J Comput Aided Mol Des (2007) 21:87–95
DOI 10.1007/s10822-007-9108-0

ORIGINAL PAPER

kScore: a novel machine learning approach that is not dependent on the data structure of the training set

Scott Oloff · Ingo Muegge

Received: 15 November 2006 / Accepted: 16 January 2007 / Published online: 28 February 2007

Abstract Currently machine learning approaches used in Quantitative Structure Activity Relationship (QSAR) model generation impose restrictions and/or make assumptions on how the training set descriptors correlate with a target activity. kScore has been developed as the first machine learning approach that does not require the training data to conform to a defined kernel, accommodates uneven data point distributions in the descriptor space, and optimizes the weight of each dimension in the descriptor space in order to identify the descriptors most relevant to the target property. The ability of kScore to adapt to virtually any correlation makes it essential that generalization terms be included to inhibit overtraining. The Structural Risk Minimization principle and the linear ε-insensitive loss terms have been added to the kScore optimization function. The resulting kScore algorithm has proven to be quite universal across several datasets and either produces results similar to or outperforms the most predictive machine learning algorithms tested, such as SVM, kNN, Recursive Partitioning, Neural Networks, Gaussian Process, and the Bayesian Classifier.

Keywords Virtual screening · Applicability domain · Predictive modeling · QSAR

S. Oloff (✉) · I. Muegge
Boehringer Ingelheim Pharmaceuticals Inc., 900 Ridgebury Road, P.O. Box 368, Ridgefield, CT 06877-368, USA
e-mail: soloff@rdg.boehringer-ingelheim.com

Introduction

Ligand-based drug design has become and will continue to be an important tool for drug discovery. Knowledge acquired from compounds already biologically tested is a vital asset that directs the design of future molecules. For a small, focused set of compounds with known biological activities a Structure Activity Relationship (SAR) can be established based on structural variations within the set. For larger, more diverse compound sets it becomes difficult to synergistically incorporate the results of all previous molecules to predict a new untested compound. Machine learning algorithms attempt to build SAR models in silico based on correlations found in training data.

There are several machine learning algorithms available that correlate training data to a prespecified function, commonly referred to as function-dependent or kernel based algorithms [1–4], such as Support Vector Machines (SVM) [3]. These tools find the best fit between the training data and activities with the assumption that the training SAR matches a specific linear or nonlinear function. There are other approaches that do not require the training data to match a specific function but come with a unique set of restrictions. The Bayesian Classifier (BC) and Recursive Partitioning (RP) for example use the activity distribution within each training descriptor to predict the activity class of a test compound; however, they cannot simultaneously analyze the combined effect of multiple descriptors [5, 6]. These types of approaches are best suitable for descriptors that additively impact the training activity. If specific combination(s) of descriptors agree well with the training activity but do

not independently correlate with the training activity then these approaches do not perform well. Yet another algorithm called the k Nearest Neighbor (kNN) approach simultaneously evaluates a combination of descriptors without a defined function; however, it assumes that each descriptor used in the model equally impacts the training activity [7, 8].

With a wealth of different algorithms available it is commonly not known which approach works best so several techniques are tried until the most predictive results for a blind test set are achieved [9–12]. This method of evaluation is required because the data structure of the training set is not known a priori. There is a need for a machine learning algorithm that does not require the training data to fit a defined function, incorporates the effects of weighted descriptor combinations, and analyzes all descriptors at the same time. This type of approach would be universal and more predictive because it would not impose restrictions or make assumptions on the data structure. This manuscript discusses such an approach called kScore, where a test data point is assigned an activity prediction based on its weighted similarity to each training set data point. In direct comparison with the most popular machine learning approaches (SVM, kNN, BC, and RP) kScore has consistently performed comparable to or better in terms of blind prediction accuracy [13].

Methods

Optimization function

The goal of kScore has been to develop an algorithm that does not confine the training data to a particular function or kernel. While the kNN algorithm achieves kernel function independence already, the algorithm has other limitations. Each descriptor in the model is forced to have an equal weight, the number of Nearest Neighbors is traditionally fixed, and descriptor selection is optimized with a random sampling algorithm such as Simulated Annealing. In order to surpass these limitations a continuous function has been created such that each dimension or descriptor is assigned a weight that is used to optimize the relative distance from one compound to another. A distance dependent function defines the relative similarity, or level of importance, each training data point has to predict an external compound rather than just using the 'k' Nearest Neighbors. This scheme accommodates changes in the density of the training data, shown below:

$$RS_k = \frac{1 \Big/ \sum_d (w_d(x_{id} - x_{kd}))^2}{\sum_k 1 \Big/ \sum_d (w_d(x_{id} - x_{kd}))^2} \quad (1)$$

where d is the number of dimensions or descriptors describing the training data, k is the number of training data points, RS_k is a value between 0 and 1 that specifies the relative similarity of a test data point, $\vec{x}_i$, to the training data point, $\vec{x}_k$, and w_d is the dth component of the weight vector being optimized, $\vec{w}$. During optimization training compounds with similar activity are positioned close to one another in the weighted space. The descriptors important for explaining the SAR become apparent from the weights (w_d) that optimize to nonzero values. Budagyan et al. [14] have also created a unique way of transposing a descriptor space into a uniform distribution by weighting each training set compound. This scheme is somewhat different than the approach used by kScore where just the density surrounding a compound of interest is considered.

