Ranking Chemical Structures for Drug Discovery: A New Machine Learning Approach

Shivani Agarwal,*,† Deepak Dugar,‡ and Shiladitya Sengupta§

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, and Department of Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, Massachusetts 02115, USA

Received October 2, 2009

With chemical libraries increasingly containing millions of compounds or more, there is a fast-growing need for computational methods that can rank or prioritize compounds for screening. Machine learning methods have shown considerable promise for this task; indeed, classification methods such as support vector machines (SVMs), together with their variants, have been used in virtual screening to distinguish active compounds from inactive ones, while regression methods such as partial least-squares (PLS) and support vector regression (SVR) have been used in quantitative structure−activity relationship (QSAR) analysis for predicting biological activities of compounds. Recently, a new class of machine learning methods − namely, ranking methods, which are designed to directly optimize ranking performance − have been developed for ranking tasks such as web search that arise in information retrieval (IR) and other applications. Here we report the application of these new ranking methods in machine learning to the task of ranking chemical structures. Our experiments show that the new ranking methods give better ranking performance than both classification based methods in virtual screening and regression methods in QSAR analysis. We also make some interesting connections between ranking performance measures used in cheminformatics and those used in IR studies.

1. INTRODUCTION

The cost of developing a new drug today is estimated to be over $1 billion.[1] A large part of this cost is the result of failed molecules: chemical compounds that appear to be promising drug candidates during initial stages of screening, but after several rounds of expensive preclinical and clinical testing, turn out to be unsuitable for further development. With chemical libraries today containing millions of structures for screening, there is an increasing need for computational methods that can help alleviate some of these challenges.[1−3] In particular, computational tools that can rank chemical structures according to their chances of clinical success can be invaluable in prioritizing compounds for screening: such tools can be used to focus expensive biological testing on a small set of highly ranked, more promising candidates, leading to potentially huge savings in time and costs.

Machine learning methods have already shown considerable promise for this task. In particular, since the early days of chemical informatics, regression methods from machine learning and pattern recognition have been used in quantitative structure−activity relationship (QSAR) analysis to predict biological activities of compounds. These include, for example, partial least-squares (PLS) regression, neural networks, genetic algorithms, regression trees and random forests, and more recently, support vector regression (SVR), all of which have been used extensively in QSAR as well as quantitative structure−property relationship (QSPR) applications.[4−10] In a typical QSAR application, one is given experimentally determined biological activities for a small number of compounds that are known to have a certain level of activity with respect to a particular therapeutic target; a model is then trained to predict the biological activities of new compounds (which are generally are also known/expected to be active with respect to the given target). Often, however, the predicted activities are then used to rank compounds; indeed, as we discuss later, the performance of the learned model is often measured in terms of the quality of the ranking produced.

Machine learning methods have also been used more recently in virtual screening applications, in which the goal is to identify active compounds from large databases containing mostly inactive compounds. In particular, binary classification methods, such as support vector machines (SVMs), have found considerable success.[11−13] In this case, one is given examples of a few compounds that are known to be active with respect to a therapeutic target of interest, together with several compounds that are known to be inactive; a classifier is then trained to distinguish actives from inactives. Again, the end goal is often to obtain a ranking or prioritization of compounds in which actives are ranked higher. Various methods have been proposed for obtaining such a ranking from a learned classifier; for example, in the case of SVMs, in which the classifier consists of a hyperplane in the chemical descriptor space, the signed distance of the point representing a compound from the classifying hyperplane has been used to obtain a ranking of the compounds, with considerable success.[12,13] As we discuss below, in this

* To whom correspondence should be addressed. Email: shivani@mit.edu.
† MIT Computer Science and Artificial Intelligence Laboratory.
‡ MIT Department of Chemical Engineering.
§ Harvard Medical School.

10.1021/ci9003865
Published on Web 04/13/2010

case also, the performance of the learned classifier is generally measured in terms of the quality of the ranking produced.

In recent years, there has been increasing interest in ranking problems in the fields of machine learning and data mining.[14−26] This interest is in part the result of an increasing number of applications of ranking, ranging from ranking documents and web pages in information retrieval (IR) to ranking genes in bioinformatics,[27] and in part because ranking problems are mathematically distinct from the classical learning problems of classification and regression, requiring distinct analysis and distinct algorithms. In particular, several new learning algorithms have been developed in the past few years that are designed to directly optimize ranking performance. For example, in the case of binary or bipartite ranking,[15,19] where there are two categories of objects (such as active and inactive compounds), and the goal is to rank one category (actives) higher than the other (inactives), a number of algorithms have been developed to maximize (some approximation of) the area under the ROC curve (AUC), a popular measure of ranking quality in the bipartite setting. We note that this is distinct from deriving a ranking from a binary classifier trained to minimize classification errors, as is done in the above-referenced works;[12,13] for a detailed discussion of this distinction, see for example the analyses of Cortes and Mohri[28] and Agarwal et al.[19]

In this paper, we report the application of these new ranking methods in machine learning to the task of ranking chemical structures. Specifically, we use kernel-based ranking algorithms that employ a kernel representation similar to SVMs for classification and SVR for regression, but that minimize a ranking loss function rather than a classification or regression loss. We refer to these ranking algorithms as RankSVM. We use a bipartite version of the RankSVM algorithm for virtual screening, which we compare with ranking based on SVM classification; we also use a version of RankSVM that incorporates real-valued labels for ranking compounds in QSAR data sets, which we compare with ranking based on SVR regression. Variants of these RankSVM algorithms were initially developed in the context of IR applications;[16,17] they have since been adapted for a variety of different settings, including bipartite ranking,[29] ranking with real-valued labels,[30] and graph-based ranking.[21] We also note that Wassermann et al.[31] recently used a related ranking method (referred to as preference ranking in their work) to rank target-selective compounds above inactive and nonselective compounds; this ranking method gave the best overall results among those compared. However the broad applicability of ranking methods in machine learning to ranking tasks commonly encountered in virtual screening and QSAR applications appears not to be recognized. In our experiments, we find that the ranking algorithms we use outperform both classification based approaches in virtual screening and regression based approaches in QSAR ranking.

The following sections describe in greater detail the methods we use, followed by our results. We discuss the bipartite ranking setting together with its application to virtual screening in section 2, followed by the setting of ranking with real-valued labels together with its application to QSAR ranking in section 3. In each case, we describe the variant of the RankSVM algorithm we use and give details of the associated data sets and performance measures used in our experiments. Section 4 contains our experimental results together with a discussion; section 5 concludes with a brief summary.

2. BIPARTITE RANKING FOR VIRTUAL SCREENING

2.1. Background. As discussed above, ranking problems have received considerable attention in machine learning in recent years. A particular setting that has been studied in depth is that of bipartite ranking.[15,19] In the bipartite ranking problem, there are two categories of objects, "positive" and "negative"; the learner is given examples of objects labeled as positive or negative, and the goal is to learn a ranking in which positive objects are ranked higher than negative ones. Such problems arise, for example, in information retrieval, where one is interested in retrieving documents from some database that are "relevant" to a given topic; in this case, the training examples given to the learner consist of documents labeled as relevant (positive) or irrelevant (negative), and the goal is to produce a ranking of the documents such that relevant documents are ranked higher than irrelevant documents.

Formally, the setting of the bipartite ranking problem can be described as follows. The learner is given a training sample $S = (S_+, S_-)$ consisting of a sequence of positive examples $S_+ = (\mathbf{x}_1^+, \ldots, \mathbf{x}_m^+)$ and a sequence of negative examples $S_- = (\mathbf{x}_1^-, \ldots, \mathbf{x}_n^-)$, the $\mathbf{x}_i^+$ and $\mathbf{x}_j^-$ being instances in some instance space X, and the goal is to learn a real-valued ranking function $f: X \rightarrow \mathbb{R}$ that ranks accurately future instances in X; in other words, that assigns higher scores to positive instances than to negative ones. The ranking quality of f in this setting is often measured in terms of the bipartite ranking error

$$\mathrm{err}(f;S) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\mathbf{I}_{\{f(\mathbf{x}_i^+)<f(\mathbf{x}_j^-)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i^+)=f(\mathbf{x}_j^-)\}}\right] \quad (1)$$

where $\mathbf{I}_{\{\phi\}}$ is 1 if ϕ is true and 0 otherwise; this is simply the expected fraction of positive-negative pairs mis-ranked by f, assuming that ties are broken uniformly at random. In recent years, a number of learning algorithms have been developed that minimize an approximate version of this bipartite ranking error on the training sample. The bipartite ranking error is in fact simply one minus the AUC, a quantity that is often used to measure the quality of a ranking in the binary/bipartite setting;[19] therefore these ranking algorithms can equivalently be viewed as maximizing (an approximation of) the training AUC. Below we describe one such algorithm that learns a ranking function from a reproducing kernel Hilbert space (RKHS), much as SVMs and SVR do in the case of classification and regression.

2.2. Bipartite RankSVM Algorithm. Because of its discrete nature, the bipartite ranking error in eq 1 cannot be minimized by an algorithm directly. The (bipartite) RankSVM algorithm[16,17,29] minimizes instead a regularized version of the following convex upper bound on the ranking error

$$\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(1 - (f(\mathbf{x}_i^+) - f(\mathbf{x}_j^-)))_+ \quad (2)$$

where

$$a_+ = \begin{cases} a & \text{if} \quad a > 0 \\ 0 & \text{otherwise} \end{cases}$$

This is similar to the use of the hinge loss as a convex upper bound on the zero-one classification loss in SVMs (see Section 2.3) and is depicted in Figure 1.

In this paper, we shall be interested primarily in the case where instances in X are described by d-dimensional vectors (such as word frequency vectors in document ranking or chemical descriptor vectors in virtual screening), so that $X \subseteq \mathbb{R}^d$ for some appropriate d. In this case, given a training sample $S = (S_+, S_-) \in X^m \times X^n$, the linear RankSVM algorithm learns a linear ranking function f: $X \rightarrow \mathbb{R}$ given by $f(\mathrm{x}) = \mathbf{w}\cdot\mathbf{x}$, where the weight vector $\mathbf{w} \in \mathbb{R}^d$ is selected as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left[\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(1 - (\mathbf{w}\cdot\mathbf{x}_i^+ - \mathbf{w}\cdot\mathbf{x}_j^-))_+ + \frac{1}{2C}||\mathbf{w}||^2\right] \tag{3}$$

where $||\mathbf{w}||$ denotes the Euclidean norm of $\mathbf{w}$ and $C > 0$ is an appropriate regularization parameter.

