Improved Machine Learning Models for Predicting Selective Compounds

Xia Ning
Computer Science & Engineering
University of Minnesota, Twin Cities
xning@cs.umn.edu

George Karypis
Computer Science & Engineering
University of Minnesota, Twin Cities
karypis@cs.umn.edu

Abstract

The identification of small potent compounds that selectively bind to the target under consideration with high affinities is a critical step towards successful drug discovery. However, there still lacks 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 multi-task learning method. The cascaded method decomposes the selectivity prediction into two steps, one model for each step, so as to effectively filter out non-selective compounds. The multi-task method incorporates both activity and selectivity models into one multi-task model so as to better differentiate compound selectivity properties. We conducted a comprehensive set of experiments and compared the results with other conventional selectivity prediction methods, and our results demonstrated that the cascaded and multi-task 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[1] 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 later 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. Selectivity test can include binding assays or clinical trials [9]. The problem with such an approach is that it defers selectivity assessment to the later stages, which if 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* [6] led to the development of computational methods for predicting Structure-Activity-Relationships (SAR). 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-Relationships (SSR) models [14]. Existing computational methods for building SSR models fall into two general classes. The first contains methods that determine selectivity by using SAR 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 do not have a good mechanism to effectively differentiate selective active compounds from non-selective active compounds. 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 SSR, builds on previously developed techniques and incorporates a pair of models on two levels. The level one is a standard SAR model which identifies the compounds that bind to the target regardless of their selectivity. The level two is a model that further screens the compounds identified by the level-one model in order 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 multi-task SSR, incorporates information from multiple targets and multiple prediction tasks, and builds a multi-task SSR model. The key insight is that

[1]This protein is referred to as the *target*.

compound activity/selectivity properties for other proteins can be utilized when building a 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 multi-task 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 datasets derived from publicly available resources (ChEMBL[2]) 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 multi-task 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 multi-task 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 is the conclusion and directions for future research.

2. RELATED WORK

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 towards this goal. Vogt *et al.*[21] investigated approaches for identifying selective compounds based on 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.*[17] 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 non-selective 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.*[23, 24] built on this work and investigated the use of Support Vector Machines (SVM) [20] 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, non-selective active, and inactive compounds, respectively, and whose negative class correspond to the compounds that did not belong to the positive class. 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 non-selective 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 non-selectives (i.e., selectives > inactives > non-selectives). 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.

Multi-task learning [19, 2] 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 multi-task 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 multi-task model utilizes back-propagation Neural Networks (NN) [3]. 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 multi-task learning methods have emerged, which include kernel methods [5], Gaussian processes [25], task clustering [19], Bayesian models [12], matrix regularization [1], *etc.* Various studies have reported promising results with the use of multi-task learning in diverse areas such as Cheminformatics [7, 13], face recognition [8], and text mining [10].

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, a set 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}

[2]http://www.ebi.ac.uk/chembl/

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[3]). A compound will be considered to be *active* for a given target if its IC_{50} value for that target is less than $1000nM$. For each target t_i, its set 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:

$$\begin{aligned} &\text{(i)} \quad c \text{ is active for } t_i, \text{ and} \\ &\text{(ii)} \quad \min_{\forall t_j \in T_i} \frac{\text{IC}_{50}(c,t_j)}{\text{IC}_{50}(c,t_i)} \geq 50. \end{aligned} \tag{1}$$

This definition follows the common practice of using the ratio of binding affinities in determining the selectivity of compounds [16]. 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 like $T_{i,1}, T_{i,2}, \cdots, T_{i,n}$. 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 *Structure-Selectivity-Relationship model* (SSR) is to predict if 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 Equation 1 are satisfied by the predicted binding affinities. The second approach is to build a classification model that is designed to directly predict if 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 Equation 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 multi-class classification models.

The use of classification- over regression-based approaches was motivated by earlier research for the problem of building SAR models, which showed that predicting a compound's binding affinity is considerably harder than that of predicting if a compound is active or not[11].

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 on the compounds that they use to define the negative class (second set of compounds).

The first model, by ignoring the non-selective active compounds (S_i^-) can potentially learn a model that differentiates between actives and inactives, irrespective of whether the active compounds are selective or not. The second model, by ignoring the inactive compounds (C_i^-) 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}.