In Eq. 1 the inverse square of the Euclidean distance has been used to calculate the relative similarity for two reasons: (1) to speed up optimization by not requiring a square root in the distance calculations and (2) allowing the RS_k value to approach zero as the distance to a training data point increases. The inverse of the Euclidean distance has been tested; however, it was found not to be as predictive as the inverse square. Other distribution functions could also be used such as a Gaussian or Poisson function. In the instance where the weighted distance between two data points approaches zero their weighted distance is set to a very small value, such as 0.0000001. This prevents the inverse square distance from approaching infinity while still producing a very high weight for that training data point relative to other training data.

In order to assign activity predictions to a test compound the relative similarity is coupled with the known activity of each training compound:

$$y_{iPRED} = \sum_k y_k RS_k \quad (2)$$

Where y_{iPRED} is the predicted activity for test data point i, y_k is the known activity of training data point k, and RS_k is the weighted relative similarity between test compound i and training compound k, determined from Eq. 1. The calculation of RS_k in Eq. 1 is designed such that the sum of the relative similarities for all training compounds is equal to 1. This allows a simple product summation to calculate

the predicted activity of a test compound in Eq. 2. Using this activity prediction the weight vector $\vec{w}$ specifying the importance of each descriptor can be optimized by minimizing the difference between predicted and actual activities in the training set through a Leave One Out (LOO) optimization. To do this the relative similarities calculated from Eq. 1 and the predicted training set activity in Eq. 2 are based on all the training compounds except the compound being left out. For these calculations the value of k in Eq. 1 and 2 becomes the number of compounds in the training set minus 1.

Two strategies extensively used in Support Vector Machines (SVM) that have proven to limit overtraining are the epsilon loss function and a Structure Risk Minimization (SRM) term, $\|w\|^2$. Both strategies are incorporated into kScore resulting in the following loss function:

$$Loss = \frac{\|\vec{w}\|^2}{C} + \frac{1}{k}\sum_{k}\begin{Bmatrix} \text{if} |y_k - y_{kPRED}| > \varepsilon \rightarrow |y_k - y_{kPRED}| - \varepsilon \\ \text{if} |y_k - y_{kPRED}| \leq \varepsilon \rightarrow 0 \end{Bmatrix} \quad (3)$$

C is a prespecified generalization factor that controls the significance of the SRM term relative to the average prediction error, ε is a prespecified acceptable error margin, and *Loss* is the value being minimized during optimization. The inverse of the total number of compounds present in the training set, k, has been added to calculate the average error per training compound so that the optimum values of C remain in a relatively consistent range regardless of the size of the training set. The weight vector $\vec{w}$ is optimized by minimizing the Loss function, i.e. the value of Loss. This has been done using a combination of conjugated gradient and steepest descent methods however a Lagrange multiplier method may also be applicable. In order to increase the optimization speed when a descriptor weight optimizes to zero and remains zero for a large number of iterations, that descriptor is removed from future distance calculations. Depending on the amount of system memory available a portion of the descriptor matrix is also transposed into compound–compound distances since the unweighted distances for each descriptor are constant. This procedure eliminates a significant number of distance calculations performed that would otherwise be redundant. Once the weight vector $\vec{w}$ is solved it can be used by Eq. 2 to predict the activity of any test data point(s).

Classification

In order to assign a class to a test data point the number of training data points in each class must be taken into account. For a two class model where the number of data points in each class is equal, the mean of the two classes can be used as the dividing threshold. Thus for classes 1 and 2, predicted activities above 1.5 would be assigned to class 2 and below 1.5 they would be assigned to class 1. However, most datasets are not equally balanced thus any knowledge based algorithm has to accommodate unbalanced datasets. kScore does this by adjusting the classification assignment threshold between two classes based on their relative sizes, as shown below:

$$\text{Threshold}_{1,2} = \text{Class1} + (\text{Class2 - Class1}) * \left(\frac{\#\ \text{Class2}}{\#\ \text{Class1} + \#\ \text{Class2}}\right) \quad (4)$$

Where $\text{Threshold}_{1,2}$ is the activity threshold between two classes, Class1 is the activity value of the first class, Class2 is the activity value of the second class, #Class1 is the number of data points in the first class, and #Class2 is the number of data points in the second class. This threshold estimate performs very well for datasets that are moderately imbalanced. Figure 1 illustrates how the average test set accuracy for a set of P38 inhibitors varies dramatically as a function of the class activity threshold used to assign test compounds to a specific class.

