model. However, the overall performance of the SSR_c model can potentially be improved if the SSR_{base} model described in section 4.1 is used as the level 2 model. This is because the SSR_{base} model also takes into account the inactive compounds while learning to identify selective compounds, and as such, it can be used as an additional filter to eliminate inactive compounds that were predicted incorrectly by the level 1 model.

Note that even though the cascaded SSR_c model is similar in spirit to the two-step approach proposed by Wassermann et al.,[7] it differs in two important ways. First, instead of sending a

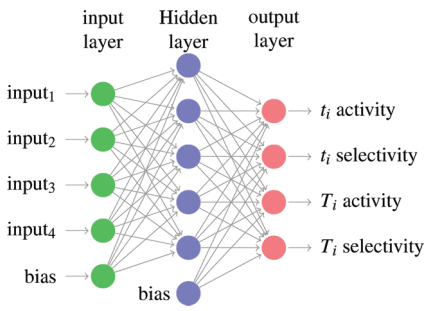


Figure 1. Multitask neural network for target t_i and challenge set T_i.

constant number of the highest ranked compounds (as predicted by the level 1 model) to the level 2 model, SSR_c uses the *minactivity* threshold to determine the compounds that will be *routed* to the level 2 model. Second, instead of using only the S_i^- compounds as the negative class of the level 2 model, SSR_c uses the compounds in $S_i^- \cup C_i^-$ as the corresponding negative class. As the experiments presented in section 6.4 show, this change leads to better performance.

4.3. Multitask SSR Models. Both the baseline and the cascaded SSR models take into account the labels of the training compounds (i.e., selective, nonselective, active, and inactive) as they were determined for the target under consideration (t_i). However, important information can also be derived by taking into account their labels as they were determined for the targets in the challenge set (T_i). For example, if a compound c is active for t_i (i.e., $IC_{50}(c,t_i) < 1\ \mu M$) and it is inactive for all the targets in T_i (i.e., $\forall\ t_j \in T_i$, $IC_{50}(c,t_j) < 1\ \mu M$), then there is a higher probability that the compound is also selective for t_i since $\forall\ t_j \in T_i$, $IC_{50}(c,t_j)/IC_{50}(c,t_i)$ is already greater than 1 (though not necessarily greater than 50 so as to be determined as selective; see the definition of selectivity in eq 1). Similarly, if a compound is selective for one target in T_i, then by definition this compound is nonselective for t_i. This indicates that the selectivity of a certain compound can be more accurately determined by considering its activity properties against other targets.

Motivated by this observation, we developed another model that, in addition to the activity and selectivity information for t_i, also incorporates the activity and selectivity information for the targets in the challenge set T_i. This additional information is typically not available for the compounds whose selectivity needs to be determined but also needs to be predicted in the course of predicting the compounds' selectivity. Since this model relies on models built to predict related tasks, it falls under the general class of multitask learning models, and we will refer to this model as SSR_{mt}.

The SSR_{mt} model extends the model used by the baseline SSR model (section 4.1) by learning compound activity and compound selectivity together. It incorporates these two different learning tasks into a single model so as to facilitate transfer of information during the training of the different models. The learning with information transfer is done by using the neural network model shown in Figure 1, which has two pairs of outputs. The first pair corresponds to the activity and selectivity for t_i, whereas the second pair corresponds to the activity and selectivity for T_i (the compound selectivity for each target $t_j \in T_i$ was determined using $\{t_i\}$ as the challenge set). The inputs to this neural network are the various features that describe the chemical structures of the compounds. Each training compound has four labels (one for each output), and during training, the various model parameters are estimated to minimize a mean-square-error (MSE) loss function (described in section 5.3) between the predicted and actual four labels at the output layer. The prediction of a compound whose selectivity for t_i needs to be determined is given by the output associated with t_i's selectivity. This model utilizes the same hidden layer to simultaneously learn how to predict the four different tasks (i.e., activity and selectivity for t_i and T_i), and as such, it can facilitate better information transfer across the different tasks during the model's training stage.

Note that the four labels for each training instance are not independent. For example, if selectivity for t_i is positive (i.e., selective for t_i), then selectivity for any other t_j has to be negative (i.e., a compound cannot be selective for two targets under consideration). Also if activity for t_i is negative, then selectivity for t_i has to be negative (selective compounds have to be active first). We do not explicitly model such dependencies through loss function but rely on the NN system and the learning process to implicitly incorporate such constraints from training instances.

4.4. Three-Way SSR Models. The performance of the SSR models described in the previous sections was also compared against the performance of another type of model that has been proposed in the past. This model is the three-way classification approach developed by Wassermann et al.[7] that operates as follows. For target t_i and its challenge set T_i, it builds three one-vs-rest binary classification models for each one of the selective (S_i^+), nonselective (S_i^-), and inactive (C_i^-) sets of compounds. During model training, since $|S_i^+| < |C_i^-| + |S_i^-|$ and $|S_i^-| < |C_i^-| + |S_i^+|$, the binary models for S_i^+ and S_i^- may be dominated by the majority class (i.e., C_i^-). To deal with this class imbalance and ito not lose any of the available information, we randomly oversampled the minority class so as to make their training instances the same counts as the majority class. During prediction, a compound c is predicted by each one of the three models, leading to three predicted values, $f_{S_i^+}(c)$, $f_{S_i^-}(c)$, and $f_{C_i^-}(c)$. A compound is considered to be selective if $f_{S_i^+}(c) = \max(f_{S_i^+}(c), f_{S_i^-}(c), f_{C_i^-}(c))$. Also, if a degree of selectivity is required (i.e., to rank a set of predicted compounds), then c's degree of selectivity is given by $f_{S_i^+}(c) - \max(f_{S_i^-}(c, f_{C_i^-}(c)))$. The underlying idea of this three-way classification method is to model different classes separately and to decide the class of a new instance on the basis of how differently it is classified by different models. We will denote this SSR model as SSR_{3way}.

4.5. Cross-SAR SSR Models. Another model was motivated by approaches used within the pharmaceutical industry in which the selectivity of a compound c_i against target t_i is determined by comparing the output on c_i from t_i's SAR model against that of the SAR model for each one of the targets in T_i. Specifically, if $f_{t_i}(c_i)$ is the prediction of t_i's SAR model on c_i and $\{f_{t_j}(c_i) | t_j \in T_i\}$ are the predictions of the SAR models for the targets in T_i on c_i, then the extent to which c_i is selective for t_i against T_i is given by $f_{t_i}(c_i) - \max_{t_j}(f_{t_j}(c_i))$. We will denote this SSR model as SSR_{xSAR}.

4.6. Three-Class SSR Models. Compound selectivity prediction can also be viewed as a multiclass classification problem, in which each compound c_i has three binary class labels, that is, selectivity, nonselectivity, and inactivity against a target t_i. For each target t_i, a three-class classifier is built from its own compounds. Then a compound is predicted by such a multiclass model as one of the three classes. Figure 2 shows a three-class neural network. The difference between the three-class neural network classifier and the multitask neural network classifier as in Figure 2 is that the compound activity label against the challenge set T_i is not included.