More generally, if $\mathcal{F}$ is an RKHS of real-valued functions on X, the RankSVM algorithm learns a ranking function $f \in \mathcal{F}$ as follows:

$$\min_{f\in\mathcal{F}}\left[\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(1 - (f(\mathbf{x}_i^+) - f(\mathbf{x}_j^-)))_+ + \frac{1}{2C}||f||_{\mathcal{F}}^2\right] \tag{4}$$

where $||f||_{\mathcal{F}}$ denotes the RKHS norm of f in $\mathcal{F}$. In practice, the above optimization problem can be solved by reduction to a convex quadratic program (QP), much as is done in the case of SVMs for classification; in particular, if K: $X \times X \rightarrow \mathbb{R}$ is the kernel function associated with $\mathcal{F}$,[32,33] then the above optimization problem reduces to the following convex QP over mn variables α_{ij} $(1 \le i \le m,\ 1 \le j \le n)$:

$$\min_{\alpha}\left[\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n}\alpha_{ij}\alpha_{kl}(K(\mathbf{x}_i^+, \mathbf{x}_k^+) - K(\mathbf{x}_i^+, \mathbf{x}_l^-) - K(\mathbf{x}_j^-, \mathbf{x}_k^+) + K(\mathbf{x}_j^-, \mathbf{x}_l^-)) - \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_{ij}\right]$$

subject to

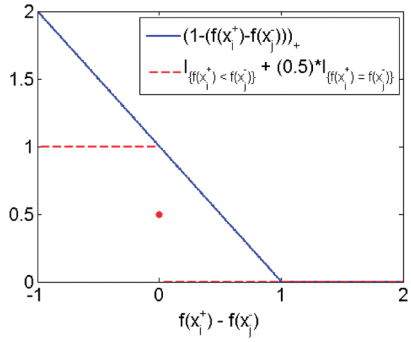

Figure 1. Convex upper bound on the bipartite ranking loss $\mathbf{I}_{\{f(\mathbf{x}_i^+)<f(\mathbf{x}_j^-)\}} + {}^1/_2\mathbf{I}_{\{f(\mathbf{x}_i^+)\ =\ f(\mathbf{x}_j^-)\}}$ of a ranking function f on a positive-negative instance pair $(\mathbf{x}_i^+, \mathbf{x}_j^-)$, as a function of the quantity $(f(\mathbf{x}_i^+) - f(\mathbf{x}_j^-))$.

$$0 \le \alpha_{ij} \le \frac{C}{mn} \text{ for all } i, j \tag{5}$$

Given the solution α to the above QP, the solution to eq 4 is given by

$$f(\mathbf{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_{ij}(K(\mathbf{x}_i^+, \mathbf{x}) - K(\mathbf{x}_j^-, \mathbf{x})) \tag{6}$$

Thus, given a training sample $S = (S_+, S_-) \in X^m \times X^n$ and a kernel function K corresponding to an RKHS $\mathcal{F}$, the bipartite RankSVM algorithm learns a ranking function f: $X \rightarrow \mathbb{R}$ in $\mathcal{F}$ by solving the QP in eq 5, and then constructing f as in eq 6.

While the QP in eq 5 above can be solved using a standard QP solver as is usually done in the case of SVMs, this generally takes $O(m^3n^3)$ time, which can be slow even for moderate-sized data sets. In our experiments, we used a gradient projection algorithm to solve the above QP. In particular, let $Q(\alpha)$ denote the (quadratic) objective function in eq 5 above, and let $\Omega = \{\ \alpha \in \mathbb{R}^{mn}: 0 \le \alpha_{ij} \le C/(mn)$ for all $i, j\}$ be the constraint set. Then the gradient projection algorithm starts at some initial value $\alpha^{(1)}$ for α, and at each iteration t, updates $\alpha^{(t)}$ using a gradient and projection step

$$\alpha^{(t+1)} \leftarrow \mathcal{P}_{\Omega}(\alpha^{(t)} - \eta_t\nabla Q(\alpha^{(t)})) \tag{7}$$

where $\eta_t > 0$ is a learning rate, ∇Q denotes the gradient of Q, and $\mathcal{P}_{\Omega}$ denotes Euclidean projection onto Ω; in particular, for Ω as above, projection onto Ω simply involves clipping values of α_{ij} outside the interval $[0, C/(mn)]$ to the interval. It is well-known from standard results in the optimization literature[34] that for a convex optimization problem, if $\eta_t = \eta/(\sqrt{t})$ for some constant $\eta > 0$, then gradient projection converges to an optimal solution, and in particular, reaches a solution whose objective value is within ε of the optimal in $O(1/\epsilon^2)$ iterations. In our case, each iteration takes $O((m + n)^2)$ time, and therefore the gradient projection algorithm reaches an ϵ-optimal solution to eq 5 in $O((m + n)^2/\epsilon^2)$ time. The algorithm is summarized in Figure 2.

2.3. SVM-Based Ranking. An alternative to the bipartite RankSVM algorithm is to learn a classifier consisting of a real-valued function f: $X \rightarrow \mathbb{R}$ and a threshold $b \in \mathbb{R}$ using the SVM algorithm and then simply use f to rank objects in X. Specifically, given a training sample $S = (S_+, S_-) \in X^m \times X^n$, one can train an SVM classifier on the equivalent labeled sample $S' = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_{m+n}, y_{m+n}))$ where

$$(\mathbf{x}_k, y_k) = \begin{cases} (\mathbf{x}_k^+, +1) & \text{for} \quad k = 1,...,m \\ (\mathbf{x}_{k-m}^-, -1) & \text{for} \quad k = m+1,...,m+n \end{cases}$$

Given such a labeled sample S' and a kernel function K corresponding to an RKHS $\mathcal{F}$, the SVM algorithm learns a classifier $h : X \rightarrow \{-1, +1\}$ given by

$$h(\mathbf{x}) = \text{sign}(f(\mathbf{x}) + b)$$

where

$$\text{sign}(u) = \begin{cases} +1 & \text{if} \quad u \ge 0 \\ -1 & \text{otherwise} \end{cases}$$

and where $f \in \mathcal{F}$ and $b \in \mathbb{R}$ are selected as follows:

$$\min_{f\in\mathcal{F},b\in R}\left[\frac{1}{m+n}\sum_{k=1}^{m+n}(1-y_k(f(\mathbf{x}_k)+b))_+ + \frac{1}{2C}||f||^2_{\mathcal{F}}\right] \quad (8)$$

Here the hinge loss on the kth example

$$(1-y_k(f(\mathbf{x}_k)+b))_+$$

serves as a convex upper bound on the zero-one classification loss

$$\mathbf{I}_{\{y_k(f(\mathbf{x}_k)+b)<0\}}$$

This is depicted in Figure 3. As noted above, the standard approach to solving the above optimization problem is via reduction to a convex QP.[35]

The above approach of using the SVM classification algorithm to learn a ranking function f has been used in virtual screening by Jorissen and Gilson[12] and Geppert et al.,[13] with considerable success. However, as can be seen, the function f in this case is actually chosen to optimize classification accuracy, whereas the RankSVM algorithm described in section 2.2 selects f to optimize ranking performance.

2.4. Virtual Screening Data Sets. In our virtual screening experiments, we used the data sets used by Jorissen and Gilson,[12] which consist of five sets of chemical compounds that each target a different protein, and a background set of compounds that are assumed to be inactive. The five active sets contain 50 compounds each in the following classes: reversible inhibitors of cyclin-dependent kinase 2 (CDK2), cyclooxygenase-2 (COX2), factor Xa (FXa), and phosphodiesterase-5 (PDE5), and reversible antagonists of the α_{1A} adrenoceptor (α_{1A}AR). The inactive set contains 1892 compounds drawn from the National Cancer Institute (NCI) diversity set. Details of these data sets can be found in the original reference.[12]

Given the prevalence of binary molecular fingerprints as chemical descriptors in virtual screening,[13,31,36,37] we chose to represent each compound in the above data sets using two

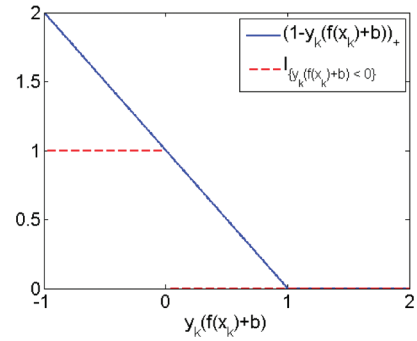

Figure 3. Convex upper bound on the zero-one classification loss $\mathbf{I}_{\{y_k(f(\mathbf{x}_k)+b)<0\}}$ of a classifier $h(\mathbf{x}) = \text{sign}(f(\mathbf{x}) + b)$ on an example $(\mathbf{x}_k, y_k)$, as a function of the quantity $y_k(f(\mathbf{x}_k) + b)$.

different types of fingerprints. The first of these is the Molprint2D fingerprint,[38,39] which has been shown recently to give superior performance compared to other types of fingerprints in SVM-based ranking applications.[13,31,37] The Molprint2D fingerprint associated with a compound consists of a set of strings describing the atom environments present in the compound; as in the above studies,[13,31,37] we converted these to bit vectors by enumerating all the atom environment strings present in our data sets and assigning a unique fingerprint position to each such string; for each compound, the bit corresponding to a given string was then set to 1 if the associated atom environment was present in the compound, and 0 otherwise. This resulted in a fingerprint representation with a total of 17 221 bit positions. The second fingerprint we used is the basic FP2 fingerprint available with the OpenBabel chemical informatics software package.[40] For each compound, the FP2 fingerprint indexes small molecular fragments (of up to 7 atoms) using a hash code that generates a total of 1021 bits.