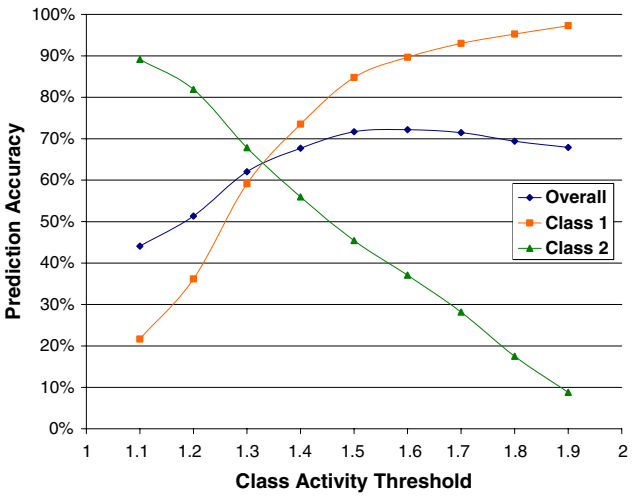

Fig. 1 Average test set classification accuracy for an unbalanced training dataset as a function of the class threshold, with 67% of the training data points having a class assignment of '1' and 33% of the training data points having a class assignment of '2'

For this dataset of P38 inhibitors, adapted from Xiao, et al. [15], an activity threshold was specified such that the compounds were split into two classes. The top third most potent P38 inhibitors were assigned to class '2' and the remaining compounds were assigned to class '1'. kScore classification models were built using 5 random training/test divisions where 30% of the compounds were assigned to the test set. Values of C and ε were varied from 100 to 1500 with a step size of 100 and 0 to 0.6 with a step size of 0.1, respectively. For this study no selection of validated models has been performed [16]. All computed models were used to calculate the average prediction accuracy. Hence the average accuracies are somewhat low. For this dataset the first activity class outnumbers the second class by a ratio of 2:1. From Eq. 2 kScore initially assigns a float prediction to each test compound and the threshold for which compounds are assigned to one class has a dramatic impact on the accuracy as seen in Fig. 1. For this dataset the estimate for the optimum class threshold from Eq. 4 is 1.33. This result agrees well with Fig. 1 that illustrates how to achieve the best balance of prediction accuracy for each class. The advantage of this approach also allows a researcher to achieve higher test accuracies for a desired activity class while sacrificing the prediction accuracy of another by modifying the threshold. For datasets where the percentage of compounds at the activity boundary in the descriptor space varies dramatically between classes a stochastic search may be needed to find the optimum activity threshold. In general, the class activity threshold in Eq. 4 serves as a good estimate of the optimum value.

Applicability domain

Activity models are based on correlations found in the descriptor space occupied by the training set. These correlations most likely do not hold true for regions outside of the training set. Thus external predictions are not considered reliable for data points highly dissimilar from the training set. As a means of improving prediction accuracy an applicability domain is commonly used to restrict the prediction of data points highly dissimilar from the training set. An applicability domain has been developed for kScore that incorporates the optimized descriptor weights found by the model. Descriptors found not to correlate with the training activity are thought to be less important than those used by the kScore model. For this reason the weighted descriptor space found by the kScore model is used to calculate the region covered by the training set rather than the unweighted descriptor space. The distance between any two data points in the kScore optimized descriptor space is calculated as follows:

$$\mathrm{Dist}_{i,k} = \sum_{d} (\mathrm{w}_d(\mathrm{x}_{id} - \mathrm{x}_{kd}))^2 \quad (5)$$

Where $\mathrm{Dist}_{i,k}$ is the weighted square distance between data point i and data point k. If a descriptor does not change for the entire training set kScore will optimize the weight for that descriptor to zero. However, it may still be critical for activity. In this case there is an option to append the distance between two compounds by the weighted difference of that descriptor where the weight used by default is the sum of all descriptor weights divided by the number descriptors that are constant for the training set but differ for the test compound. This causes 50% of the weighted distance to result from the kScore weighted distance and the remaining 50% is a result of deviations from constant descriptor values seen in the training set. This limits the chance a data point completely dissimilar from the training set would be predicted. However, it would also decrease the chance of predicting novel compounds.