5. MATERIALS

5.1. Data Sets. We evaluated the performance of the various SSR models on a set of protein targets and their ligands that are extracted from ChEMBL, which is a database of molecular targets and their published assays with bioactive druglike small molecules. We first selected an initial set of molecular targets and their corresponding ligands from ChEMBL on the basis of the following criteria:

- The target is a single protein.
- The assay for the target is a binding assay.
- For each target t_i, there are at least 20 active compounds.

These criteria ensure that the binding affinities measure how well a compound binds to a single target and also there are sufficient compounds to learn a model. From this initial set of targets, we eliminated those targets that satisfied any of the following criteria:

- The target does not share any of its active compounds with other targets in the initial set of targets.
- The target has less than 10 selective compounds against any single target in the initial set.

The first condition eliminates targets for which we cannot assess whether their active compounds are selective, whereas the second condition is designed to keep the targets that contain a sufficient number of selective compounds to learn an SSR model. These filtering steps resulted in a data set with 98 protein targets. For each of these 98 targets t_i, we used all of t_i's known active compounds and generated an equal-size set of inactive compounds as follows. If t_i had more inactive compounds than active compounds, the desired number of compounds were randomly selected among them. If t_i had fewer inactive compounds than active compounds, then all of its inactive compounds were selected and the rest of the compounds were selected randomly from the compounds in ChEMBL that show extremely low binding affinities for any of the targets in our data set. Note that, in the second case, the selection procedure may introduce some false negatives. However, since the selection is fully random, the false-negative rate is expected to be low.

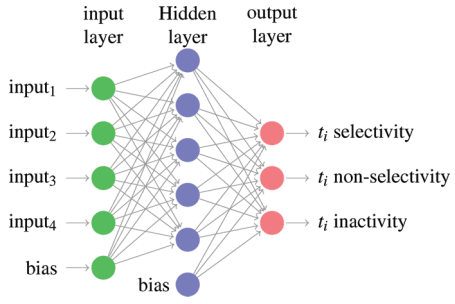


Figure 2. Three-class neural network for target t_i.

Figure 3 shows the distribution of the active compounds with respect to the number of targets that they are active for. Most of the compounds are active for a small number of targets, and only less than 5% of the compounds are active for more than 10 targets.

Using these 98 targets, we constructed two data sets for experimental testing. The first data set, referred to as DS1, contains 116 individual SSR prediction tasks involving a single target t_i as the target of interest and another single target t_j as its challenge set (i.e., $T_i = \{t_j\}$). These 116 SSR prediction tasks were identified by considering all possible (i.e., 98 × 97) SSR prediction tasks of this type and then selecting only those for which (i) targets t_i and t_j have some common active compounds (i.e., those compounds that are active for both t_i and t_j) and (ii) when t_j is used as the sole member of t_i's challenge set, the resulting SSR prediction task results in at least 10 selective compounds for t_i. Both of these filtering steps are essential to ensure that there are a sufficiently large number of training compounds to accurately learn and assess the selectivity of the target of interest. In these 116 SSR prediction tasks, the average numbers of active and selective compounds for the target of interest are 172 and 26, respectively. Note that each target t_i can potentially be the target of interest in multiple SSR prediction tasks and that a compound c may have different selectivity properties for t_i when different t_i targets are considered. The second data set, referred to as DS2, contains 19 individual SSR prediction tasks involving a single target t_i as the target of interest and multiple targets in its challenge set T_i. The 19 prediction tasks were identified according to the criteria that (i) target t_i and each $t_j \in T_i$ share common active compounds, (ii) $|T_i| \geq 2$, and (iii) there are at least 10 selective compounds for t_i against T_i determined on the bais of eq 1. These criteria result in on average 3.9 targets in each challenge set, and the average numbers of active and selective compounds for the target of interest are 198 and 27, respectively.

The first data set is constructed so as to maximize the number of selective compounds for each t_i to train a reliable model. This is also a common practice in other selectivity learning and data set construction exercises[7,22] and in real experimental settings. Meanwhile, it maximizes the number of interested targets to test for any statistically significant conclusions. The second data set is constructed to test the generalizability of SSR models. Additional details on the targets, compounds, and the two data sets are available at http://www-users.cs.umn.edu/~xning/selectivity/. Figure 4 shows DS1 as well as DS2.

5.2. Compound Representations. We generated 2048 bit binary Chemaxon compound descriptors (http://www.chemaxon.com/) for all the compounds extracted as described in section 5.1. Then we applied a principal component analysis (PCA) based dimension reduction method such that the 2048 dimensions were reduced to 1000 dimensions. Each compound is then represented by such a 1000-dimension feature vector, and thus, an NN with 1000 input nodes can be trained on such compound representations.

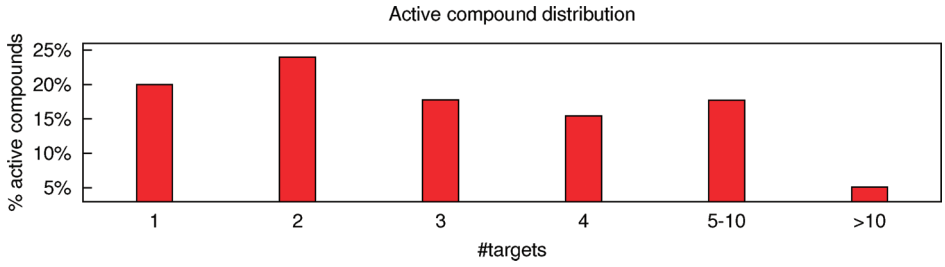


Figure 3. Compound distribution with respect to the number of targets they are active against.