We compared the performance of the bipartite RankSVM algorithm with that of the SVM-based ranking method on the above data sets. For each of the five targets, we had 50 active compounds, and a total of 2092 inactives (including the 1892 background compounds and the 200 compounds belonging to the other four active sets). We considered three

Algorithm Bipartite RankSVM – Gradient Projection Algorithm

Inputs:

Training sample $S = (S_+, S_-) \in X^m \times X^n$

Kernel function $K : X \times X \to \mathbb{R}$

Parameters C, T, η

Initialize:

$\alpha_{ij}^{(1)} \leftarrow \frac{C}{1000mn}$ $\forall 1 \le i \le m, 1 \le j \le n$ (initialize to some small values)

For $t = 1$ to T do:

- [Gradient step]

  $\boldsymbol{\alpha}^{(t+1/2)} \leftarrow \boldsymbol{\alpha}^{(t)} - \frac{\eta}{\sqrt{t}}\nabla Q(\boldsymbol{\alpha}^{(t)})$

- [Projection step]

  For $i = 1$ to m do:

    For $j = 1$ to n do:

      If $(\alpha_{ij}^{(t+1/2)} < 0)$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow 0$

      Else If $(\alpha_{ij}^{(t+1/2)} > \frac{C}{mn})$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow \frac{C}{mn}$

      Else $\alpha_{ij}^{(t+1)} \leftarrow \alpha_{ij}^{(t+1/2)}$

Output:

$$f(\mathbf{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_{ij}^{(t^*)}(K(\mathbf{x}_i^+, \mathbf{x}) - K(\mathbf{x}_j^-, \mathbf{x})), \quad \text{where } t^* = \arg\min_{1\le t\le T+1} Q(\boldsymbol{\alpha}^{(t)}).$$

Figure 2. Fast bipartite RankSVM algorithm based on the gradient projection method. Here Q refers to the objective function in eq 5.

different ways to split these into training and test sets, two of which were used in the original study by Jorissen and Gilson.[12] In particular, Jorissen and Gilson listed the 50 active compounds for each target in a manner such that compounds with similar chemistries were grouped together. They then split the compounds in two ways. In the first split, the top 25 compounds in each list were separated from the bottom 25, with one-half going to the training set and the other half going to the test set; thus in this case, the active compounds in the training and test sets are mostly quite different. In the second split, the odd-numbered compounds in each list were separated from the even-numbered compounds, again with one-half going to the training set and the other half going to the test set; in this case, the active compounds in the training and test sets are highly similar. In each case, the background compounds were partitioned in a fixed manner (odd-numbered entries in a fixed ordering were separated from even-numbered entries), with half the compounds going to the training set and half going to the test set.

The above splits of the compounds into training and test sets are both extremes, representing the worst-case and best-case possibilities, respectively. In practice, the training set is likely to be more representative of the test set than in the first (worst-case) split above, and less representative than in the second (best-case) split. To simulate this, we also considered a third split, in which the active and background sets were each partitioned randomly, with one-half going to the training set and the other half going to the test set. As above, for each target, this resulted in training and test sets each containing 25 actives and 1046 inactives. In addition, to evaluate the impact of the training set size, we conducted experiments in which 20%, 40%, 60%, 80%, and 100% of the training compounds were used. This process was repeated with 10 different random partitions of the compounds into training and test sets.

2.5. Performance Measures. For each train/test split of the virtual screening data sets described above, the performance of the ranking function learned by each algorithm on the training set was evaluated on the corresponding test set. The following performance measures were used:

1. Bipartite ranking error: This is the quantity (approximately) optimized by the bipartite RankSVM algorithm. Specifically, given a ranking function f: $X \rightarrow \mathbb{R}$ and a test sample $T = (T_+, T_-)$ consisting of m positive instances $T_+ = (\mathbf{x}_1^+, ..., \mathbf{x}_m^+) \in X^m$ and n negative instances $T_- = (\mathbf{x}_1^-, ..., \mathbf{x}_n^-) \in X^n$, the bipartite ranking error of f with respect to T measures the fraction of positive-negative pairs in T mis-ranked by f, and is given by

$$\mathrm{err}(f;T) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\mathbf{I}_{\{f(\mathbf{x}_i^+)<f(\mathbf{x}_j^-)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i^+)=f(\mathbf{x}_j^-)\}}\right]$$

2. Area under the ROC curve (AUC): This is simply one minus the bipartite ranking error; we include this measure since it has the intuitive interpretation of being the fraction of positive-negative pairs in T that are ranked correctly by f, and is widely used as a ranking performance measure in the binary setting.[19,41]

$$\mathrm{AUC}(f;T) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\mathbf{I}_{\{f(\mathbf{x}_i^+)>f(\mathbf{x}_j^-)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i^+)=f(\mathbf{x}_j^-)\}}\right]$$

3. Average precision: The precision at a given position r is widely used in information retrieval to measure the concentration of relevant documents in the top r documents in a ranked list. In our case, the precision of f at position r in T (for $1 \le r \le m + n$) is defined as the proportion of actives in the top r compounds returned by f

$$\mathrm{prec}_r(f;T) = \frac{1}{r}\sum_{i=1}^{r}\mathbf{I}_{\{y_{\pi^{-1}(i)}=+1\}}$$

where $y_{\pi^{-1}(i)}$ is the label (+1 for active and −1 for inactive) of the ith ranking compound returned by f. In particular, if $T' = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_{m+n}, y_{m+n}))$ is constructed from T according to

$$(\mathbf{x}_k, y_k) = \begin{cases} (\mathbf{x}_k^+, +1) & \text{for } k = 1,...,m \\ (\mathbf{x}_{k-m}^-, -1) & \text{for } k = m+1,...,m+n \end{cases}$$

then we have used $\pi(k)$ to denote the position of $\mathbf{x}_k$ in the ranking returned by f, and $\pi^{-1}(i)$ to denote the index in T' of the ith ranking compound returned by f.

The precision is closely related to the enrichment factor (EF) used, for example, by Jorissen and Gilson[12] to evaluate rankings in virtual screening. In particular, the enrichment factor at a given position r measures the proportion of actives in the top r compounds relative to the overall proportion of actives in the complete sample:

$$\mathrm{EF}_r(f;T) = \frac{\mathrm{prec}_r(f;T)}{\mathrm{prec}_{m+n}(f;T)}$$

The precision and enrichment factor measure the retrieval quality at a specific cutoff point. The average precision (AP) has been used to measure the overall quality of a ranking and, in our case, is defined as the average of the precision values at the positions corresponding to active compounds:

$$\mathrm{AP}(f;T) = \frac{1}{m}\sum_{r=1}^{m+n}[\mathrm{prec}_r(f;T) \times \mathbf{I}_{\{y_{\pi^{-1}(r)}=+1\}}]$$

Unlike the bipartite ranking error and AUC, which measure the global quality of a ranking, the average precision emphasizes ranking quality at the top of a ranked list. This is often important for retrieval-type applications (for example, in web search, the quality of the webpages returned at the top of the list is of paramount importance); consequently, the average precision is popular as a performance measure in information retrieval.

We note that if the ranking function f assigns distinct scores to all the compounds in T, then the average precision can also be written as

$$\mathrm{AP}(f;T) = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\sum_{j=1}^{m}\mathbf{I}_{\{f(\mathbf{x}_j)\ge f(\mathbf{x}_i)\}}}{\sum_{k=1}^{m+n}\mathbf{I}_{\{f(\mathbf{x}_k)\ge f(\mathbf{x}_i)\}}}\right]$$

This bears some resemblance to the modified enrichment factor used by Jorissen and Gilson,[12] which when

evaluated over the complete sample is inversely proportional to the average rank (AR) of the actives.

$$\mathrm{EF}'(f;T) = \frac{m+n}{2\mathrm{AR}(f;T)} = \frac{m+n}{\frac{2}{m}\sum_{i=1}^{m}\left[\sum_{k=1}^{m+n}\mathbf{I}_{\{f(\mathbf{x}_k)\geq f(\mathbf{x}_i)\}}\right]}$$

4. Number of actives in top 25 compounds: A simple measure of the quality of a ranking at a cutoff r is the number of actives returned in the top r compounds

$$\mathrm{act}_r(f;T) = \sum_{i=1}^{r}\mathbf{I}_{\{y_{\pi^{-1}(i)}=+1\}}$$

We report the number of actives in the top 25 compounds, $\mathrm{act}_{25}(f; T)$.

Note that the precision at r (see above) is simply

$$\mathrm{prec}_r(f;T) = \frac{\mathrm{act}_r(f;T)}{r}$$

Together with precision, another measure frequently reported when evaluating a ranking at a cutoff r is the recall, which is the proportion of all actives in the sample that are returned in the top r compounds

$$\mathrm{recall}_r(f;T) = \frac{\mathrm{act}_r(f;T)}{m}$$

In our data sets, the total number of actives in each test sample is $m = 25$ (see section 2.4). In an ideal ranking in our case, all 25 active compounds would be placed above all the inactive compounds, which would give $\mathrm{act}_{25}(f; T) = 25$.

5. Number of actives in top 100 compounds: We also report the number of actives in the top 100 compounds, $\mathrm{act}_{100}(f; T)$. Since the total number of actives in each test sample in our case is $m = 25$ (see above), we have

$$\mathrm{act}_{25}(f;T) \leq \mathrm{act}_{100}(f;T) \leq 25$$

3. RANKING WITH REAL-VALUED LABELS FOR QSAR DATA

3.1. Background. In many cases, one has a collection of compounds that are all known (or expected) to have some degree of activity with respect to a particular target, and it is of interest to construct a fine-grained ranking among these compounds to identify the most promising candidates. In such cases, one is given experimentally determined biological activities for a small number of compounds in the collection, and the goal is to rank the remaining compounds such that those with greater activities are ranked higher. Thus, unlike virtual screening, where the compounds are associated with binary active/inactive labels, the compounds to be ranked here are associated with real-valued activity labels.

The problem of ranking with real-valued labels has been studied recently by Agarwal and Niyogi.[30] The general setting in this case can be described as follows. The learner is given a labeled training sample $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$, the $\mathbf{x}_i$ being instances in some instance space X and the y_i being real-valued labels denoting the "relevance" of the corresponding $\mathbf{x}_i$, and the goal is to learn a ranking function $f: X \rightarrow \mathbb{R}$ that ranks accurately future instances in X; in other words, that assigns higher scores to more relevant instances than to less relevant ones. The ranking quality of f in this setting can be measured in terms of the following relevance-weighted ranking error

$$\mathrm{err}(f;S) = \frac{1}{|P|}\sum_{(i,j)\in P}(y_i - y_j)\left[\mathbf{I}_{\{f(\mathbf{x}_i)<f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}_j)\}}\right] \quad (9)$$

where

$$P = \{(i,j)|y_i > y_j\}$$

denotes the set of "preference pairs" in S. Thus, mis-ranking a pair of instances with a large difference in relevance values (pair of compounds with a large difference in activities) incurs a larger loss or penalty than mis-ranking a pair of instances with similar relevance values (pair of compounds with similar activities). Below we describe a variant of the RankSVM algorithm that minimizes an approximate version of this ranking error on the training sample.