SSR_{base} method constructs the SSR model by treating both the inactive and non-selective 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 similar characteristics to the model learned using only C_i^- as the negative class. To overcome this problem, while constructing the negative class, an equal number of compounds from S_i^- and C_i^- were 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

The 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. In order to address this shortcoming, we developed a *cascaded* SSR model that takes into account all the compounds (selectives, non-selectives, 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 one is a normal SAR model that tries to differentiate between ac-

[3]http://www.ncgc.nih.gov/guidance/section3.html

tive and inactive compounds, and the model on level two is a model that tries to differentiate between selective and non-selective compounds. The level-one model serves as a filter for the level-two 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-one model, and only those compounds whose prediction value is above a certain threshold, referred to as the *minactivity* threshold, go through the level-two SSR model. Only compounds classified as positive by the level-two SSR model will be considered as selective. This two-level cascaded SSR model is refereed to as SSR_{c}.

The level-one 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-two 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-one 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-two 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-one model.

Note that even though the cascaded SSR_{c} model is similar in spirit to the two-step approach proposed by Wassermann *et al* [23], it differs in two important ways. First, instead of sending a constant number of the highest ranked compounds (as predicted by the level-one model) to the level-two model, SSR_{c} uses the *minactivity* threshold to determine the compounds that will be *routed* to the level-two model. Second, instead of using only the S_i^- compounds as the negative class of the level-two model, SSR_{c} uses the compounds in $S_i^- \cup C_i^-$ as the corresponding negative class. As the experiments presented in Section 6.3 show, this change leads to better performance.

4.3 Multi-Task SSR Models

Both the baseline and the cascaded SSR models take into account the labels of the training compounds (i.e., selective, non-selective, 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 is active for t_i and it is inactive for all the targets in T_i, then there is a higher probability that the compound is also selective for t_i. Similarly, if a compound is selective for one target in T_i, then by definition, this compound is non-selective for t_i.

Motivated by this observation, we developed another model that in addition to the activity and selectivity information for t_i, it 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

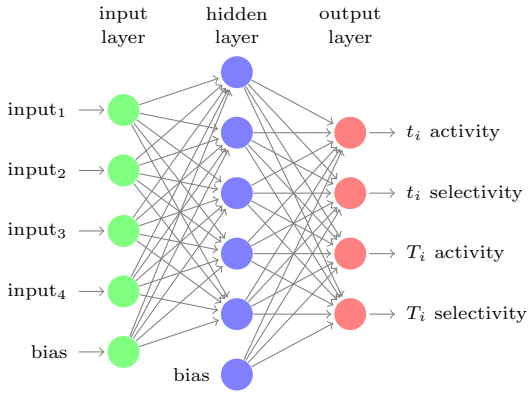


Figure 1: A multi-task neural network for target t_i and challenge set T_i.

multi-task 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 structure of the compounds. Each training compound has four labels (one for each output) and during training, the various model parameters are estimated so that 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, than 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 3-way classification approach developed by Wassermann *et al.* [23] 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^+), non-selective (S_i^-), and inactive (C_i^-) sets of compounds, respectively. 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., in order 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 3-way classification method is to model different classes separately and to decide the class of a new instance based on how differently it is classified by different models. We will denote this SSR model as SSR_{3way}.

5. MATERIALS

5.1 Datasets

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 drug-like small molecules. We first selected an initial set of molecular targets and their corresponding ligands from ChEMBL based on 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 if they satisfy 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 access if their active compounds are selective or not, whereas the second condition is designed to keep the targets that contain a sufficient number of selective compounds in order to learn a SSR model. These filtering steps resulted in a dataset with 98 protein targets. For each target t_i of these targets, we used all of its known active compounds and constructed an equal-size set of inactive compounds. The inactive compounds were selected from t_i's known inactive compounds if sufficient, otherwise, they were randomly selected from ChEMBL that always show extremely low binding affinities against arbitrary targets. Such compounds are very likely to be truly inactive against t_i even no experimental results are available, because they exhibit some strong drug-unlike properties evidenced by other targets. Note that during the random compound selection for t_i, we guaranteed that the randomly-selected inactive compounds are not in C_j^+ or C_j^- of any other targets (i.e., $i \neq j$). This is done in order to avoid the situation in which a randomly selected compound is determined to be selective for some targets.