In order to calculate the region of descriptor space covered by the training set the pair-wise distance between all training compounds is calculated using Eq. 5. For each training compound the nearest neighbor distance is extracted and the largest nearest neighbor distance found in the training set is used as the default nearest neighbor distance threshold. By using this nearest neighbor distance threshold kScore creates a sphere around each training set compound. This collection of spheres defines the region in the descriptor space covered by the training set. This distance threshold is also the minimum distance required for all training compounds to be predicted in a LOO fashion. For external prediction this distance threshold is made adjustable by using a simple multiplication factor, shown below:

$$\mathrm{Dist\_Threshold} = \mathrm{Default\_Dist}^*(1 + \theta) \quad (6)$$

Where Dist_Threshold is the adjusted applicability domain radii, Default_Dist is the default applicability domain radii, and θ is a multiplication factor specified by the user.

For Classification models a separate nearest neighbor distance threshold is calculated for each activity class. The weighted distance threshold for each class is calculated similar to the method described above with the exception that the maximum nearest neighbor distance is calculated for each activity class rather than

the entire training set. This is useful when for example data points of one class are tightly clustered together and data points from another class are widely distributed throughout the descriptor space. One could also use the weighted nearest neighbor distance for each training data point. Thus each training data point would have a different applicability domain radius depending on the density of training data around that point; however, this technique has not yet been tested.

Rank ordering external databases

When screening a large external database for compounds of a particular activity class a researcher may only have the resources to experimentally test a small portion of the database. In this case it is better suited to prioritize compounds in a database and select the top X% that are predicted to have the desired activity class rather than assigning activity classes to each compound. kScore is very well suited for this task since the activity prediction for a test data point is initially a float value prior to an activity class being assigned (see the Class Activity Threshold discussed earlier). This float activity prediction corresponds to the likelihood a test compound belongs to a particular activity class. For example, if a classification model is built with the activity classes 1 and 2, a test compound with a predicted activity of 1.9 is more likely to belong to Class 2 than a compound with a predicted activity of 1.6.

Given a large compound database with unknown activities a collection of predictive kScore models are used to predict the database in parallel. The predicted float activities are averaged across all models for each compound to arrive at a consensus activity prediction. These consensus activity predictions are used to rank a database based on a compound's likelihood that it belongs to a particular activity class. From this ranking scheme traditional enrichment plots can be easily generated and the top X% of a database can be selected for experimental validation.

Results and discussion

Dopamine D1 antagonists

As a test dataset, the pharmacological data for 48 D1 antagonists has been collected from a previous database screening study [11]. Molconn-Z descriptors [17] were calculated for each compound and linearly normalized to fall within the range zero and one based on the maximum and minimum values of each descriptor (i.e., range-scaled). Descriptor Normalization is required to prevent unequal descriptor weighting during kNN model building. Using these descriptors, regression models were generated in a comparison study using kScore, the k-Nearest Neighbor algorithm (kNN) [7], and Support Vector Machines [3, 11]. The dataset was then randomly subdivided into three sets, a training set (30 compounds), a test set (10 compounds), and a second blind test set (8 compounds). Five separate random divisions of these training/test sets were used for model building and validation. The statistical significance of the models is characterized with the leave-one-out cross-validated R2 (q^2) statistic for the training, test, and blind set. SVM and kNN models were generated similar to a previous study on this same dataset with the same applicability parameters [11]. For kNN models the range of descriptors used in each model has been searched from 10 to 50 with a step size of 2. For each descriptor number, 10 kNN models have been generated to ensure adequate descriptor optimization. For SVM models the value of C has been searched from 50 to 1000 with an increment of 50 and the value of ε is varied from 0 to 1.5 with an increment of 0.1. For kScore model building the exact same search parameters were used for C and ε as SVM's. The default kScore applicability domain radii cutoff was used with θ set equal to zero.

After model building, the top 5% most predictive models for the training and first test set were collected from each method and used to predict the blind test set in a consensus fashion. While the q^2 statistics were quite similar across all three methods for the training set and first test set the predictions for the blind test set varied quite dramatically, as shown in Fig. 2.

Across the three algorithms tested Fig. 2 illustrates that accuracy for a training set and test set does not ensure the model(s) will be as predictive for an external blind test set. The fact that kScore's predictions for this blind test case are more predictive than the training/test sets is a coincidence related to the small number of compounds in that set. Most often predictions made for a blind dataset are equal or less predictive compared to the training set.

kScore's advantage over other popular machine learning approaches is its ability to adapt to any data structure and accommodate changes in the density of the training data points throughout the descriptor space. All current machine learning approaches that perform descriptor optimization require the solution to fit an arbitrary function, impose restrictions on how descriptors may impact a model such as equal weights, or descriptors are considered individually rather than as a whole. kScore does not impose these requirements; rather it allows the model to adopt any structure