3.2. RankSVM Algorithm for Real-Valued Labels. As in the bipartite case, the ranking error in eq 9 cannot be minimized directly. The RankSVM algorithm for this setting[30] minimizes a regularized version of the following convex upper bound on the ranking error

$$\frac{1}{|P|}\sum_{(i,j)\in P}((y_i - y_j) - (f(\mathbf{x}_i) - f(\mathbf{x}_j)))_+ \quad (10)$$

This is depicted in Figure 4. In particular, given a training sample $S \in (X \times \mathbb{R})^m$ as above and an RKHS $\mathcal{F}$ of real-valued functions on X, the algorithm selects a ranking function $f \in \mathcal{F}$ as follows:

$$\min_{f\in\mathcal{F}}\left[\frac{1}{|P|}\sum_{(i,j)\in P}((y_i - y_j) - (f(\mathbf{x}_i) - f(\mathbf{x}_j)))_+ + \frac{1}{2C}\|f\|_{\mathcal{F}}^2\right] \quad (11)$$

It can be shown that if $K: X \times X \rightarrow \mathbb{R}$ is the kernel function associated with $\mathcal{F}$, then the above optimization problem reduces to the following convex QP in $|P|$ variables α_{ij} (for $(i, j) \in P$)

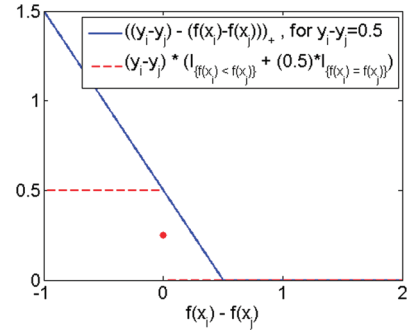

Figure 4. Convex upper bound on the relevance-weighted loss $(y_i - y_j)[\mathbf{I}_{\{f(\mathbf{x}_i) < f(\mathbf{x}_j)\}} + {}^1/_2\mathbf{I}_{\{f(\mathbf{x}_i) = f(\mathbf{x}_j)\}}]$ of a ranking function f on a pair of examples $(\mathbf{x}_i, y_i),(\mathbf{x}_j, y_j)$ with $y_i > y_j$, illustrated for $(y_i - y_j) = 0.5$, as a function of the quantity $(f(\mathbf{x}_i) - f(\mathbf{x}_j))$.

$$\min_{\alpha}\Big[\frac{1}{2}\sum_{(i,j)\in P}\sum_{(k,l)\in P}\alpha_{ij}\alpha_{kl}(K(\mathbf{x}_i,\mathbf{x}_k) - K(\mathbf{x}_i,\mathbf{x}_l) - K(\mathbf{x}_j,\mathbf{x}_k) + K(\mathbf{x}_j,\mathbf{x}_l)) - \sum_{(i,j)\in P}\alpha_{ij}(y_i - y_j)\Big]$$

subject to

$$0 \le \alpha_{ij} \le \frac{C}{|P|} \text{ for all } (i,j) \tag{12}$$

Given the solution α to the above QP, the solution to eq 11 is given by

$$f(\mathbf{x}) = \sum_{(i,j)\in P}\alpha_{ij}(K(\mathbf{x}_i,\mathbf{x}) - K(\mathbf{x}_j,\mathbf{x})) \tag{13}$$

Thus, given a training sample $S \in (X \times \mathbb{R})^m$ and a kernel function K corresponding to an RKHS $\mathcal{F}$, the RankSVM algorithm for the real-valued labels setting learns a ranking function f: $X \to \mathbb{R}$ in $\mathcal{F}$ by solving the QP in eq 12, and then constructing f as in eq 13.

Solving the QP in eq 12 using a standard QP solver takes $O(|P|^3)$ time, which can be as high as $O(m^6)$ for $|P| = O(m^2)$. As in the bipartite case, in our experiments, we used a faster gradient projection algorithm to solve the above QP. As before, the gradient projection algorithm reaches an ϵ-optimal solution in $O(1/\epsilon^2)$ iterations; in this case, each iteration takes $O(m^2)$ time, and therefore the algorithm reaches an ϵ-optimal solution in $O(m^2/\epsilon^2)$ time (see section 2.2). The algorithm is summarized in Figure 5.

3.3. SVR-Based Ranking. An alternative to RankSVM is to simply use a regression algorithm such as SVR to learn a prediction model consisting of a real-valued function f: $X \to \mathbb{R}$ that predicts the labels of new instances in X (activities of new compounds) and then ranks the instances in decreasing order of predicted labels (predicted activities).

Regression algorithms, including SVR, as well as several others, have been used widely in QSAR prediction tasks; in particular, such algorithms are frequently used to find structural features of chemical compounds that correlate with biological activities, as a means to better understand the structure of potential drug candidates in relation to the activity being studied. If the goal is to predict activities or to find structural features that help in such prediction, then learning a prediction model using regression methods is indeed the right approach. However, as we shall see, if the goal is to rank compounds in decreasing order of activity, then the criteria optimized by SVR or other regression algorithms may be suboptimal.

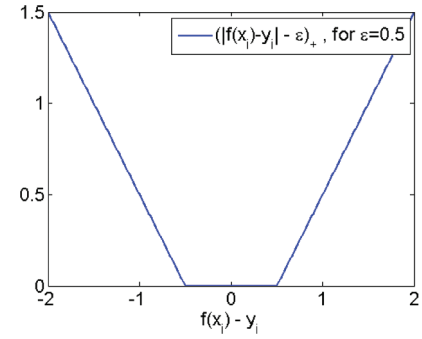

Figure 6. ϵ-Insensitive loss of a prediction function f on an example $(\mathbf{x}_i, y_i)$, illustrated for $\epsilon = 0.5$, as a function of the quantity $(f(\mathbf{x}_i) - y_i)$.

In particular, given a training sample $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (X \times \mathbb{R})^m$ and a kernel function K corresponding to an RKHS $\mathcal{F}$, the SVR algorithm learns a prediction function f: $X \to \mathbb{R}$ as follows:

$$\min_{f\in\mathcal{F}}\left[\frac{1}{m}\sum_{i=1}^{m}(|f(\mathbf{x}_i) - y_i| - \epsilon)_+ + \frac{1}{2C}|f|^2_{\mathcal{F}}\right] \tag{14}$$

where ϵ is a sensitivity parameter. Here the term

$$(|f(\mathbf{x}_i) - y_i| - \epsilon)_+$$

is the ϵ-insensitive loss on the ith example and is a measure of the prediction error of f on $(\mathbf{x}_i, y_i)$, which is zero if $|f(\mathbf{x}_i) - y_i| < \epsilon$ and grows with $|f(\mathbf{x}_i) - y_i|$ otherwise; this is depicted in Figure 6. As with SVMs, the standard approach to solving the above optimization problem is via reduction to a convex QP.[42]

Algorithm RankSVM for Real-Valued Labels – Gradient Projection Algorithm

Inputs:

Training sample $S \in (X \times \mathbb{R})^m$

Kernel function $K : X \times X \to \mathbb{R}$

Parameters C, T, η

Initialize:

$\alpha_{ij}^{(1)} \leftarrow \frac{C}{1000|P|}\ \forall (i,j) \in P$ (initialize to some small values)

For $t = 1$ to T do:

- [Gradient step]

  $\boldsymbol{\alpha}^{(t+1/2)} \leftarrow \boldsymbol{\alpha}^{(t)} - \frac{\eta}{\sqrt{t}}\nabla Q(\boldsymbol{\alpha}^{(t)})$

- [Projection step]

  For $(i,j) \in P$ do:

  If $(\alpha_{ij}^{(t+1/2)} < 0)$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow 0$

  Else If $(\alpha_{ij}^{(t+1/2)} > \frac{C}{|P|})$ **Then** $\alpha_{ij}^{(t+1)} \leftarrow \frac{C}{|P|}$

  Else $\alpha_{ij}^{(t+1)} \leftarrow \alpha_{ij}^{(t+1/2)}$

Output:

$f(\mathbf{x}) = \sum_{(i,j)\in P}\alpha_{ij}^{(t^*)}(K(\mathbf{x}_i,\mathbf{x}) - K(\mathbf{x}_j,\mathbf{x}))$, where $t^* = \arg\min_{1\le t\le T+1} Q(\boldsymbol{\alpha}^{(t)})$.

Figure 5. Fast RankSVM algorithm for real-valued labels based on the gradient projection method. Here Q refers to the objective function in eq 12.

As discussed above, SVR regression has been used to obtain an implicit ranking of compounds in QSAR data sets. However, as can be seen, the function f in this case is actually chosen to optimize prediction accuracy, while the RankSVM algorithm described in section 3.2 selects f to optimize ranking performance.

3.4. QSAR Data Sets. In our QSAR ranking experiments, we used two QSAR data sets used by Sutherland et al.,[10] which consist of inhibitors of dihydrofolate reductase (DHFR) and cyclooxygenase-2 (COX2), respectively, together with corresponding biological activities represented as pIC_{50} values. The DHFR inhibitor data set contains 361 compounds, with pIC_{50} values ranging from 3.3 to 9.8; the COX2 inhibitor data set contains 282 compounds, with pIC_{50} values ranging from 4.0 to 9.0.

Each compound in the above data sets was represented using two different types of descriptors. The first of these are the 2.5D chemical descriptors used in the study by Sutherland et al.[10] These include 2D descriptors that are calculated simply from the connection graph of a molecule (such as χ indices, counts of rotatable bonds, molecular weight, and E-state indices), as well as whole-molecule 3D descriptors (such as molecular volume and charged partial surface area descriptors); details of these descriptors can be found in ref 10 and the references therein. The DHFR inhibitor data set contains 70 real-valued descriptors; the COX2 inhibitor data set contains 74 real-valued descriptors. In our experiments, each of these descriptors was scaled to lie between 0 and 1. The second type of descriptors we used are the OpenBabel[40] FP2 fingerprints, which as described in section 2.4, result in a vector of 1021 bits for each compound.

We compared the performance of the RankSVM algorithm with that of the SVR-based ranking method on the above data sets. We considered two different types of train/test splits of these data sets. The first split was used in the original study by Sutherland et al.;[10] in this case, for each data set, approximately one-third of the compounds were selected using a maximum dissimilarity algorithm and assigned to the test set, while the remaining two-thirds were assigned to the training set. This resulted in a training set of 237 compounds and test set of 124 compounds for the DHFR inhibitor data set and a training set of 188 compounds and test set of 94 compounds for the COX2 inhibitor data set.

Again, the above train/test split represents a worst-case possibility, with the test set being maximally diverse and requiring considerable extrapolation from the training set. We also considered a random split, in which each data set was partitioned randomly into training and test sets of the same sizes as above. In addition, to evaluate the impact of the training set size, we conducted experiments in which 10%, 20%, 30%, ..., 100% of the training compounds were used. This process was repeated with 10 different random partitions of the compounds into training and test sets.

3.5. Performance Measures. For each train/test split of the QSAR data sets described above, the performance of the ranking function learned by each algorithm on the training set was evaluated on the corresponding test set. The following performance measures were used:

1. Ranking error: This is the quantity (approximately) optimized by the RankSVM algorithm in the real-valued labels setting described above. Specifically, given a ranking function $f: X \rightarrow \mathbb{R}$ and a test sample $T = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)) \in (X \times \mathbb{R})^m$, the ranking error of f with respect to T measures the average relevance-weighted ranking loss of f on preference pairs in T

$$\mathrm{err}(f;T) = \frac{1}{|P|} \sum_{(i,j)\in P} (y_i - y_j)\left[\mathbf{I}_{\{f(\mathbf{x}_i)<f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}_j)\}}\right]$$

where $P = \{(i, j) \mid y_i > y_j\}$ denotes the set of preference pairs in T.