Using these 98 targets, we constructed two datasets for experimental testing. The first dataset, 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 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 number of active and selective compounds for the target of interest is 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_is are considered.

The second dataset, 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 based on Equation 1. These criteria result in on average 3.9 targets in each challenge set, and the average number of active and selective compounds for the target of interest is 198 and 27, respectively.

The first dataset 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 dataset construction exercise [23, 18] and in real experimental settings. Meanwhile, it maximizes the number of interested targets to test for any statistically significant conclusions. The second dataset is constructed to test the generalizability of SSR models. Additional details on the targets, compounds, and the two datasets are available at [4].

5.2 Compound Representations

We generated 2048-bit binary Chemaxon compound descriptors[5] for all the compounds extracted as in 5.1. Then we applied a PCA-based dimension reduction method such that the 2048 dimensions are reduced to 1000 dimensions. Each compound is then represented by such a 1000-dimension feature vector and thus a NN with 1000 input nodes can be trained on such compound representations. We used the Chemaxon software `generatemd` to generate initial descriptors, and a Matlab dimension reduction toolbox[6] with PCA option to reduce descriptor dimensions.

Note that chemical compounds can be represented by different fingerprints [22]. However, since our study does not

[4] http://www-users.cs.umn.edu/~xning/selectivity/
[5] http://www.chemaxon.com/
[6] http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

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 reduced is performed since the NN may suffer from the curse of dimensionality [4] if high-dimension inputs are encountered.

5.3 Neural Networks

We used the publicly available neural network software FANN[7] for our neural network implementation. FANN implements fast multi-layer artificial neural networks with support for both fully connected and sparsely connected networks. We used 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 non-linear 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 as of neuron i.

At the output layer, we used Sum of Mean Square Errors (MSE) as the loss function so as to serve as the object to minimize as the NN is trained. MSE is defined as

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

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 Back-Propagation (BP) algorithm for NN training [15]. 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 100000, steepness 1.0 on hidden layers and 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 maximum number of training iterations.

We did a preliminary study on the range of 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, respectively. 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.

[7] http://leenissen.dk/fann/

5.4 Evaluation Methodology & Metrics

The performance 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 5 folds, four folds for model learning and the rest 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 defined as follows in Equation 5:

$$\text{F}_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

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 in Equation 6.

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

Recall in Equation 5 is the fraction of correctly classified selective compounds (i.e., true positive) over all selective compounds in testing data (i.e., true positive and false negative) by SSR models. Recall is defined as in Equation 7.

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

Intuitively, if precision and recall are both high, or one of the two is very high, F_1 measure will be high, and 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 case of 0/1 binary labels), it is considered as positive prediction, and thus 0.5 by default serves as a threshold (referred to as α) to determine if a prediction is positive or not, and precision and recall values are calculated based on 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 measure corresponding to each of the thresholds α, and search for the best parameter α which gives best F_1 measure. We refer to this best F_1 values as F_1^{b}. During the experimental evaluation, we report the average F_1^{b} values across 5 folds.

6. RESULTS

6.1 Effects of Dimension Reduction

Figure 2 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 trained SSR_{base} models on respective reduced features. For each number of dimensions the reported results

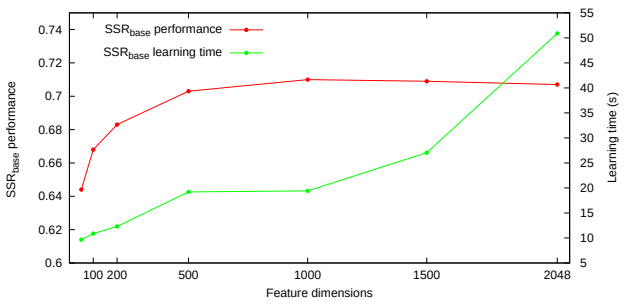


Figure 2: Effects of dimension reduction for DS1.