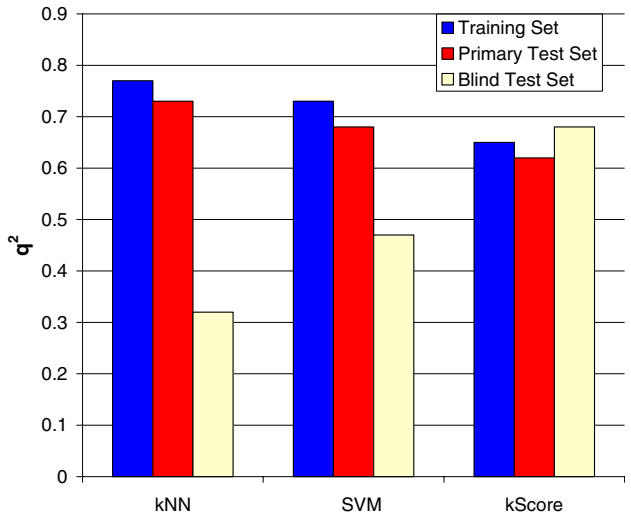

Fig. 2 A comparison study of blind test set predictions for a dataset of D1 antagonists using kScore and two popular machine learning approaches, kNN and SVM's

it chooses while limiting over prediction with the Generalization constraints proposed by Vapnik for SVM's [3] (see Eq. 3). The epsilon loss function used by Vapnik is also incorporated to limit overtraining by accommodating noise in the training activities, or dependent variable. If not taken into account, noise in the dependent variable can result in a suboptimal model that attempts to exactly fit each training data point rather than characterizing more general trends throughout the descriptor space. Since the activities in this D1 antagonist dataset have been tested in triplicate the dependent variable is relatively clean which agrees with the optimized values of ε, 0.5 for SVM's and 0.3 for kScore.

P38 Inhibitors

As a test classification dataset the pharmacological data for 131 P38 mitogen-activated protein (MAP) kinase inhibitors has been adapted from Xiao, et al. [15]. The biological activity associated with each compound is expressed as IC_{50} values for the inhibition of p38 MAP kinase ranging from 0.11 to 114,000 nM. A cutoff of 100 nM was used as a class threshold where 33% of the compounds were grouped into a potent inhibitor class and the remaining 67% of the compounds were classified as weak inhibitors. Molconn-Z descriptors [17] were calculated for each compound and linearly normalized to fall within the range zero and one based on the maximum and minimum values of each descriptor (i.e., range-scaled). The dataset was then randomly subdivided into three sets, a training set (76 compounds), a test set (30 compounds), and a second blind test set (25 compounds). Five separate random divisions of these training/test sets were used for model building and validation. Using these descriptors, classification models were generated in a comparison study using kScore, the k-Nearest Neighbor algorithm (kNN) [7], and Support Vector Machines [3, 11]. The statistical significance of the models is characterized by the average prediction accuracy within each class for the training, test, and blind set. For kNN models the range of descriptors used in each model has been searched from 10 to 50 with a step size of 2. For each descriptor number, 10 kNN models have been generated to ensure adequate descriptor optimization. For SVM models with a linear kernel the value of C has been searched from 50 to 1000 with an increment of 50 and the value of ε is varied from 0 to 1.5 with an increment of 0.1. For kScore model building the exact same search parameters were used for C and ε as SVM's. The default kScore applicability domain radii cutoff was used with θ set equal to zero.

After model building, the top 5% most predictive models for the training and first test set were collected from each method and used to predict the blind test set in a consensus fashion. While the training/test statistics were quite similar across all three methods the predictions for the blind test set varied quite dramatically, as shown in Fig. 3.

Figure 3 illustrates another example that accuracy for a training set and test set does not ensure the model(s) will be as predictive for an external blind test set. If the true SAR within the dataset is not ade-

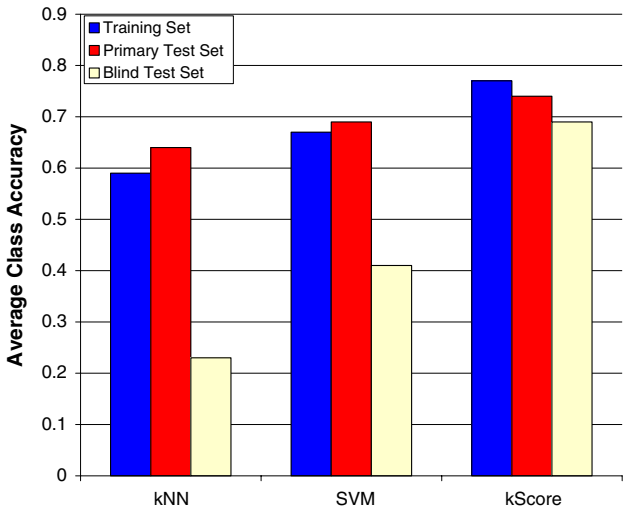

Fig. 3 A comparison study of blind test set predictions for a dataset of P38 inhibitors using kScore and two popular machine learning approaches, kNN and SVM's

quately represented by the kernel or descriptor space used then it is unlikely that a majority of external predictions will also be accurate. This however is difficult to gauge a priori since the training set results can be misleading [16]. kScore attempts to characterize general trends in the data making it applicable to a wider range of datasets.