2. Correlation: Two performance measures that have frequently been used in evaluating QSAR models are the root mean squared error (RMSE) and correlation.[8] The RMSE measures the predictive (in)accuracy of a model; the correlation, on the other hand, can be viewed as a measure of ranking performance. Specifically, the (Pearson) correlation between the vector $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_m))$ of predicted activities (or in our case, simply scores assigned by the learned function) and the vector $\mathbf{y} = (y_1, ..., y_m)$ of actual activities is given by

$$\mathrm{corr}(f;T) = \frac{\frac{1}{m-1}\sum_{i=1}^{m}(f(\mathbf{x}_i) - \mu_f)(y_i - \mu_y)}{\sigma_f \sigma_y}$$

where μ_f, σ_f are the mean and standard deviation of $\mathbf{f}$, respectively, and μ_y, σ_y are similarly the mean and standard deviation of $\mathbf{y}$, respectively. As is well-known, this measures the strength of a linear relationship between the two vectors. In particular, if the relative differences in the scores/activities of compounds under the two vectors are roughly proportional to each other (resulting in a similar ranking), the correlation will be high. Note however that if the rankings are similar but the scores have a nonlinear relationship, then the correlation may not be an accurate measure of ranking performance. The correlation lies between -1 and 1, with 1 representing a perfect positive linear relationship and -1 representing a perfect negative linear relationship.

3. Kendall's τ rank correlation coefficient: Kendall's τ[43] is a popular rank correlation coefficient used to measure the agreement between two rankings; it effectively measures the fraction of pairs of objects on which two rankings agree. In our case, it was used to measure the expected agreement between the learned ranking of the compounds in T and the true ranking based on their biological activities, assuming that ties in the learned ranking are broken uniformly at random:

$$\tau(f;T) = 2\left(\frac{1}{|P|}\sum_{(i,j)\in P}\left[\mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}_j)\}}\right]\right) - 1$$

The Kendall τ coefficient takes a value between -1 and 1, with 1 representing perfect agreement between the rankings and -1 representing perfect disagreement.

4. Spearman's ρ rank correlation coefficient: Spearman's ρ[44] is another popular rank correlation coefficient used to measure the agreement between two rankings; it measures the standard Pearson correlation between the vectors of ranks of objects (i.e., their positions in sorted order) resulting from two rankings. In our case, it was used to measure the correlation between the learned ranking of the compounds in T and the true ranking based on their biological activities. In particular, if $\beta_f(i)$ denotes the rank of $\mathbf{x}_i$ in the ranking returned by f (such that compounds

that are assigned the same score by f receive the average rank among them) and $\beta_y(i)$ denotes similarly the rank of $\mathbf{x}_i$ in the ranking based on the actual activities, then the Spearman ρ coefficient is given by the Pearson correlation between the vectors $\beta_f = (\beta_f(1), ..., \beta_f(m))$ and $\beta_y = (\beta_y(1), ..., \beta_y(m))$

$$\rho(f;T) = \frac{\frac{1}{m-1}\sum_{i=1}^{m}(\beta_f(i) - \mu'_f)(\beta_y(i) - \mu'_y)}{\sigma'_f \sigma'_y}$$

where μ'_f and σ'_f are the mean and standard deviation of β_f, respectively, and μ'_y and σ'_y are similarly the mean and standard deviation of β_y, respectively. The Spearman ρ coefficient also takes a value between -1 and 1, with 1 representing perfect agreement and -1 representing perfect disagreement.

5. Normalized discounted cumulative gain (NDCG): The NDCG[45] is a popular measure of ranking performance in information retrieval, where the relevance of the top few items returned by a ranking (such as the top few web pages returned in a web search) is especially important, and is used instead of the average precision (see section 2.5) when there are more than two possible relevance levels. As with the average precision, the NDCG places greater emphasis on ranking accuracy at the top of a ranked list: starting from the top, it accumulates a gain value for each object in the list that is based on the relevance of the object, but applies a (logarithmic) discounting factor that reduces the gain for objects lower down in the list. In our case, the NDCG was used to evaluate the ranked list of compounds in T returned by f; the gain value of each compound in the list was based on the biological activity of the compound. In particular, if $\pi^{-1}(i)$ denotes the index of the compound in T that appears in the ith position in the ranking returned by f, then the NDCG is given by

$$\mathrm{NDCG}(f;T) = \frac{1}{Z}\sum_{i=1}^{m}\frac{1}{\log_2(i+1)} \cdot (2^{y_{\pi^{-1}(i)}} - 1)$$

where $1/\log_2(i+1)$ is the discounting factor for the ith-ranking compound, and Z is a normalization constant chosen so that the NDCG is at most 1. Higher values of the NDCG correspond to better ranking performance.

4. RESULTS AND DISCUSSION

4.1. Virtual Screening Results. As discussed in section 2, we compared the performance of the bipartite RankSVM algorithm with that of SVM-based ranking on five virtual screening data sets consisting of actives with respect to CDK2, COX2, FXa, PDE5, and α_{1A}AR, and a background set of inactives. Two different molecular fingerprint representations were used for these data sets: MolPrint2D fingerprints and OpenBabel FP2 fingerprints. As described in section 2.4, using MolPrint2D fingerprints, each compound in these data sets was represented as a binary fingerprint vector with 17 221 bit positions; thus the instance space in this case was $X = \{0, 1\}^{17,221}$. Using FP2 fingerprints, each compound was represented as a vector with 1021 bit positions; thus in this case we had $X = \{0, 1\}^{1021}$. For both fingerprints, we used the Tanimoto kernel, which is widely used with binary fingerprint vectors and is defined as

$$K_{\mathrm{Tanimoto}}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x} \cdot \mathbf{x}'}{\mathbf{x} \cdot \mathbf{x} - \mathbf{x} \cdot \mathbf{x}' + \mathbf{x}' \cdot \mathbf{x}'} \qquad (15)$$

The SVM algorithm was implemented using the SVM$^{\mathrm{light}}$ software of Joachims.[46,47] The regularization parameter C in each training run was selected by 5-fold cross-validation from the range $\{0.1, 1, 10, 100, 1000\}$; in each case, the value of C that gave the lowest average bipartite ranking error (equivalently, the highest average AUC) across the five folds was used in training.

For the bipartite RankSVM algorithm, we used the gradient projection algorithm described in section 2.2, implemented in C++. The number of iterations T was fixed to 1000; the learning rate η and the regularization parameter C were selected by 5-fold cross-validation as above, with C being selected from the same range as above, and η being selected from the range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$.

As described in section 2.4, we considered three different splits of the data sets into training and test sets. We refer to the two original splits used by Jorissen and Gilson[12] as 1st/2nd (in which the first half of each list of actives, grouped by structure, was used for training and the second half for testing; the reverse split 2nd/1st was also considered) and Odd/Even (in which the odd-numbered compounds in each list of actives were used for training and the even-numbered compounds for testing; again the reverse split Even/Odd was also considered). The third group of splits consisted of randomly partitioning each data set and using one-half for training and the other half for testing.

The results on the 1st/2nd splits using MolPrint2D fingerprints are shown in Table 1; those using FP2 fingerprints are shown in Table 2. In each case, the better performance is shown in bold typeface. Recall from section 2.4 that these splits represent a worst-case situation, in which the training and test compounds are chosen to be structurally different. While there is no clear winner between the two

Table 1. Virtual Screening Results on 1st/2nd Splits Using MolPrint2D Fingerprints (with the Tanimoto Kernel)

		ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	train/test split	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	1st/2nd	0.2616	**0.2313**	0.7384	**0.7687**	**0.2841**	0.1573	**8**	6	**13**	**13**
	2nd/1st	**0.2734**	0.2797	**0.7266**	0.7203	**0.3428**	0.3357	**7**	**7**	**13**	12
COX2	1st/2nd	0.1688	**0.1534**	0.8312	**0.8466**	0.6914	**0.7246**	**18**	**18**	**19**	**19**
	2nd/1st	0.2672	**0.2207**	0.7328	**0.7723**	0.2891	**0.3299**	**6**	**6**	12	**15**
FXa	1st/2nd	**0.0026**	0.0031	**0.9974**	0.9969	**0.9305**	0.9162	**22**	**22**	**25**	**25**
	2nd/1st	0.1101	**0.0581**	0.8899	**0.9419**	0.6927	**0.6944**	17	**18**	**18**	**18**
PDE5	1st/2nd	**0.1467**	0.1660	**0.8533**	0.8340	**0.1226**	0.1053	**4**	3	**15**	14
	2nd/1st	0.2180	**0.1280**	0.7820	**0.8720**	0.0935	**0.1064**	1	**2**	**12**	9
α_{1A}AR	1st/2nd	0.0810	**0.0517**	0.9190	**0.9483**	0.6875	**0.7091**	**16**	**16**	20	**21**
	2nd/1st	0.0291	**0.0268**	0.9709	**0.9732**	0.6271	**0.6801**	14	**15**	20	**22**

Table 2. Virtual Screening Results on 1st/2nd Splits using FP2 Fingerprints (with the Tanimoto Kernel)

		ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	train/test split	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	1st/2nd	0.2047	**0.1031**	0.7953	**0.8969**	0.1902	**0.3123**	**8**	7	8	**18**
	2nd/1st	0.2244	**0.1929**	0.7756	**0.8071**	0.2167	**0.3148**	7	**8**	13	**14**
COX2	1st/2nd	0.1095	**0.0999**	0.8905	**0.9001**	0.7370	**0.7431**	**17**	**17**	20	**21**
	2nd/1st	0.1772	**0.1702**	0.8228	**0.8298**	**0.4316**	0.3730	**10**	7	**17**	**17**
FXa	1st/2nd	0.0281	**0.0204**	0.9719	**0.9796**	0.7784	**0.8019**	**18**	**18**	**23**	**23**
	2nd/1st	0.0273	**0.0176**	0.9727	**0.9824**	**0.7597**	0.7338	**17**	16	22	**23**
PDE5	1st/2nd	**0.0732**	0.0869	**0.9268**	0.9131	**0.1703**	0.1443	**3**	3	**16**	14
	2nd/1st	**0.0932**	0.1042	**0.9068**	0.8958	**0.2043**	0.2023	5	**6**	**17**	16
α_{1A}AR	1st/2nd	0.0439	**0.0368**	0.9561	**0.9633**	**0.6647**	0.6097	**15**	14	19	**20**
	2nd/1st	0.0409	**0.0280**	0.9591	**0.9720**	0.5320	**0.5920**	10	**12**	**21**	**21**