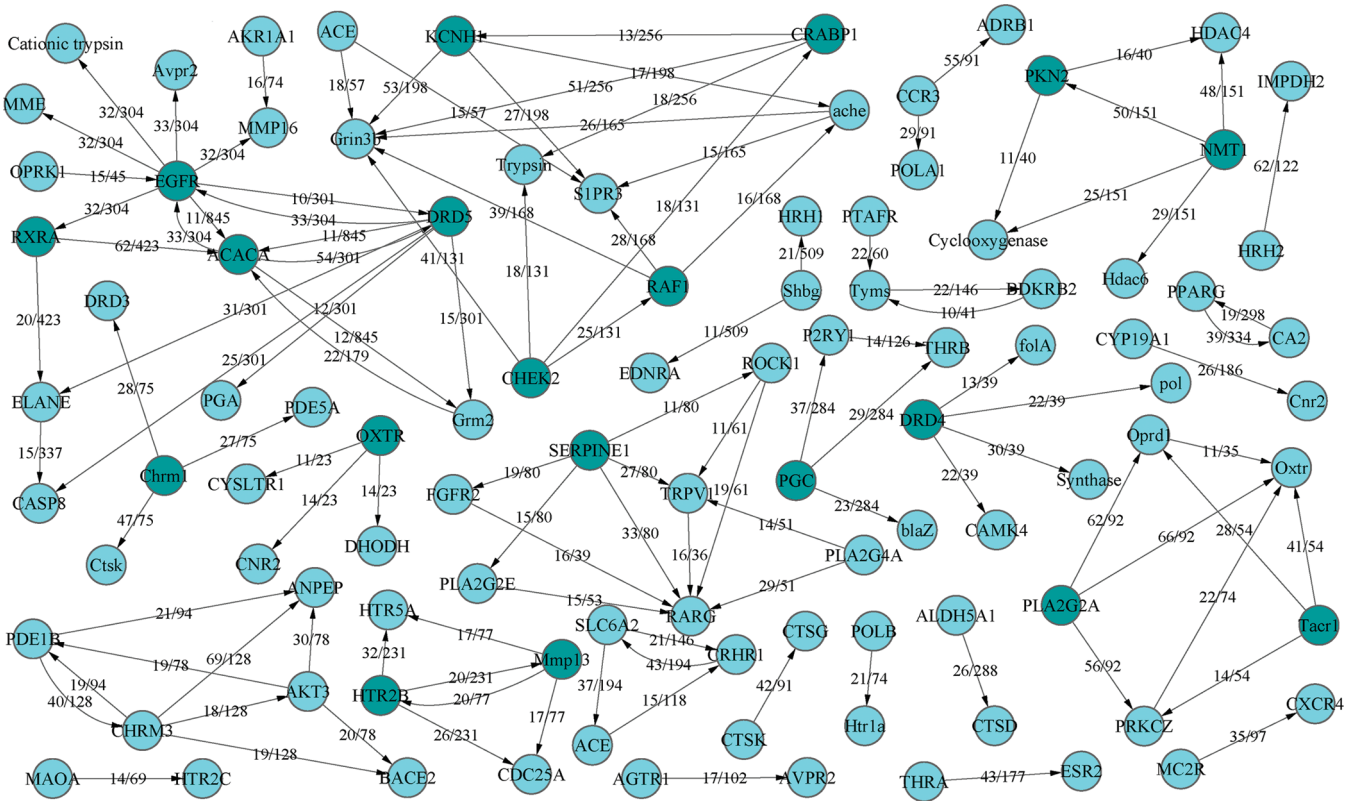


Figure 4. Data set: The nodes in the graph represent the targets in DS1. The directed edge from target A to target B with a label x/y represents that target A has y active compounds, out of which x compounds are selective for target A against target B. The dark nodes represent the targets that are selected as targets of interest into DS2.

We used the Chemaxon software to generate initial descriptors and a Matlab dimension reduction toolbox (http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html) with a PCA option to reduce descriptor dimensions.

Note that chemical compounds can be represented by different fingerprints.[23] However, since our study does not aim to evaluate the performance of different fingerprints for compound selectivity, we only applied Chemaxon compound descriptors because it is one of the most popular choices. Dimensionality reduction is performed since the NN may suffer from the curse of dimensionality[24] if high-dimension inputs are encountered.

5.3. Neural Networks. We used the publicly available neural network software FANN (http://leenissen.dk/fann/) for our neural network implementation. FANN implements fast multilayer artificial neural networks with support for both fully connected and sparsely connected networks. We used the sigmoid function as the squash function on hidden and output neurons, which is defined as follows:

$$\sigma(y_j) = \frac{1}{1 + e^{-sy_j}} \tag{2}$$

where y_j is the output at a certain hidden/output neuron j and s is a steepness parameter that determines how aggressive the nonlinear transform is. The output of each neuron is calculated as

$$y_j = \sum_{i=1}^{n} w_{ij}x_{ij} + \theta_k \tag{3}$$

where x_{ij} is the input from neuron i to neuron j (on different layers), w_{ij} is the weight from neuron i to neuron j, and θ_k is the bias on the layer of neuron i.

At the output layer, we used the sum of MSEs as the loss function so as to serve as the object to minimize as the NN is trained. MSE is defined as

$$L(\vec{w}) = \text{MSE} = \frac{1}{2|D|}\sum_{d \in D}\sum_{k \in \text{outputs}}(t_{dk} - o_{dk})^2 \tag{4}$$

where D is the set of training data, t_{dk} is the target label of training instance d at output neuron k, o_{dk} is the output at output neuron k from the NN for instance d, and $\vec{w}$ is the weights on the net.

5.3.1. NN Training Parameters. We used the back-propagation (BP) algorithm for NN training.[25] BP requires a set of learning parameters, and in the following experiments, we specified such learning parameters as follows: learning rate 0.005, maximum number of iterations 100 000, steepness 1.0 on hidden layers and the output layer, and momentum 0.001.

In the following experiments, we denoted "minMSE" as the desired MSE such that once the training error reaches minMSE, the NN training process is terminated. Thus, minMSE is one of the NN training termination conditions, in addition to the maximum number of training iterations.

We did a preliminary study on the range of the optimal number of hidden layers and optimal number of hidden neurons by performing a grid search on such numbers using SSR_{base} models with 1 and 2 hidden layers and 64, 128, 256, and 512 hidden neurons on each layer. The results demonstrated that only one hidden layer suffices to learn a good model. All the experiments that are reported below utilized a neural network with a single hidden layer. Experiments with additional hidden layers did not lead to any improvements, so they are not reported.

In addition, we tested oversampling techniques with NN training and found that oversampling improves NN performance and allows a reasonable model learned from even a small set of training instances. In the experiments that are reported below, oversampling techniques are all applied.

5.4. Evaluation Methodology Metrics. The performances of the different methods are evaluated via a five-fold cross-validation in which the corresponding active compounds and inactive compounds of each target are randomly split into five folds, four folds for model learning and the other fold for testing, and each of these folds is enforced to have about the same number of selectively active compounds.

The quality of the SSR models is measured using F_1, which is the harmonic mean of precision and recall and is defined as

$$F_1 = \frac{2(\text{precision})(\text{recall})}{\text{precision} + \text{recall}} \tag{5}$$

in which precision is the fraction of correctly classified selective compounds (i.e., true positive) over all compounds that are classified as selective (i.e, true positive and false positive) by SSR models. Precision is defined as

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{6}$$

Recall in eq 5is the fraction of correctly classified selective compounds (i.e., true positive) over all selective compounds in the testing data (i.e., true positive and false negative) by SSR models. Recall is defined as

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{7}$$

Intuitively, if precision and recall are both high, or one of the two is very high, F_1 will be high. Thus, F_1 is a measure which leverages both precision and recall, and higher F_1 values indicate better performance.