CoEPrA

For comparison against other well renowned machine learning approaches kScore was used to build classification and regression models in the ''Comparative Evaluation of Predictive Algorithms'' competition (CoEPrA) [13]. A diverse set of machine learning algorithms were also tried by other groups in the competition, such as SVM's, kNN, Recursive Partitioning, Bayesian Classifiers, Gaussian Processes, and Linear Regression. Research groups were given a set of training data with descriptors, dependent variables, and asked to predict the dependent variable for a blind test case. There were four rounds of blind external prediction and between each dataset the rankings varied quite dramatically, kScore consistently was one of the most predictive algorithms. Interestingly, several groups reported using the same machine learning algorithm but arrived at completely different predictions for the same dataset. This illustrates that how a particular algorithm is used can be just as important as the algorithm itself.

In the case of SVM's one major factor that impacts the external prediction is the kernel selected for model building. The best kernel depends on the distribution of data points in the training set and it is very difficult to know a priori which one is most predictive. This is illustrated in both the classification and regression results between different CoEPrA datasets. For one dataset a linear kernel may work very well, however for another dataset that exact same kernel results in blind predictions that are equivalent to random. kScore has the advantage of not restricting the model to a defined kernel making the model building/prediction process easier, predictive, and more stable from one researcher to another.

Several algorithms tested in CoEPrA achieved excellent predictions for one or two datasets but not consistently across all datasets. This finding is corroborated by recent studies using multiple machine learning approaches tested with various descriptor types [9, 10]. Each machine learning approach tested is best suited to fit training data of a particular shape and/or density distribution. It is very difficult to know which method would be the most predictive since a high training accuracy does not guarantee similar results for a test set [16]. kScore has been designed to be very adaptive in the hope of providing consistent results from one dataset to another. This is achieved by allowing the data points in the descriptor space to adopt any shape or trend that best correlates with the training set dependent variable without any restriction(s), while maximizing generalization.

Database enrichment

With the ability to rank order a compound database based on the likelihood that compounds belong to a target class, kScore results can allow one to easily select the top X% of a database rather than merely assigning classes. As a test of enrichment capabilities kScore models were generated for a corporate dataset where ~10,200 training set compounds were available against a specific kinase. In this dataset roughly 17% of the compounds are active inhibitors of the target kinase and the remaining compounds are known inhibitors of other kinases but not active against the target kinase. For comparison Molconn-Z, Daylight Fingerprint (DFP), Atom Pair (AP), and Ghose Crippen (GC) descriptors were calculated for the training set. The dataset was randomly divided into 5 training/test splits whereby the ratio of compounds in the training vs. test set was 3:1. For each descriptor type the training/test splits were identical and used to build kScore models. The optimum values of C were searched from 100 to 1500 with an interval of 100 and the optimum value of ε was searched from 0 to 0.6 with an interval of 0.1. Each model predicted its corresponding test set and the accuracies for each class in the training and test set were collected. Models with accuracies greater than 75% for each class in both the training and test sets were considered predictive.

For this target kinase there is also high throughput screening (HTS) data available on ~775,000 compounds of which ~3,700 were found to be active inhibitors of the kinase. This equates to roughly a 0.5% hit rate. The predictive kScore models for each descriptor type were used to screen the large compound collection for active kinase inhibitors. Float activity predictions were assigned to each database compound using the kScore classification models whereby the float prediction corresponds to a compound's likelihood of belonging to either activity class. The activity predictions were averaged for each database compound across all model predictions and then the compound database was rank ordered based on the predicted likelihood an HTS compound is active. This

process was repeated for models generated with each of the four descriptor types mentioned previously.

From the predicted activity rankings the top X% of a database can be selected for screening or if the true class activities are known then database enrichment plots can be easily generated. In Fig. 4 the database enrichment plot for each descriptor type is shown using kScore.