Table 3. Virtual Screening Results on Odd/Even Splits using MolPrint2D Fingerprints (with the Tanimoto Kernel)

		ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	train/test split	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	Odd/Even	0.0217	**0.0212**	0.9783	**0.9788**	**0.8936**	0.8700	**21**	18	**24**	**24**
	Even/Odd	0.0084	**0.0069**	0.9916	**0.9931**	0.9243	**0.9381**	**22**	**22**	**24**	**24**
COX2	Odd/Even	0.0030	**0.0021**	0.9970	**0.9979**	0.9430	**0.9575**	**22**	**22**	**25**	**25**
	Even/Odd	**0.0044**	0.0261	**0.9956**	0.9739	**0.8922**	0.8263	**22**	20	**25**	23
FXa	Odd/Even	**0.0010**	0.0033	**0.9990**	0.9967	**0.9623**	0.9212	**22**	21	**25**	**25**
	Even/Odd	0.0008	**0.0005**	0.9992	**0.9995**	0.9683	**0.9802**	**23**	**23**	**25**	**25**
PDE5	Odd/Even	0.0021	**0.0016**	0.9979	**0.9984**	0.9709	**0.9736**	**24**	**24**	**25**	**25**
	Even/Odd	0.0025	**0.0020**	0.9975	**0.9980**	0.9503	**0.9570**	**23**	**23**	**25**	**25**
α_{1A}AR	Odd/Even	**0.0072**	0.0083	**0.9928**	0.9917	0.8916	**0.9079**	20	**21**	**24**	**24**
	Even/Odd	0.0063	**0.0037**	0.9937	**0.9963**	0.8932	**0.9181**	20	**21**	24	**25**

Table 4. Virtual Screening Results on Odd/Even Splits Using FP2 Fingerprints (with the Tanimoto Kernel)

		ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	train/test split	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	Odd/Even	**0.0060**	0.0249	**0.9940**	0.9751	**0.8841**	0.7719	**19**	18	**25**	21
	Even/Odd	0.0225	**0.0150**	0.9775	**0.9850**	0.8307	**0.8714**	18	**20**	**24**	**24**
COX2	Odd/Even	0.0086	**0.0079**	0.9914	**0.9921**	0.9262	**0.9348**	**23**	**23**	23	**24**
	Even/Odd	0.0509	**0.0149**	0.9491	**0.9851**	0.8096	**0.8898**	20	**21**	20	**23**
FXa	Odd/Even	0.0176	**0.0005**	0.9824	**0.9995**	0.8899	**0.9830**	22	**23**	22	**25**
	Even/Odd	0.0078	**0.0020**	0.9922	**0.9980**	0.8949	**0.9534**	21	**22**	24	**25**
PDE5	Odd/Even	0.0049	**0.0041**	0.9951	**0.9959**	0.9415	**0.9507**	22	**24**	**24**	**24**
	Even/Odd	0.0127	**0.0091**	0.9873	**0.9909**	0.9394	**0.9638**	23	**24**	**24**	**24**
α_{1A}AR	Odd/Even	0.0102	**0.0077**	0.9898	**0.9923**	0.8597	**0.9105**	20	**22**	**24**	**24**
	Even/Odd	0.0381	**0.0288**	0.9619	**0.9712**	0.8325	**0.8561**	**19**	**19**	**23**	**23**

algorithms for these splits overall, we note that using the FP2 fingerprints in particular, the RankSVM algorithm tends to give better performance than the SVM algorithm in terms of the bipartite ranking error or AUC. We also note that overall, the FP2 fingerprint representation tends to result in superior ranking performance for these splits than the MolPrint2D fingerprint representation (as measured by the ranking error or AUC), independent of the particular algorithm used.

Table 3 shows results on the Odd/Even splits using MolPrint2D fingerprints; Table 4 shows results on these splits using FP2 fingerprints. Recall that these splits represent a best-case situation, in which the training compounds are chosen to be highly similar to the test compounds. Consequently, the performance on these splits is significantly better than the performance on the 1st/2nd splits above; this is true for both algorithms, regardless of the data set or representation considered. Again, while there is no clear winner between the two algorithms for these splits, we note that using FP2 fingerprints in particular, the RankSVM algorithm tends to give better ranking performance than the SVM algorithm, particularly in terms of the bipartite ranking error or AUC, and in this case, also in terms of the average precision.

Both the worst-case situation represented by the 1st/2nd splits and the best-case situation represented by the Odd/Even splits are extremes that make it difficult to compare algorithms in an objective manner: in the former situation, it is difficult for any learning algorithm to perform well; in the latter, almost any reasonable algorithm should give good results. Indeed, as seen above, the difference in results between the 1st/2nd and Odd/Even splits is much greater than the difference in results between the two algorithms or across different representations for any of these splits.

The third group of train/test splits, in which the data sets were randomly partitioned into training and test sets, were designed to simulate a situation in which the training set would be more representative of the test set than in the first (worst-case) situation above, but less similar to the test set than in the second (best-case) situation. The results on these random splits using MolPrint2D fingerprints are shown in Table 5; those using FP2 fingerprints are shown in Table 6. Each number shown in these tables is the average over 10 different random splits. In each case, for each of the five targets, the 50 actives and 2092 inactives (see section 2.4) were first partitioned randomly into training and test sets, each containing (as in the 1st/2nd and Odd/Even splits above) 25 actives and 1046 inactives; in addition, to evaluate the

Table 5. Virtual Screening Results on Random Splits Using MolPrint2D Fingerprints (with the Tanimoto Kernel)

	no. train	ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	(act−inact)	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	5−209	0.3088	**0.2575**	0.6912	**0.7425**	0.3302	**0.3350**	7.8	**7.9**	11.5	**12.1**
	10−418	0.1880	**0.1705**	0.8120	**0.8295**	**0.5392**	0.5110	**13.2**	12.6	**16.6**	16.3
	15−627	0.1504	**0.1262**	0.8496	**0.8738**	0.6168	**0.6504**	14.9	**16.1**	17.7	**18.4**
	20−836	0.1070	**0.0655**	0.8930	**0.9345**	0.7189	**0.7377**	17.3	**17.5**	20.0	**21.4**
	25−1046	0.0632	**0.0545**	0.9368	**0.9455**	**0.7751**	0.7575	**18.5**	17.8	21.6	**22.1**
COX2	5−209	0.2208	**0.1958**	0.7792	**0.8042**	0.6188	**0.6293**	15.1	**15.4**	**17.1**	**17.1**
	10−418	0.1315	**0.1159**	0.8685	**0.8841**	0.7079	**0.7321**	17.2	**17.6**	19.1	**19.9**
	15−627	0.0653	**0.0652**	0.9347	**0.9348**	**0.8032**	0.8016	**19.4**	19.2	**21.5**	21.4
	20−836	0.0709	**0.0376**	0.9291	**0.9624**	0.8251	**0.8481**	19.9	**20.0**	22.3	**23.0**
	25−1046	0.0306	**0.0198**	0.9694	**0.9802**	**0.8613**	0.8578	**20.4**	20.1	**23.6**	23.4
FXa	5−209	0.0751	**0.0604**	0.9249	**0.9396**	**0.7428**	0.7408	**19.0**	**19.0**	20.8	**21.0**
	10−418	0.0589	**0.0414**	0.9411	**0.9586**	**0.8189**	0.8146	20.1	**20.3**	22.0	**22.1**
	15−627	0.0280	**0.0223**	0.9720	**0.9777**	**0.8549**	0.8475	20.6	**20.7**	**23.1**	**23.1**
	20−836	**0.0164**	0.0177	**0.9836**	0.9823	**0.8706**	0.8589	**20.9**	20.7	**23.8**	23.5
	25−1046	0.0298	**0.0155**	0.9702	**0.9845**	**0.8693**	0.8666	**21.1**	21.0	23.7	**23.8**
PDE5	5−209	0.2362	**0.2108**	0.7638	**0.7892**	**0.4288**	0.4119	**10.7**	10.5	13.2	**13.7**
	10−418	0.1093	**0.0904**	0.8907	**0.9096**	0.7034	**0.7175**	**16.7**	**16.7**	**20.3**	20.0
	15−627	0.0462	**0.0383**	0.9538	**0.9617**	**0.8188**	0.8142	**19.4**	19.2	**22.2**	22.1
	20−836	0.0323	**0.0238**	0.9677	**0.9762**	0.8599	**0.8733**	20.2	**20.5**	**23.4**	**23.4**
	25−1046	0.0192	**0.0168**	0.9808	**0.9832**	0.8988	**0.9031**	**21.4**	21.3	23.5	**23.8**
α_{1A}AR	5−209	**0.1047**	0.1053	**0.8953**	0.8947	**0.6420**	0.6377	14.4	**14.5**	18.4	**18.6**
	10−418	**0.0564**	0.0587	**0.9436**	0.9413	**0.7106**	0.6998	**15.8**	15.7	**21.2**	21.1
	15−627	**0.0331**	0.0366	**0.9669**	0.9634	**0.7702**	0.7627	**17.2**	16.9	**22.4**	22.1
	20−836	0.0286	**0.0169**	0.9714	**0.9831**	0.7902	**0.8249**	18.1	**18.9**	22.3	**23.3**
	25−1046	0.0195	**0.0114**	0.9805	**0.9886**	0.8061	**0.8510**	18.5	**19.5**	23.3	**23.6**