Table 1: SSR_{base} Average F_1^b Scores.

minMSE	DS1			DS2		
	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

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 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 1 hidden layer in all the NNs.

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, 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 2/5 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.

6.2 Results for Baseline Models

Table 1 shows the performance achieved by the SSR_{base} model on the DS1 and DS2 datasets for different number 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.

Table 2: SAR Average F_1^b Scores.

minMSE	DS1			DS2		
	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

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 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 1 hidden layer in all the NNs.

6.3 Results for Cascaded Models

Recall from Section 4.2 that SSR_c uses a SAR model (level-one model) to identify the compounds that have a predicted activity value greater than or equal to the *minactivity* threshold and then uses a SSR_{base} model (level-two 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 in order to determine their optimal set of model parameters, and then on the evaluation of model sensitivity to the *minactivity* threshold parameter.

Table 2 shows the performance of the SAR models for the two datasets. 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 recalls 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-two 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-two 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-two model for only those compounds that pass the *minactivity*-threshold filter of the level-one model.

The first two rows of Table 3 show the performance achieved by the SSR_c models for different values of the *minactivity* threshold parameter. In these experiments, the level-one models correspond to the SAR model with the optimal parameter combination that achieved the best results in Table 2 (i.e., DS1: 128 hidden neurons and *minMSE* 0.003; DS2: 256 hidden neurons and *minMSE* 0.005) and the level-two models correspond to the SSR_{base} model with the op-

Table 3: Cascaded Model Average F_1^b Scores.

scheme	dataset	*minactivity*				
		0.3	0.4	0.5	0.6	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

The rows corresponding to SSR_c show the results when the level-two model is trained using S_i^+ as positive training instances and $S_i^- \bigcup C_i^-$ as negative training instances (i.e., SSR_{base}). The rows corresponding to Wass show the results when the level-two model is trained using S_i^+ as positive training instances and S_i^- as negative training instances [23]. Rows for DS1 and DS2 show the results for dataset one and dataset two, respectively. Each column corresponds to the results with corresponding *minactivity* threshold used. The **bold** numbers indicate the best average performance over all *minactivity* thresholds. For dataset one, level-one SAR models have 128 hidden nodes and *minMSE* 0.003, and level-two SSR_{base} models have 64 hidden nodes and *minMSE* 0.05 for both SSR_c and Wass models. For dataset two, level-one SAR models have 256 hidden nodes and *minMSE* 0.005, and level-two SSR_{base} models have 64 hidden nodes and *minMSE* 0.07 for both SSR_c and Wass methods.

timal parameter combination that achieved the best results in Table 1 (i.e., DS1: 64 hidden neurons and *minMSE* 0.05; DS2: 64 hidden neurons and *minMSE* 0.07). The overall best average performance achieved by SSR_c is 0.729 and 0.679 for the DS1 and DS2 datasets, respectively and occurs 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 3 show the performance of a cascaded SSR model in which the level-two 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* [23]. Note that these results were obtained using the same level-one model as that used by SSR_c and the same NN model/learning parameters used by SSR_c. The best average performance achieved by this alternate approach is 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-two model leads to better SSR prediction results.

6.4 Results for Multi-Task Models

Table 4 shows the performance achieved by the SSR_{mt} model for different number 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 increases/decreases) the performance of the resulting model degrades due to under- and over-fitting.

Table 4: SSR_{mt} Average F_1^b Scores.

minMSE	DS1			DS2		
	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

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 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 1 hidden layer in all the NNs.

Table 5: SSR_{3way} Average F_1^b Scores.

minMSE	DS1			DS2		
	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	0.610	0.582	0.508

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 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 1 hidden layer in all the NNs.

6.5 Results for Three-Way Models

Table 5 shows the performance achieved by the SSR_{3way} models for different number 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.