Conventionally, in NN settings, if prediction scores are above 0.5 (in the case of 0/1 binary labels), it is considered a positive prediction. Thus, 0.5 by default serves as a threshold (referred to as α) to determine whether a prediction is positive, and precision and recall values are calculated on the basis of the threshold. However, in some cases a different threshold α may be preferred so as to favor or disfavor predictions above or below a certain value. In our study, we evaluated threshold values and calculated precision and recall, and F_1 corresponding to each of the thresholds α, and searched for the best parameter α which gives the best F_1. We refer to this best F_1 value as F_1^b . During the experimental evaluation, we report the average F_1^b values across five folds.

6. RESULTS

In this section, we present the results for the selectivity studies. We present the detailed results for the first data set, in which each challenge set has only one target and each target may have multiple challenge sets. In the first data set, we simply refer to t_j as the challenge target of the interested target t_i since $T_i = \{t_j\}$. In the end, we present an overall performance summary for the second data set, in which each challenge set has multiple targets and each target has only one challenge set, since the results for the second data set show a trend similar to that of the results of the first data set.

Table 1. Compound Similarity[a]

sim_s2s	sim_ns2 ns	sim_s2sns	sim_a2a	sim_a2s	sim_a2 ns	sim_rnd
0.570	0.385	0.343	0.371	0.372	0.357	0.287

[a] In this table, for each target t_i of interest in the first data set, "sim_s2s" is the average similarity between selective compounds, "sim_ns2 ns" is the average similarity between nonselective compounds, "sim_-s2ns" is the average similarity between selective compounds and nonselective compounds, "sim_a2a" is the average compound similarity between active compounds, "sim_a2s" is the average compound similarity between active compounds and selective compounds, "sim_a2ns" is the average similarity between active compounds and nonselective compounds, and "sim_rnd" is the average compound similarity between random compounds.

6.1. Compound Similarities. First, we conducted a test on compound similarity in the first data set so as to learn the nature of the problem and assess the quality of our data set. Particularly, we tested the compound similarities among selectives against selectives, actives against actives, actives against selectives, and actives against nonselectives using the Tanimoto coefficient,[26] which is defined as follows:

$$\text{sim}_c(c_x, c_y) = \frac{\sum_k c_{x,k} c_{y,k}}{\sum_k c_{x,k}^2 + \sum_k c_{y,k}^2 - \sum_k c_{x,k} c_{y,k}} \tag{8}$$

where c_x and c_y are the fixed-length feature vectors for two compounds, k goes over all the dimensions of the feature vector space, and $c_{x,k}$ is the value on the kth dimension. Note that, after dimension reduction, compound feature vectors may have negative values, and thus, the Tanimoto coefficient can be negative in this case. However, their relative comparison still reflects the differentiation across compound groups.

The detailed results are available in Supporting Information. Some statistics are available in Table 1. We also tried different compound feature vectors/fingerprints to calculate compound similarities, and the trend remains the same; that is, selective compounds are more similar to other selective compounds, nonselective compounds are more similar to other nonselective compounds, and on average active compounds are more similar to selective compounds than to nonselective compounds. This trend of compound similarities indicates that, in general, the compound selectivity problem is not trivial since we try to identify a small subset of active compounds, which are similar to other active compounds to a large extent. Also the compounds are diverse enough since they have low similarities, and therefore, it is a hard set of compounds and they are suitable for testing the proposed methods.

6.2. Effects of Dimension Reduction. Figure 5 shows how the PCA-based dimensionality reduction impacts the performance of the SSR_{base} model. The metrics plotted correspond to the average F_1^b values and model learning time over the 116 SSR tasks of DS1. This plot was obtained by reducing the dimensions of the original binary fingerprints from 2048 to 50, 100, 200, 500, 1000, and 1500, and then training SSR_{base} models on the respective reduced features. For each number of dimensions the reported results correspond to the best prediction performance over all learning parameters (i.e., learning rate, steepness, etc., and number of hidden neurons, as specified in section 5.3.1).

These results indicate that, as the number of dimensions increases, the accuracy of the SSR_{base} model improves. However,

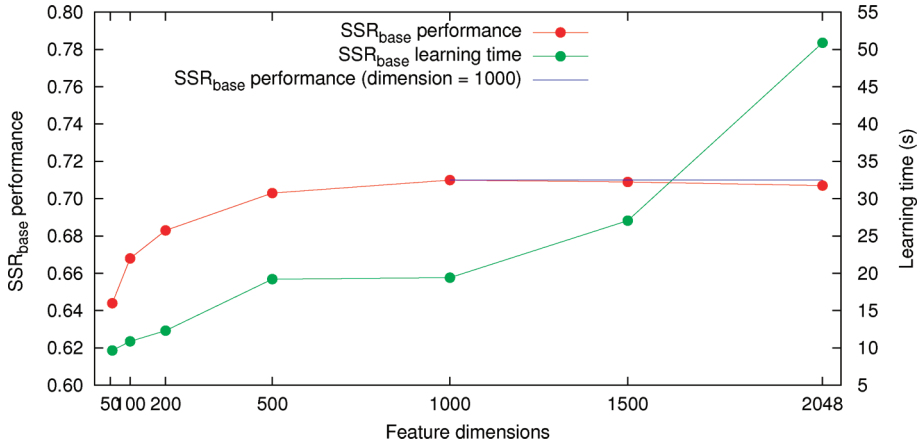


Figure 5. Effects of dimension reduction for DS1.

Table 2. SSR_{base} Average F_1^b Scores[a]

	DS1			DS2		
minMSE	32	64	128	32	64	128
0.01	0.700	0.701	0.699	0.649	0.638	0.646
0.03	0.700	0.704	0.705	0.652	0.641	0.651
0.05	0.707	**0.710**	0.707	0.658	0.654	0.654
0.07	0.704	0.708	0.702	0.655	**0.657**	0.639
0.10	0.704	0.706	0.681	0.648	0.643	0.584

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "32", "64", etc. corresponds to the results using 32, 64, etc. hidden neurons in NNs. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs.

Table 3. SAR Average F_1^b Scores[a]

	DS1			DS2		
minMSE	64	128	256	128	256	512
0.001	0.906	0.906	0.904	0.912	0.913	0.901
0.003	0.904	**0.906**	0.906	0.914	0.917	0.907
0.005	0.904	0.905	0.904	0.913	**0.918**	0.910
0.007	0.902	0.903	0.903	0.916	0.910	0.906
0.010	0.901	0.901	0.899	0.910	0.911	0.899

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "64", "128", etc. corresponds to the results using 64, 128, etc. hidden neurons in NNs. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs.

the best prediction performance is achieved at 1000 dimensions. Moreover, when 1000 dimensions are used, the amount of time required to learn the NN models is about two-fifths of that required when no dimensionality reduction is performed. For these reasons, in all of our subsequent experiments, we used the reduced 1000-dimension features to represent compounds.