Table 6. Virtual Screening Results on Random Splits using FP2 Fingerprints (with the Tanimoto Kernel)

	no. train	ranking error		AUC		average precision		actives in top 25		actives in top 100	
data set	(act−inact)	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM	SVM	RankSVM
CDK2	5−209	0.1993	**0.1262**	0.8007	**0.8738**	0.3564	**0.3849**	8.6	**8.8**	12.8	**13.4**
	10−418	**0.0896**	0.0897	**0.9104**	0.9103	**0.5714**	0.5486	**13.6**	12.9	**18.9**	17.9
	15−627	0.0654	**0.0630**	0.9346	**0.9370**	**0.6726**	0.6712	15.9	**16.3**	**20.2**	20.1
	20−836	0.0392	**0.0362**	0.9608	**0.9638**	**0.7314**	0.7263	**17.0**	16.9	**21.8**	21.3
	25−1046	0.0406	**0.0341**	0.9594	**0.9659**	**0.7539**	0.7493	17.3	**17.5**	**22.2**	21.7
COX2	5−209	0.1539	**0.1306**	0.8461	**0.8694**	**0.5007**	0.4851	**11.8**	11.4	16.8	**17.6**
	10−418	0.1102	**0.0991**	0.8898	**0.9009**	**0.6468**	0.6355	**15.6**	15.1	**19.1**	18.6
	15−627	0.0893	**0.0726**	0.9107	**0.9274**	0.7103	**0.7155**	16.6	**17.0**	19.9	**20.2**
	20−836	0.0690	**0.0449**	0.9310	**0.9551**	0.7486	**0.7926**	17.7	**18.0**	20.5	**22.0**
	25−1046	0.0666	**0.0366**	0.9334	**0.9634**	0.7708	**0.8188**	18.4	**18.8**	20.7	**21.9**
FXa	5−209	0.0653	**0.0550**	0.9347	**0.9450**	**0.6324**	0.6219	**14.1**	**14.1**	**20.7**	20.4
	10−418	0.0183	**0.0178**	0.9817	**0.9822**	**0.8288**	0.7976	**19.1**	18.0	23.3	**23.5**
	15−627	0.0117	**0.0105**	0.9883	**0.9895**	**0.8896**	0.8865	**20.4**	20.2	23.9	**24.2**
	20−836	**0.0121**	0.0122	**0.9879**	0.9878	**0.9133**	0.9097	21.2	**21.4**	**24.0**	**24.0**
	25−1046	0.0131	**0.0111**	0.9869	**0.9889**	0.9219	**0.9273**	21.5	**21.7**	**24.1**	**24.1**
PDE5	5−209	0.1334	**0.0978**	0.8666	**0.9022**	0.4080	**0.4511**	9.8	**11.0**	15.4	**17.2**
	10−418	0.0583	**0.0475**	0.9417	**0.9525**	**0.6796**	0.6744	15.2	**15.5**	20.3	**21.5**
	15−627	0.0336	**0.0326**	0.9664	**0.9674**	0.7850	**0.7938**	18.5	**18.8**	**22.5**	22.3
	20−836	0.0247	**0.0190**	0.9753	**0.9810**	0.8278	**0.8621**	19.4	**20.5**	22.8	**23.2**
	25−1046	0.0171	**0.0137**	0.9829	**0.9863**	0.8671	**0.9161**	20.0	**21.9**	23.8	**23.9**
α_{1A}AR	5−209	0.1149	**0.0882**	0.8851	**0.9118**	0.4516	**0.4638**	**11.0**	10.9	**16.9**	16.7
	10−418	0.0580	**0.0578**	0.9420	**0.9422**	**0.6183**	0.5593	**14.3**	13.3	**19.8**	19.5
	15−627	0.0427	**0.0372**	0.9573	**0.9628**	0.6835	**0.6905**	**15.8**	15.5	20.9	**21.1**
	20−836	0.0289	**0.0251**	0.9711	**0.9749**	0.7405	**0.7468**	16.3	**16.8**	22.3	**22.8**
	25−1046	0.0257	**0.0186**	0.9743	**0.9814**	0.7599	**0.8079**	16.6	**18.1**	22.5	**23.5**

impact of the training set size, experiments were then conducted in which increasing fractions of the training set (20%, amounting to 5 actives and 209 inactives; 40%, amounting to 10 actives and 418 inactives; and so on, up to 100%, amounting to 25 actives and 1046 inactives) were actually used to train the algorithms.

There are several observations to be made. First, consistent with observations made by Geppert et al.[13] in the context of SVM-based ranking, the performance on the test set generally improves with an increase in the training set size. This is true for both SVM and RankSVM algorithms, regardless of the data set or representation used.

Second, when the complete training set consisting of 25 actives and 1046 inactives is used (same training set size as in the 1st/2nd and Odd/Even splits above), we find that in agreement with our expectations, the performance in the case of these random splits lies between the two extreme performance levels observed in the case of the 1st/2nd and Odd/Even splits; again, this is true for both algorithms, regardless of the particular data set/representation used. This

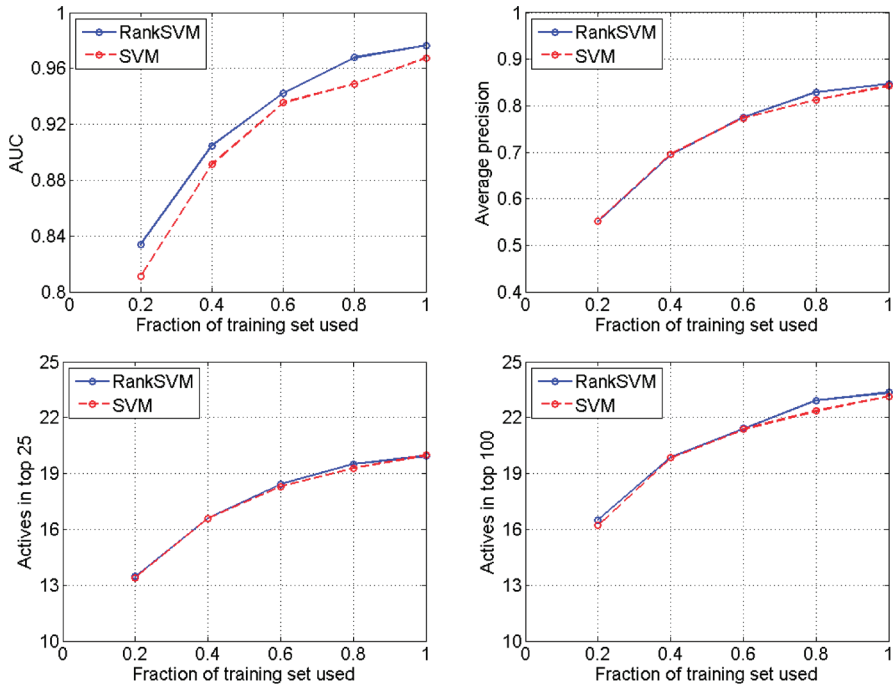

Figure 7. Virtual screening results on random splits using MolPrint2D fingerprints (with the Tanimoto kernel), averaged over the five data sets.

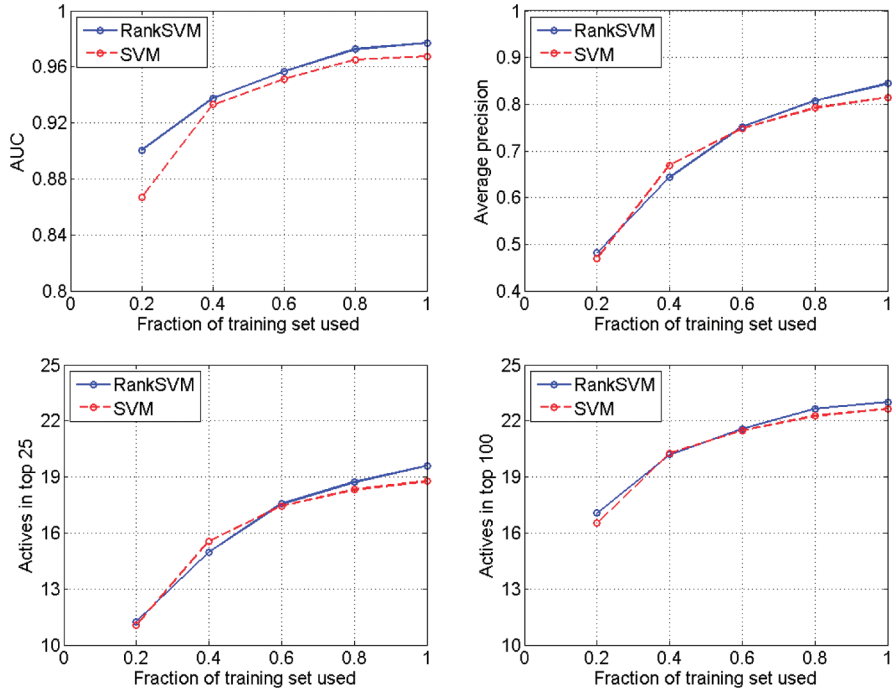

Figure 8. Virtual screening results on random splits using FP2 fingerprints (with the Tanimoto kernel), averaged over the five data sets.

confirms that the training sets in the random splits are on average more representative of the corresponding test sets than in the 1st/2nd splits and less similar to the test sets than in the Odd/Even splits.

Third, as observed previously in the case of the 1st/2nd splits, we note that the FP2 fingerprints tend to result in somewhat better ranking performance than the MolPrint2D fingerprints when measured in terms the bipartite ranking error or AUC, particularly for smaller training set sizes.

Finally, we observe that on these random train/test splits, the ranking performance of the RankSVM algorithm in terms of the bipartite ranking error or AUC is in general superior to SVM-based ranking. This is true for all five data sets and for both fingerprint representations, and is also illustrated in Figures 7 and 8, which show the performance of the two algorithms using the MolPrint2D and FP2 fingerprints, respectively, averaged over the five data sets. We note that these figures also show that when measured in terms of the other performance measures, such as average precision or number of actives retrieved in the top portion of the ranking, the performance of the two algorithms is broadly similar. This is not surprising given that the RankSVM algorithm optimizes (approximately) the bipartite ranking error or AUC, which are global measures of ranking performance, whereas

Table 7. QSAR Ranking Results on Original Splits using 2.5D Descriptors (with the Gaussian Kernel)

		ranking error		correlation		Kendall's τ		Spearman's ρ		NDCG	
data set	train/test split	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM
DHFR	original	0.1837	**0.1726**	0.7519	**0.7618**	0.5571	**0.5747**	0.7552	**0.7758**	0.8540	**0.8632**
COX2	original	**0.3138**	0.3173	**0.5836**	0.5703	**0.4351**	0.4346	0.6100	**0.6174**	**0.9399**	0.9231

Table 8. QSAR Ranking Results on Original Splits using FP2 Fingerprints (with the Tanimoto Kernel)

		ranking error		correlation		Kendall's τ		Spearman's ρ		NDCG	
data set	train/test split	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM
DHFR	original	0.1873	**0.1629**	0.7486	**0.7838**	0.5760	**0.5892**	0.7752	**0.8015**	0.8426	**0.8498**
COX2	original	0.3335	**0.3129**	0.5658	**0.5897**	0.4374	**0.4442**	0.6013	**0.6199**	0.9314	**0.9351**

the other three measures focus on local ranking accuracy at the top of a ranking (see section 2.5); we discuss this further in section 4.3 below.