6.6 Overall Comparison

Table 6 summarizes the best average F_1^b results achieved from SSR_{base}, SSR_{3way}, SSR_c and SSR_{mt} models on the DS1 and DS2 datasets. These results correspond to the bold-faced entries of Tables 1, 5, 3, and 4, respectively. 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

Table 6: SSR **Model Performance Comparison.**

dataset	pfm/imprv	scheme			
		SSR_{base}	SSR_{3way}	SSR_{c}	SSR_{mt}
DS1	best	0.710	0.722	0.729	0.759
	#imprv	-	72	72	79
	imprv	-	1.6%	2.6%	7.5%
DS2	best	0.657	0.664	0.676	0.681
	#imprv	-	11	15	11
	imprv	-	0.8%	3.2%	3.5%

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 the corresponding SSR method performs better than baseline SSR_{base}. The rows labeled imprv present the geometric mean of pairwise performance improvement over baseline SSR_{base} model.

SSR_{base}.

For both the datasets, the SSR_{3way}, SSR_{c} 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 selectivity property alone improves selectivity prediction performance. Among the different schemes, the SSR_{mt} models achieve the best SSR prediction results. Their average improvement over SSR_{base} for DS1 and DS2 is 7.5% and 3.5%, respectively. The SSR_{c} models achieve the next best performance, which corresponds to an average improvement of 2.6% and 3.2% for DS1 and DS2, respectively. Finally, even though SSR_{3way} improves upon the SSR_{base} model, the gains achieved are rather modest (1.6% for DS1 and 0.8% for DS2).

A finner-grain picture of the performance of the different methods on the different SSR prediction tasks involved in DS1 and DS2 is shown in the plots of Figure 3. 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 (Figure 3(a)) and the 19 SSR tasks of DS2 (Figure 3(b)). For each SSR model, the results in Figure 3 are presented in a non-increasing order according to these log-ratios. Figure 3 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}. The actual number of SSR prediction tasks for which each method outperforms the baseline are shown in the row labeled "#imprv" of Table 6.

Finally, comparing the performance across the two datasets 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 problem 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 two previously developed methods SSR_{base} and SSR_{3way}. Our results showed that the SSR_{mt} approaches achieve the best results, substantially outperforming other methods on a large number of different SSR prediction tasks. This multi-task 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 multi-task 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 multi-target challenge sets is considerably lower. This indicates that future research is required for developing methods which better predict SSR with multi-target challenge sets. A potential approach is that SSR_{mt} models can be constructed to have outputs for each of the targets in challenge set such that more information is expected to be learned through the multi-task learning.

Acknowledgment

This work was supported in part by NSF (IIS-0905220, OCI-1048018, and IOS-0820730) 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.

8. REFERENCES

[1] A. Agarwal, A. Rakhlin, and P. Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008.

[2] E. Bonilla, F. Agakov, and C. Williams. Kernel multi-task learning using task-specific features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007. Omnipress.

[3] R. Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664, Denver, CO, USA, 1995. Morgan Kaufmann.

[4] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nat Rev Cancer*, 8(1):37–49, Jan 2008.

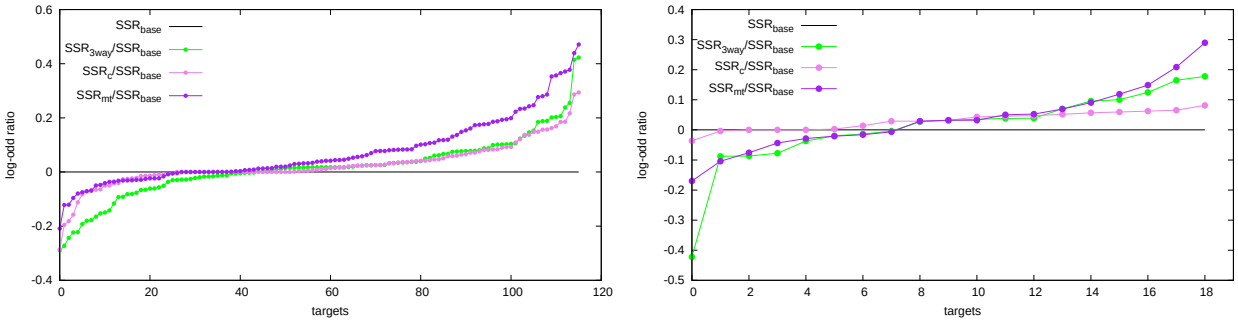
[5] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

[6] C. Hansch, R. M. Muir, T. Fujita, C. F. Maloney, and S. M. The correlation of biological activity of plant growth-regulators and chloromycetin derivatives with hammett constants and partition coefficients. *Journal of American Chemical Society*, 85:2817–1824, 1963.