6.3. Results for SSR_{base} Models. Table 2 shows the performance achieved by the SSR_{base} model on DS1 and DS2 for different numberd of hidden neurons and different minMSE values for stopping NN training. The best performance is achieved with 64 hidden neurons and minMSE values of 0.05 for DS1 and 0.07 for DS2. These results also show that when minMSE decreases, the models tend to overfit the training data and when minMSE increases, the models tend to underfit the training data. Similar trends can be observed when the number of hidden neurons increases or decreases.

A promising observation is that the overall best performance of 0.710 for DS1 and 0.657 for DS2 is substantially better than that of a random prediction, indicating that machine learning methods can be utilized to build SSR models. Also the performance on DS2 is lower than that achieved on DS1, indicating that learning SSR models when the challenge set contains multiple targets is considerably harder.

6.4. Results for SSR_c Models. Recall from section 4.2 that SSR_c uses an SSR model (level 1 model) to identify the compounds that have a predicted activity value greater than or equal to the minactivity threshold and then uses an SSR_{base} model (level 2 model) to predict which of those compounds are selective for the target of interest. For this reason, our experimental evaluation initially focuses on assessing the performance of the SAR models themselves to determine their optimal set of model parameters and then on the evaluation of model sensitivity to the minactivity threshold parameter.

Table 3 shows the performance of the SAR models for the two data sets. The best average performance for DS1 (0.906) is achieved for 128 hidden neurons and a minMSE value of 0.003, whereas the best average performance for DS2 (0.918) is achieved for 256 hidden neurons and a minMSE value of 0.005. These high F_1 scores, which result from high values of the underlying precision and recall measures, are encouraging for two reasons. First, the compounds that will be filtered out will be predominately inactives (high precision), which makes the prediction task of the level 2 model easier as it does not need to consider a large number of inactive compounds. Second, most of the selective compounds will be passed through to the level 2 model (high recall), which ensures that most of the selective compounds will be considered (i.e., asked to be predicted) by that model. This is important as the selectivity determination is done only by the level 2 model for only those compounds that pass the minactivity threshold filter of the level 1 model.

Table 4. Cascaded Model Average F_1^b Scores[a]

scheme	data set	minactivity = 0.3	minactivity = 0.4	minactivity = 0.5	minactivity = 0.6	minactivity = 0.7
SSR_c	DS1	0.727	**0.729**	0.728	0.727	0.725
	DS2	0.671	**0.676**	0.674	0.673	0.671
Wass	DS1	0.721	**0.723**	0.723	0.723	0.722
	DS2	0.631	**0.631**	0.631	0.630	0.628

[a] The rows corresponding to SSR_c show the results when the level 2 model is trained using S_i^+ as positive training instances and $S_i^- \cup C_i^-$ as negative training instances (i.e., SSR_{base}). The rows corresponding to Wass show the results when the level 2 model is trained using S_i^+ as positive training instances and S_i^- as negative training instances.[7] Rows for DS1 and DS2 show the results for data set 1 and data set 2, respectively. Each column corresponds to the results with the corresponding minactivity threshold used. The bold numbers indicate the best average performance over all minactivity thresholds. For data set 1, level 1 SAR models have 128 hidden nodes and minMSE = 0.003 and level 2 SSR_{base} models have 64 hidden nodes and minMSE = 0.05 for both SSR_c and Wass models. For data set 2, level 1 SAR models have 256 hidden nodes and minMSE = 0.005 and level 2 SSR_{base} models have 64 hidden nodes and minMSE = 0.07 for both SSR_c and Wass methods.

Table 5. SSR_{mt} Average F_1^b Scores[a]

	DS1			DS2		
minMSE	128	256	512	64	128	256
0.01	0.484	0.426	0.414	0.590	0.611	0.615
0.03	0.753	0.757	0.756	0.649	0.662	0.657
0.05	0.756	**0.759**	0.754	0.667	0.664	0.671
0.07	0.747	0.747	0.746	0.672	**0.681**	0.660
0.10	0.738	0.735	0.737	0.662	0.671	0.656

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "64", "128", etc. corresponds to the results using "64", "128", etc. hidden neurons in NNs. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs.

Table 6. SSR_{3way} Average F_1^b Scores[a]

	DS1			DS2		
minMSE	32	64	128	32	64	128
0.01	0.707	0.711	0.712	0.640	0.649	0.637
0.03	0.707	0.712	0.707	0.653	**0.664**	0.643
0.05	0.718	**0.722**	0.713	0.636	0.650	0.616
0.07	0.721	0.717	0.697	0.626	0.641	0.563
0.10	0.711	0.698	0.641	0.582	0.582	0.508

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "32", "64", etc. corresponds to the results using 32, 64, etc. hidden neurons in NNs. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs.

The first two rows of Table 4 show the performance achieved by the SSR_c models for different values of the minactivity threshold parameter. In these experiments, the level 1 models correspond to the SAR model with the optimal parameter combination that achieved the best results in Table 2 (i.e., for DS1, 128 hidden neurons and minMSE = 0.003 and, for DS2, 256 hidden neurons and minMSE = 0.005) and the level 2 models correspond to the SSR_{base} model with the optimal parameter combination that achieved the best results in Table 3 (i.e., for DS1, 64 hidden neurons and minMSE = 0.05 and, for DS2, 64 hidden neurons and minMSE = 0.07). The overall best average performances achieved by SSR_c are 0.729 and 0.679 for the DS1 and DS2 data sets, respectively, and occur when the minactivity threshold value is 0.4. Also, these results show that, as minactivity changes, the performance of the resulting SSR_c models changes as well. However, these results show that, for a relatively large number of reasonable minactivity threshold values, the overall performance remains relatively similar. Of course, if minactivity is too small or too large, then the resulting model either becomes identical to SSR_{base} or may fail to identify selective compounds due to low recall.

The second two rows of Table 4 show the performance of a cascaded SSR model in which the level 2 model uses only the nonselective compounds as the negative class. This is similar to the model used by the two-step approach developed by Wassermann et al.[7] Note that these results were obtained using the same level 1 model as that used by SSR_c and the same NN model/learning parameters used by SSR_c. The best average performances achieved by this alternate approach are 0.723 and 0.631 for DS1 and DS2, respectively, both of which are worse than those achieved by SSR_c. These results indicate that taking into account the inactive compounds in the level 2 model leads to better SSR prediction results.

6.5. Results for SSR_{mt} Models. Table 5 shows the performance achieved by the SSR_{mt} model for different numbers of hidden neurons and minMSE values. The best average performance for DS1 (0.759) happens for 256 hidden neurons and a minMSE value of 0.05, whereas the best performance for DS2 (0.681) happens for 128 hidden neurons and a minMSE value of 0.07. The performance characteristics of the SSR_{mt} model as a function of the number of hidden neurons and the minMSE value are similar to those observed earlier for the SSR_{base} model. As the number of hidden neurons decreases/increases (or the minMSE values increase/decrease), the performance of the resulting model degrades due to under- and overfitting.