These results illustrate a 35-fold enrichment over random can be achieved when selecting the top one percent of a database ranked by kScore models. For this dataset AP descriptors appears to narrowly outperform DFP's, followed by Molconn-Z and GC descriptors. Despite the high enrichment rate from Fig. 4 it is difficult to gauge how beneficial kScore actually is because several of the active HTS compounds may be highly similar to active compounds in the training set. To verify that enrichment is still present for both compounds similar and dissimilar from the training set the HTS compounds were divided into two classes. The first class contains compounds with one or more core scaffolds present in the training set (~265,000 compounds) and the second class contains all remaining compounds (~510,000 compounds). When enrichment plots are generated for each of the two HTS compound classes the ability of kScore to rank compounds within the same class and scaffold hopping performance can be analyzed. Figure 5 illustrates the ability of kScore to identify active kinase inhibitors among compounds with scaffolds identical to those in the training set.

Of the ~3,700 active compounds present in the entire HTS dataset ~3,500 active compounds have a

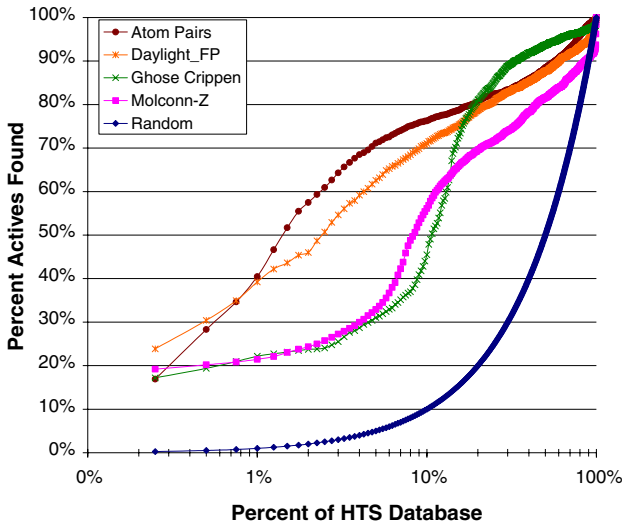

Fig. 4 Database enrichment of a large HTS dataset containing 775,000 compounds using kScore models developed with various descriptor types

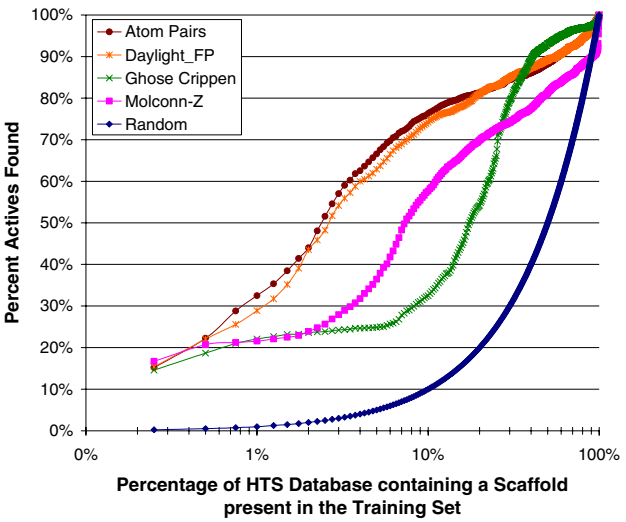

Fig. 5 kScore enrichment of an HTS dataset with ~265,000 compounds that each contain a scaffold present in the training set

scaffold present in the training set. Figure 5 shows that the kScore algorithm performs well prioritizing compounds most likely to be an inhibitor of the target kinase with AP and DFP descriptors performing the best. A common challenge however is to identify compounds with a novel scaffold in the hope that it demonstrates a unique pharmacological profile. This is a difficult challenge for ligand based approaches especially when the descriptors used to describe the molecules are so heavily dependent on the core scaffold [18]. For this type of study a descriptor type that does not take into account the internal connectivity of the molecule may be best suited. In Fig. 6 the enrichment of the remaining ~200 active kinase inhibitors is shown among the ~510,000 HTS compounds that do not contain a scaffold present in the training set.

For scaffold hopping ability the higher levels of enrichment are seen with AP and GC descriptors, while Molconn-Z and DFP descriptors are not much better than random. While simple, the GC descriptors are not designed to differentiate compounds among a common class which explains their poor results in Fig. 5. At the same time GC descriptors identify atomic features prevalent in each activity class. This approach is well suited for scaffold hopping because the atomic connectivity of the core scaffold is not taken into account. AP descriptors likewise perform well because the bond distance between specific features is used rather than characterizing the entire molecule. The use of an applicability domain further improves the results for all descriptor types. As the applicability domain is tightened the results improve in the beginning percentage of the database but then plateau as