[7] L. Jacob, B. Hoffmann, V. Stoven, and J.-P. Vert. Virtual screening of gpcrs: An in silico chemogenomics approach. *BMC Bioinformatics*, 9(1):363, 2008.

[8] F. Jin and S. Sun. A multitask learning approach to face recognition based on neural networks. In *Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] M. W. Karaman, S. Herrgard, D. K. Treiber, P. Gallant, C. E. Atteridge, B. T. Campbell, K. W. Chan, P. Ciceri,



(a) Dataset 1.

(b) Dataset 2.

Figure 3: Pairwise Improvement.

M. I. Davis, P. T. Edeen, R. Faraoni, M. Floyd, J. P. Hunt, D. J. Lockhart, Z. V. Milanov, M. J. Morrison, G. Pallares, H. K. Patel, S. Pritchard, L. M. Wodicka, and P. P. Zarrinkar. A quantitative analysis of kinase inhibitor selectivity. *Nat Biotechnol*, 26(1):127–132, Jan 2008.

[10] M. Keller and S. Bengio. A multitask learning approach to document representation using unlabeled data. IDIAP-RR 44, IDIAP, 2006.

[11] R. Kim and J. Skolnick. Assessment of programs for ligand binding affinity prediction. *J Comput Chem*, 29(8):1316–1331, Jun 2008.

[12] K. Ni, L. Carin, and D. Dunson. Multi-task learning for sequential data via ihmms and the nested dirichlet process. In *Proceedings of the 24th international conference on Machine learning*, pages 689–696, New York, NY, USA, 2007. ACM.

[13] X. Ning, H. Rangwala, and G. Karypis. Multi-assay-based structure-activity relationship models: improving structure-activity relationship models by incorporating activity information from related targets. *J Chem Inf Model*, 49(11):2444–2456, Nov 2009.

[14] L. Peltason, Y. Hu, and J. Bajorath. From structure-activity to structure-selectivity relationships: quantitative assessment, selectivity cliffs, and key compounds. *ChemMedChem*, 4(11):1864–1873, Nov 2009.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Mit Press Computational Models Of Cognition And Perception Series*, pages 318–362, 1986.

[16] D. Stumpfe, H. E. A. Ahmed, I. Vogt, and J. Bajorath. Methods for computer-aided chemical biology. part 1: Design of a benchmark system for the evaluation of compound selectivity. *Chem Biol Drug Des*, 70(3):182–194, Sep 2007.

[17] D. Stumpfe, H. Geppert, and J. Bajorath. 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*, 71(6):518–528, Jun 2008.

[18] D. Stumpfe, E. Lounkine, and J. Bajorath. Molecular test systems for computational selectivity studies and systematic analysis of compound selectivity profiles. *Methods Mol Biol*, 672:503–515, 2011.

[19] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *International Conference on Machine Learning*, pages 489–497, Bari, Italy, 1996. Morgan Kaufmann.

[20] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, NY, US, 1998.

[21] I. Vogt, D. Stumpfe, H. E. A. Ahmed, and J. Bajorath. Methods for computer-aided chemical biology. part 2: Evaluation of compound selectivity using 2d molecular fingerprints. *Chem Biol Drug Des*, 70(3):195–205, Sep 2007.

[22] N. Wale, X. Ning, and G. Karypis. Trends in chemical graph data mining. In A. K. Elmagarmid, C. C. Aggarwal, and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 581–606. Springer US, New York, NY, US, 2010.

[23] A. M. Wassermann, H. Geppert, and J. Bajorath. Searching for target-selective compounds using different combinations of multiclass support vector machine ranking methods, kernel functions, and fingerprint descriptors. *J Chem Inf Model*, 49(3):582–592, Mar 2009.

[24] A. M. Wassermann, H. Geppert, and J. Bajorath. Application of support vector machine-based ranking strategies to search for target-selective compounds. *Methods Mol Biol*, 672:517–530, 2011.

[25] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of 22nd International Conference on Machine Learning*, pages 1012–1019, Bonn, Germany, 2005. ACM Press.