6.6. Results for Three-Way Models. Table 6 shows the performance achieved by the SSR_{3way} models for different numbers of hidden neurons and minMSE values. These results were obtained by using the same set of model and learning parameters (i.e., number of hidden neurons and minMSE value) for each one of the three binary models involved (i.e., S_i^+ vs the rest, S_i^- vs the rest, and C_i^- vs the rest). The best average performance for DS1 (0.722) happens for 64 hidden neurons and a minMSE value of 0.05, whereas the best performance for DS2 (0.664) happens for 64 hidden neurons and a minMSE value of 0.03.

Table 7. SSR_{xSAR} Average F_1^b Scores[a]

	DS1			DS2		
minMSE	32	64	128	32	64	128
0.01	0.710	0.705	0.700	0.663	0.661	0.662
0.03	**0.715**	0.704	0.700	0.663	0.662	0.661
0.05	0.707	0.690	0.677	0.662	0.661	0.662
0.07	0.696	0.673	0.649	0.664	**0.664**	0.663
0.10	0.656	0.638	0.636	0.662	0.664	0.663

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "32", "64", etc. corresponds to the results using 32, 64, etc. hidden neurons in NNs of T_i. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs. The SSR_{base} models are the best ones as in Table 2.

Table 8. SSR_{3class} Average F_1^b Scores[a]

	DS1			DS2		
minMSE	64	128	512	32	64	128
0.01	0.708	0.707	0.710	0.639	0.617	0.612
0.03	0.734	0.733	0.737	0.664	**0.670**	0.657
0.05	0.740	0.739	0.66	0.661	0.667	0.650
0.07	0.739	**0.741**	0.741	0.659	0.660	0.654
0.10	0.737	0.737	0.737	0.644	0.647	0.650

[a] minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for data set 1 and data set 2, respectively. Each column labeled "32", "64", etc. corresponds to the results using 32, 64, etc. hidden neurons in NNs. The bold numbers indicate the best average performance over all minMSE values and numbers of hidden neurons for all the targets. There is only one hidden layer in all the NNs.

6.7. Results for SSR_{xSAR} Models. Table 7 shows the performance of SSR_{xSAR} models. In the SSR_{xSAR} models for t_i, t_i's only SAR model is its best baseline SAR model as identified from Table 2. The performances shown in Table 7 are achieved by using each t_i's best baseline SAR model and different numbers of hidden neurons from t_i's SAR models and minMSE values. The best average performance for DS1 (0.715) happens for 32 hidden neurons in t_i's NNs and a minMSE value of 0.03, whereas the best performance for DS2 (0.664) happens for 64 hidden neurons in t_i's NNs and a minMSE value of 0.07.

6.8. Results for Multiclass SSR_{3class} Models. Table 8 shows the performance of SSR_{3class} models achieved by different numbers of hidden neurons and minMSE values. The best average performance for DS1 (0.741) is achieved by 128 hidden neurons and a minMSE value of 0.07, whereas the best performance for DS2 (0.670) is achieved by 64 hidden neurons and a minMSE value of 0.03.

6.9. Overall Comparison. Table 9 summarizes the best average F_1^b results achieved from SSR_{base}, SSR_{xSAR}, SSR_{3way}, SSR_c, SSR_{3class}, and SSR_{mt} models on the DS1 and DS2 data sets. These results correspond to the bold-faced entries of Tables 2, 7, 6, 4, 8 and 5, respectively. (Many of the results reported in Tables 2−8 have small variations, indicating that the performance of the various methods is stable over a wide range of parameter values. In selecting the specific parameter-value combinations to report in Table 9, we used the combinations that achieved the best results in those tables. However, the relative performance of the various schemes will remain the same for other combinations of the various parameter values.) In addition, for each scheme other than SSR_{base}, the rows labeled "#imprv" show the number of prediction tasks for which the corresponding scheme outperforms the SSR_{base} models. Similarly, the rows labeled "imprv" show the average performance improvement achieved by that scheme of the SSR_{base} models. The performance improvement is calculated as the geometric mean of pairwise performance improvement over SSR_{base}.

Table 9. SSR Model Performance Comparison[a]

data set	pfm/imprv	SSR_{base}	SSR_{xSAR}	SSR_{3way}	SSR_c	SSR_{3class}	SSR_{mt}
DS1	best	0.710	0.715	0.722	0.729	0.741	0.759
	#imprv		61	72	72	71	79
	imprv		0.6%	1.6%	2.6%	4.0%	7.5%
DS2	best	0.657	0.664	0.664	0.676	0.670	0.681
	#imprv		11	11	15	11	11
	imprv		2.9%	0.8%	3.2%	2.9%	3.5%

[a] The rows labeled "best" correspond to the best performance from the corresponding SSR model given minMSE and number of hidden neurons fixed for all prediction tasks. The rows labeled "#imprv" present the number of prediction tasks for each of the corresponding SSR methods that perform better than SSR_{base}. The rows labeled "imprv" present the geometric mean of pairwise performance improvement over the baseline SSR_{base} model.

Many of the results reported in Tables 2−8 have small variations. We believe this is a nice property of the methods, as their performance is stable over a wide range of parameter values. The purpose of those tables was to illustrate the various performance trends as a function of these parameters and not necessarily to argue that one specific parameter value choice is much better than the rest. In Table 9, the performance of the methods was also assessed using statistical significance testing. In selecting the specific parameter-value combinations to report in that table, we selected those combinations that achieved the best results.

For both the data sets, the SSR_{xSAR}, SSR_{3way}, SSR_c, SSR_{3class}, and SSR_{mt} models outperform the SSR_{base} model. This indicates that by incorporating additional information (i.e., compound activity properties for the target of interest, compound activity and selectivity properties against challenge sets, etc.) rather than focusing on the selectivity property alone improves selectivity prediction performance. Among the different schemes, the SSR_{mt} models achieve the best SSR prediction results. Their average improvements over SSR_{base} for DS1 and DS2 are 7.5% and 3.5%, respectively. The SSR_{3class} models achieve the second best performance for DS1, which corresponds to an average improvement of 4.0%, and the third best performance for DS2, which corresponds to an average improvement of 2.9%. The SSR_c models achieve the third best performance for DS1, which corresponds to an average improvement of 2.6%, and the second best performance for DS2, which corresponds to an average improvement of 3.2%. Finally, even though SSR_{3way} and SSR_{xSAR} improve upon the SSR_{base} model, the gains achieved are rather modest (1.6% for DS1 and 0.8% for DS2 for SSR_{3way}, 0.6% for DS1 for SSR_{xSAR}).