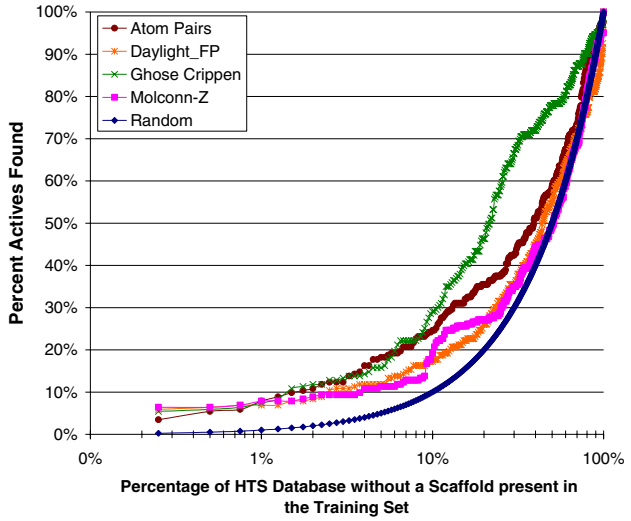

Fig. 6 kScore enrichment of an HTS dataset with ~510,000 compounds that do not contain a scaffold present in the training set

active compounds are excluded by the applicability domain.

These results illustrate that kScore is well suited for compound database enrichment especially when the top X% of a large database is desired. Class predictions made for an entire database may result in compound numbers predicted to have the target property that are much larger than what can be experimentally verified. By prioritizing a database kScore allows one to select a specific number of compounds with the highest chance of being active. This approach can also be useful for prioritizing reagent lists for a combinatorial library or prioritizing compounds series based on their relative average rank.

Conclusions

Based on previous studies illustrating that no single machine learning approach works best for multiple datasets there is a need for a more universal machine learning algorithm. A tool that does not impose restrictions and/or makes assumptions on how the training set descriptors correlate with a target activity would be applicable to a larger range of datasets. Such a tool would significantly reduce the time to build predictive models because only one tool would be needed to obtain near optimum results rather than testing a wide range of approaches. kScore was developed for this purpose and has shown wide applicability to diverse datasets. Results from the CoEPrA competition also illustrate that reasonable prediction accuracy for training and test sets does not always guarantee blind prediction accuracy. The generalization capabilities of kScore attempt to limit overtraining and appear to consistently work well for blind test cases.

The flexibility of the kScore algorithm allows for rank ordering of large libraries of compounds based on their likelihood of belonging to a target class. Enrichment plots show kScore performs well in this regard. As expected there are significant performance variations based on the descriptors used, especially for scaffold hopping. The Class Activity Threshold used by kScore is a novel idea that may be applicable to other similarity based approaches such as kNN. As a dataset becomes biased towards one class over another the predicted activity for a test compound with equal distance to two classes would change based on the density of compounds in each class. kScore by default takes this bias into account and allows a researcher to increase the number of true positives in a particular class while sacrificing the accuracy in another. This behavior is highly beneficial in focused screening studies where a larger number of false positives are acceptable in order to increase the number of active molecules recognized.

References

1. Schlkopf B, Smola AJ (2001) Learning with Kernels. The MIT Press, Cambridge, MA
2. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK
3. Vapnik V (1995) The nature of statistical learning theory. Springer-Verlag New York, Inc., New York
4. Muegge I, Oloff S (2006) Drug Discov.Today Technologies 3:405
5. Fukunaga K (1990) Introduction to statistical pattern recognition. Academic Press, New York
6. Kononenko I (2001) Artif Intell Med 23:89
7. Zheng W, Tropsha A (2000) J Chem Inf Comput Sci 40:185
8. Itskowitz P, Tropsha A (2005) J Chem Inf Model 45:777
9. de Cerqueira LP, Golbraikh A, Oloff S, Xiao Y, Tropsha A (2006) J Chem Inf Model 46:1245
10. Kovatcheva A, Golbraikh A, Oloff S, Xiao YD, Zheng W, Wolschann P, Buchbauer G, Tropsha A (2004) J Chem Inf Comput Sci 44:582
11. Oloff S, Mailman RB, Tropsha A (2005) J Med Chem 48:7322
12. Votano JR, Parham M, Hall LH, Kier LB, Oloff S, Tropsha A, Xie Q, Tong W (2004) Mutagenesis 19:365
13. Comparative Evaluation of Prediction Algorithms (CoEPrA); www.coepra.org. (2006)
14. Budagyan L, Abagyan R (2006) Bioinformatics 22:2597
15. Xiao Z, Varma S, Xiao YD, Tropsha A (2004) J Mol Graph Model 23:129
16. Golbraikh A, Tropsha A (2002) J Mol Graph Model 20:269
17. Molconn-Z Version 4.09 is available from EduSoft (http://www.edusoft-lc.com/molconn/)
18. Zhang Q, Muegge I (2006) J Med Chem 49:1536