A finer grained picture of the performance of the different methods on the different SSR prediction tasks involved in DS1

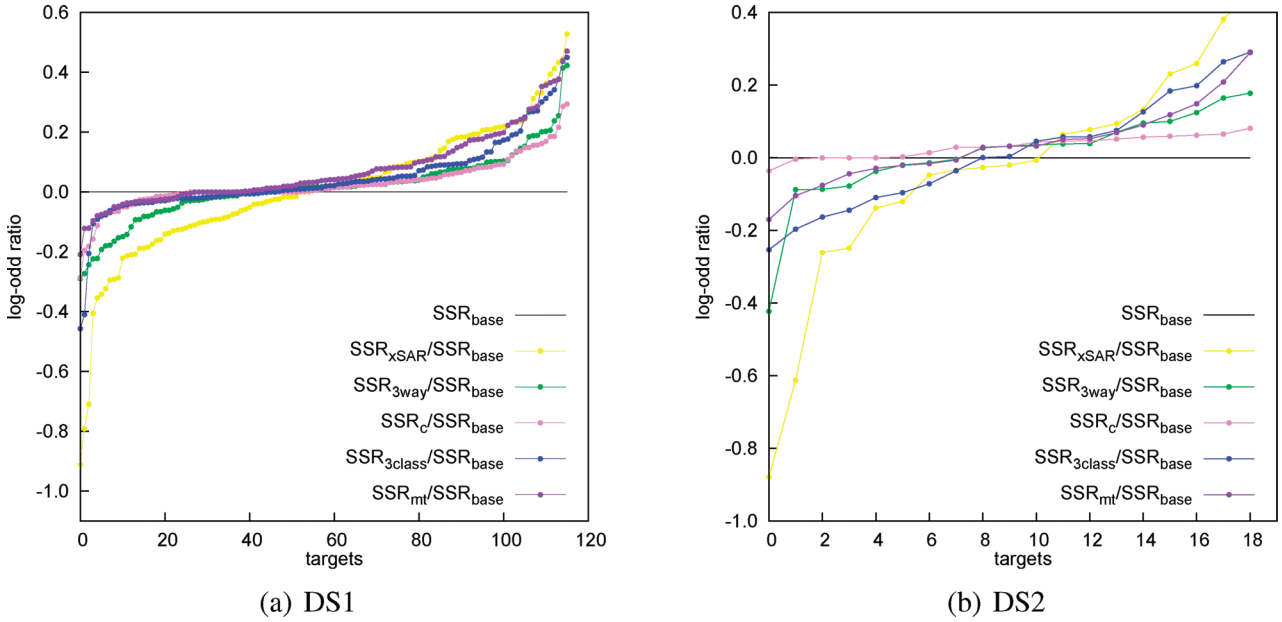


Figure 6. Pairwise improvement.

Table 10. Paired *t* Test[a]

data set	scheme	SSR_{base}	SSR_{xSAR}	SSR_{xSAR}	SSR_c	SSR_{3class}	SSR_{mt}
DS1 (116 tasks)	SSR_{base}	−/−	7.102E−01/0	6.380E−02/0	01.697E−04/1	7.019E−06/1	1.642E−10/1
	SSR_{xSAR}		−/−	5.766E−01/0	2.169E−01/0	03.402E−02/1	1.719E−04/1
	SSR_{3way}			−/−	4.157E−01/0	06.612E−03/1	2.179E−07/1
	SSR_c				−/−	7.168E−02/0	1.226E−05/1
	SSR_{3class}					−/−	6.237E−06/1
	SSR_{mt}						−/−
DS2 (19 tasks)	SSR_{base}	−/−	8.697E−01/0	6.896E−01/0	01.356E−03/1	5.840E−01/0	1.171E−01/0
	SSR_{xSAR}		−/−	9.973E−01/0	7.720E−01/0	8.715E−01/0	6.414E−01/0
	SSR_{3way}			−/−	4.705E−01/0	7.926E−01/0	4.188E−01/0
	SSR_c				−/−	7.759E−01/0	7.667E−01/0
	SSR_{3class}					−/−	4.702E−01/0
	SSR_{mt}						−/−

[a] The x/y values correspond to the p value (i.e., x) and whether the null hypothesis (i.e., the SSR method of the column performs statistically the same as the SSR method of the row) is rejected (i.e., $y = 1$) or not (i.e., $y = 0$) at the 5% significance level.

and DS2 is shown in the plots of Figure 6. These plots show the log ratios of the F_1^b scores achieved by each model over that achieved by the baseline model for the 116 SSR tasks of DS1 (6.9) and the 19 SSR tasks of DS2 (6.9). For each SSR model, the results in Figure 6 are presented in a nonincreasing order according to these log ratios. Figure 6 shows that SSR_{mt} leads to higher improvements for more individual SSR prediction tasks than SSR_{3way} and SSR_c and that SSR_c performs slightly better than SSR_{3way}. SSR_{xSAR} and SSR_{3class} have more dynamic performance than the other models. The actual numbers of SSR prediction tasks for which each method outperforms the baseline are shown in the row labeled "#imprv" of Table 9.

Table 10 presents the paired *t* test across different SSR methods. It shows that, for DS1, SSR_c, SSR_{3class}, and SSR_{mt} all outperform SSR_{base} significantly. In addition, SSR_{3class} outperforms other SSR methods significantly except SSR_c, and SSR_{mt} outperforms all the other SSR methods significantly. However, for DS2, the statistical test did not show significant differences among all the SSR methods, even though SSR_{mt} outperforms the others in terms of F_1^b.

Comparing the performance across the two data sets, we see that the proposed methods are able to achieve considerably better improvements for DS1 than DS2. We believe that this is due to the fact that the underlying learning problems associated with the prediction tasks in DS2 are harder, since the challenge sets contain more than one target.

7. CONCLUSIONS

In this paper, we developed two machine learning methods, SSR_c and SSR_{mt}, for building SSR models and experimentally evaluated them against the previously developed methods SSR_{base}, SSR_{xSAR}, SSR_{3way}, and SSR_{3class}. Our results (i.e., Tables 9 and 10) showed that the SSR_{mt} approaches achieve the best results, substantially outperforming other methods on a large

number of different SSR prediction tasks. This multitask model combines activity and selectivity models for multiple proteins into a single model such that, during training, the two models are learned simultaneously and compound preference over targets is learned and transferred across. This suggests that collectively considering information from multiple targets and also compound activity and selectivity properties for each target benefits selectivity prediction.

Our experiments showed that even though the multitask learning-based SSR models can achieve good performance for the SSR prediction tasks in which the challenge set contains a single other target, their performance for multitarget challenge sets is considerably lower. This indicates that future research is required for developing methods which better predict SSR with multitarget challenge sets. A potential approach is that SSR_{mt} models can be constructed to have outputs for each of the targets in the challenge set such that more information is expected to be learned through the multitask learning.

We use neural networks as the learning algorithms due to their flexible and adaptive nature for multitask learning, despite the fact that there exist some other (stronger in general) learning algorithms. Another widely used algorithm for compound classification is SVMs.[9] In our primary studies, we conducted experiments using SVMs with different kernel functions on the reduced 1000 bit compound features and original 2048 bit compound features for the SSR_{base} and SSR_m methods on DS1. Our results demonstrated that, in both SSR_{base} and SSR_{mt}, SVMs did not perform better than NNs on our data set, so we did not apply SVMs for SSR models.

■ ASSOCIATED CONTENT

Supporting Information. Data sets (i.e., compounds, targets, and compound−target activity information) and detailed README file. Warning: This is a very large file. This material is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author
*E-mail: karypis@cs.umn.edu.

■ ACKNOWLEDGMENT

This paper is an extension of the paper "Improved Machine Learning Models for Predicting Selective Compounds" presented at the ACM Conference on Bioinformatics, Computational Biology and Biomedicine 2011, Chicago, IL, Aug 1−3, 2011. This work was supported in part by the National Science Foundation (NSF; Grants IIS-0905220, OCI-1048018, and IOS-0820730), the Department of Energy (DOE) Grant USDOE/DE-SC0005013, and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

■ REFERENCES

(1) Karaman, M. W. A Quantitative Analysis of Kinase Inhibitor Selectivity. *Nat. Biotechnol.* **2008**, *26*, 127–132.

(2) Hansch, C.; Maloney, P. P.; Fujita, T.; Muir, R. M. Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substituent Constants and Partition Coefficients. *Nature* **1962**, *194*, 178–180.

(3) Hansch, C.; Muir, R. M.; Fujita, T.; Maloney, P. P.; Geiger, F.; Streich, M. The Correlation of Biological Activity of Plant Growth-Regulators and Chloromycetin Derivatives with Hammett Constants and Partition Coefficients. *J. Am. Chem. Soc.* **1963**, *85*, 2817–1824.

(4) Peltason, L.; Hu, Y.; Bajorath, J. From Structure-Activity to Structure-Selectivity Relationships: Quantitative Assessment, Selectivity Cliffs, and Key Compounds. *ChemMedChem* **2009**, *4*, 1864–1873.

(5) Vogt, I.; Stumpfe, D.; Ahmed, H. E. A.; Bajorath, J. Methods for Computer-Aided Chemical Biology. Part 2: Evaluation of Compound Selectivity Using 2D Molecular Fingerprints. *Chem. Biol. Drug Des.* **2007**, *70*, 195–205.

(6) Stumpfe, D.; Geppert, H.; Bajorath, J. Methods for Computer-Aided Chemical Biology. Part 3: Analysis of Structure-Selectivity Relationships through Single- or Dual-Step Selectivity Searching and Bayesian Classification. *Chem. Biol. Drug Des.* **2008**, *71*, 518–528.

(7) Wassermann, A. M.; Geppert, H.; Bajorath, J. Searching for Target-Selective Compounds Using Different Combinations of Multiclass Support Vector Machine Ranking Methods, Kernel Functions, and Fingerprint Descriptors. *J. Chem. Inf. Model.* **2009**, *49*, 582–592.

(8) Wassermann, A. M.; Geppert, H.; Bajorath, J. Application of Support Vector Machine-Based Ranking Strategies to Search for Target-Selective Compounds. *Methods Mol. Biol.* **2011**, *672*, 517–530.

(9) Vapnik, V. *Statistical Learning Theory*; John Wiley: New York, 1998.

(10) Thrun, S.; O'Sullivan, J. Discovering Structure in Multiple Learning Tasks: The TC Algorithm. *Proceedings of the International Conference on Machine Learning*, Bari, Italy, 1996; Morgan Kaufmann: San Mateo, CA; pp 489−497.

(11) Bonilla, E.; Agakov, F.; Williams, C. Kernel Multi-task Learning using Task-specific Features. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007, Omnipress: San Juan, Puerto Rico, 2007; pp 43−50.

(12) Caruana, R. Learning Many Related Tasks at the Same Time with Backpropagation. *Adv. Neural Inf. Process. Syst.* **1995**, *22*, 657–664.

(13) Evgeniou, T.; Micchelli, C. A.; Pontil, M. Learning Multiple Tasks with Kernel Methods. *J. Mach. Learn. Res.* **2005**, *6*, 615–637.

(14) Yu, K.; Tresp, V.; Schwaighofer, A. Learning Gaussian Processes from Multiple Tasks. *Proceedings of 22nd International Conference on Machine Learning*, 2005, ACM Press: Bonn, Germany, 2005; pp 1012−1019.

(15) Ni, K.; Carin, L.; Dunson, D. Multi-Task Learning for Sequential Data via iHMMs and the Nested Dirichlet Process. *Proceedings of the 24th International Conference on Machine Learning*, 2007, ACM: New York, 2007; pp 689−696.

(16) Agarwal, A.; Rakhlin, A.; Bartlett, P. *Matrix Regularization Techniques for Online Multitask Learning*; Technical Report UCB/EECS-2008-138; EECS Department, University of California: Berkeley, CA, 2008.

(17) Jacob, L.; Hoffmann, B.; Stoven, V.; Vert, J.-P. Virtual Screening of GPCRs: An in Silico Chemogenomics Approach. *BMC Bioinf.* **2008**, *9*, 363.

(18) Ning, X.; Rangwala, H.; Karypis, G. Multi-Assay-Based Structure-Activity Relationship Models: Improving Structure-Activity Relationship Models by Incorporating Activity Information from Related Targets. *J. Chem. Inf. Model.* **2009**, *49*, 2444–2456.

(19) Jin, F.; Sun, S. A Multitask Learning Approach to Face Recognition Based on Neural Networks. *Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, 2008, Springer-Verlag: Berlin, Heidelberg, 2008; pp 24−31.

(20) Keller, M.; Bengio, S. *A Multitask Learning Approach to Document Representation Using Unlabeled Data*; Technical Report 44; IDIAP Research Institute: Switzerland, 2006.

(21) Stumpfe, D.; Ahmed, H. E. A.; Vogt, I.; Bajorath, J. Methods for Ccomputer-Aided Chemical Biology. Part 1: Design of a Benchmark System for the Evaluation of Compound Selectivity. *Chem. Biol. Drug Des.* **2007**, *70*, 182–194.

(22) Stumpfe, D.; Lounkine, E.; Bajorath, J. Molecular Test Systems for Computational Selectivity Studies and Systematic Analysis of Compound Selectivity Profiles. *Methods Mol. Biol.* **2011**, *672*, 503–515.

(23) Wale, N.; Ning, X.; Karypis, G. Trends in Chemical Graph Data Mining. In *Managing and Mining Graph Data*; Aggarwal, C. C., Wang, H., Eds.; Springer: New York, 2010; Vol. 40, pp 581−606.

(24) Clarke, R.; Ressom, H. W.; Wang, A.; Xuan, J.; Liu, M. C.; Gehan, E. A.; Wang, Y. The Properties of High-Dimensional Data Spaces: Implications for Exploring Gene and Protein Expression Data. *Nat. Rev. Cancer* **2008**, *8*, 37–49.

(25) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning Internal Representations by Error Propagation. *Comput. Models Cognit. Percept. Ser.* **1986**, 318–362.

(26) Willett, P.; Barnard, J. M.; Downs, G. M. Chemical Similarity Searching. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 983–997.