4.2. QSAR Ranking Results. As discussed in Section 3, we compared the performance of the RankSVM algorithm for ranking with real-valued labels with that of SVR-based ranking on two QSAR data sets consisting of inhibitors of DHFR and COX2, respectively. Two different descriptor representations were used for these data sets: 2.5D descriptors, which were used in the original study by Sutherland et al.,[10] and OpenBabel FP2 fingerprints, which were also used in the virtual screening experiments described above. As described in section 3.4, the 2.5D representation for the DHFR data set contained 70 real-valued descriptors; that for the COX2 data set contained 74 real-valued descriptors. All descriptors were scaled to lie between 0 and 1, thus yielding an instance space of the form $X = [0, 1]^d$, with $d = 74$ for the DHFR data set and $d = 70$ for the COX2 data set. For these descriptors, we used the Gaussian kernel, also known as the radial basis function (RBF) kernel, which is widely used with real-valued descriptor vectors and is defined as

$$K_{\text{Gauss},\gamma}(\mathbf{x}, \mathbf{x}') = e^{-\gamma||\mathbf{x}-\mathbf{x}'||^2} \tag{16}$$

where $\gamma > 0$ is a parameter for this kernel. As discussed previously, the FP2 fingerprint representation yields an instance space of the form $X = \{0, 1\}^{1021}$, where each compound is represented as a 1021-dimensional bit vector; in this case, we used the Tanimoto kernel (see eq 15), which as described in section 4.1, is widely used with binary fingerprint vectors.

The SVR algorithm was implemented using the SVM$^{\text{light}}$ software of Joachims.[46,47] The regularization parameter C and the sensitivity parameter ϵ in each training run were selected by 5-fold cross-validation from the ranges {0.1, 1, 10, 100, 1000, 10000} and {0.01, 0.05, 0.1, 0.5, 1}, respectively. In addition, when using the Gaussian kernel, the parameter γ was similarly selected from the range {1/16, 1/4, 1, 4, 16}. In each case, the combination of parameters that gave the lowest ranking error (see Section 3) across the five folds was used in training.

For the RankSVM algorithm for real-valued labels, we used the gradient projection algorithm described in section 3.2, implemented in C++. The number of iterations T was fixed to 1000; the learning rate η, the regularization parameter C, and when using the Gaussian kernel, the kernel parameter γ, were all selected in each training run by 5-fold cross-validation as above. The parameters C and γ were chosen from the same ranges as above; the parameter η was selected from the range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$.

As described in section 3.4, we considered two different splits of the data sets into training and test sets: the original split used by Sutherland et al.,[10] in which roughly one-third of each data set was chosen to form the test set using a maximum dissimilarity algorithm, and a group of random splits in which each data set was randomly partitioned into training and test sets of the same sizes as in the original split.

The results on the original splits using the 2.5D descriptors are shown in Table 7; those using FP2 fingerprints are shown in Table 8. Again, in each case, the better performance is shown in bold typeface. While the performance of the two algorithms on these splits is similar overall, we note that using the FP2 fingerprints in particular, the RankSVM algorithm gives slightly better ranking performance than the SVR algorithm.

As discussed in section 3.4, the original splits above represent a worst-case situation, since the test compounds are selected to be maximally diverse and are therefore likely to require considerable extrapolation from the training set. The random splits were designed to simulate a situation in which the training set would be more representative of the test set. The results on these random splits using the 2.5D descriptors are shown in Table 9; those using FP2 fingerprints are shown in Table 10. Each number shown in these tables is the average over 10 different random splits. In each case, 237 out of 361 compounds in the DHFR data set and 188 of 292 compounds in the COX2 data set were randomly selected to form the training set; the remaining compounds were assigned to the test set (same sizes as in the original splits as above). In addition, to evaluate the impact of the training set size, experiments were conducted in which increasing fractions of the training set (10%, 20%, and so on up to 100%) were actually used to train the algorithms.

Again, there are several observations to be made. A first observation is that as with the virtual screening experiments described in section 4.1, and consistent with the observations of Geppert et al.[13] in the context of SVM-based ranking, an increase in training set size generally leads to improvement in performance on the test set. This is observed most consistently for performance measured in terms of the ranking error, but is broadly true also for the other performance measures.

A second observation is that the FP2 fingerprints (with the Tanimoto kernel) tend to give better ranking performance

Table 9. QSAR Ranking Results on Random Splits Using 2.5D Descriptors (with the Gaussian Kernel)

		ranking error		correlation		Kendall's τ		Spearman's ρ		NDCG	
data set	no. train	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM
DHFR	24	0.4755	**0.4601**	**0.0982**	0.0966	**0.0727**	0.0627	**0.1082**	0.0935	**0.7055**	0.7004
	48	**0.3430**	0.3509	**0.1857**	0.1747	**0.1225**	0.1156	**0.1813**	0.1723	**0.7084**	0.7022
	72	0.2840	**0.2726**	**0.2355**	0.2231	**0.1552**	0.1513	**0.2274**	0.2204	0.7123	**0.7129**
	96	0.2483	**0.2351**	0.2489	**0.2548**	**0.1715**	0.1699	**0.2506**	0.2482	**0.7116**	0.7086
	120	0.2171	**0.2121**	0.2519	**0.2560**	0.1692	**0.1714**	0.2466	**0.2518**	**0.7107**	0.7105
	144	**0.2023**	0.2032	0.2409	**0.2511**	**0.1667**	0.1640	**0.2439**	0.2399	**0.7085**	0.7079
	168	0.2019	**0.1817**	0.2480	**0.2774**	0.1657	**0.1867**	0.2407	**0.2722**	0.7109	**0.7156**
	192	0.1808	**0.1749**	0.2706	**0.2775**	0.1830	**0.1851**	0.2683	**0.2717**	**0.7146**	0.7103
	216	0.1816	**0.1722**	0.2596	**0.2686**	0.1722	**0.1806**	0.2543	**0.2660**	**0.7223**	0.7114
	237	0.1714	**0.1681**	0.2606	**0.2712**	0.1754	**0.1803**	0.2575	**0.2667**	**0.7089**	0.7085
COX2	19	0.4362	**0.4294**	**0.1194**	0.1120	0.0937	**0.1040**	0.1423	**0.1567**	**0.7845**	0.7786
	38	**0.3777**	0.3922	**0.1605**	0.1576	0.1257	**0.1300**	0.1813	**0.1881**	0.7882	**0.7963**
	57	0.3325	**0.3237**	**0.1745**	0.1647	**0.1478**	0.1362	**0.2146**	0.1933	**0.7924**	0.7897
	76	0.3046	**0.3041**	**0.1722**	0.1708	**0.1444**	0.1433	**0.2080**	0.2070	0.7880	**0.7974**
	95	0.2667	**0.2629**	**0.1960**	0.1837	**0.1615**	0.1509	**0.2313**	0.2164	0.7853	**0.7901**
	114	0.2633	**0.2616**	**0.1840**	0.1839	0.1455	**0.1505**	0.2096	**0.2144**	**0.7906**	0.7899
	133	0.2760	**0.2688**	0.1957	**0.2034**	0.1560	**0.1596**	0.2247	**0.2291**	0.7885	**0.7977**
	152	**0.2520**	**0.2520**	0.2011	**0.2022**	**0.1626**	0.1613	**0.2340**	0.2318	0.7925	**0.8018**
	171	**0.2392**	0.2397	0.1988	**0.2125**	0.1650	**0.1714**	0.2356	**0.2454**	0.8001	**0.8101**
	188	**0.2153**	0.2250	**0.1912**	0.1899	0.1553	**0.1589**	0.2198	**0.2241**	0.7958	**0.8006**

Table 10. QSAR Ranking Results on Random Splits Using FP2 Fingerprints (with the Tanimoto Kernel)

		ranking error		correlation		Kendall's τ		Spearman's ρ		NDCG	
data set	no. train	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM	SVR	RankSVM
DHFR	24	0.3793	**0.3546**	**0.2658**	0.2577	**0.1786**	0.1694	**0.2634**	0.2485	0.7091	**0.7194**
	48	0.2905	**0.2896**	**0.2959**	0.2950	0.1970	**0.1979**	**0.2900**	**0.2900**	0.7090	**0.7234**
	72	0.2517	**0.2421**	0.3169	**0.3222**	**0.2118**	0.2116	**0.3097**	0.3077	**0.7148**	0.7144
	96	0.2343	**0.2201**	0.3021	**0.3236**	0.2019	**0.2150**	0.2958	**0.3124**	0.7127	**0.7140**
	120	0.2147	**0.2052**	0.3260	**0.3321**	0.2219	**0.2250**	0.3255	**0.3269**	**0.7238**	0.7196
	144	0.2166	**0.1988**	0.3054	**0.3400**	0.2018	**0.2219**	0.2974	**0.3237**	0.7131	**0.7225**
	168	0.2096	**0.1966**	0.3146	**0.3413**	0.2115	**0.2278**	0.3074	**0.3306**	0.7265	**0.7321**
	192	0.2056	**0.1962**	0.3207	**0.3528**	0.2147	**0.2357**	0.3126	**0.3421**	0.7230	**0.7359**
	216	0.1907	**0.1787**	0.3336	**0.3558**	0.2214	**0.2378**	0.3258	**0.3453**	0.7202	**0.7216**
	237	0.1924	**0.1798**	0.3357	**0.3578**	0.2275	**0.2419**	0.3321	**0.3516**	**0.7316**	0.7306
COX2	19	**0.3494**	0.3597	**0.1621**	0.1317	**0.1467**	0.1305	**0.2190**	0.1936	**0.7840**	0.7824
	38	0.3478	**0.3400**	0.1703	**0.1855**	0.1376	**0.1468**	0.2054	**0.2172**	**0.7835**	0.7808
	57	**0.3123**	0.3171	0.1519	**0.1771**	0.1345	**0.1499**	0.2000	**0.2234**	0.7721	**0.7853**
	76	0.3096	**0.3050**	0.1681	**0.1738**	**0.1442**	0.1429	**0.2103**	0.2075	0.7809	**0.7875**
	95	**0.2840**	0.2884	**0.2178**	0.1933	**0.1741**	0.1550	**0.2549**	0.2280	**0.7949**	0.7908
	114	0.2787	**0.2608**	0.1839	**0.1992**	0.1510	**0.1644**	0.2189	**0.2349**	0.7952	**0.8027**
	133	0.2653	**0.2479**	0.1868	**0.1973**	0.1523	**0.1602**	0.2219	**0.2318**	0.7885	**0.7953**
	152	0.2465	**0.2388**	**0.2050**	0.2039	**0.1659**	0.1630	**0.2404**	0.2381	**0.7944**	**0.7944**
	171	0.2394	**0.2317**	**0.2078**	0.2062	0.1661	**0.1683**	0.2418	**0.2463**	0.7958	**0.7978**
	188	0.2300	**0.2136**	**0.1924**	0.1908	**0.1591**	0.1579	**0.2314**	0.2293	0.7948	**0.7950**

than the 2.5D descriptors (with the Gaussian kernel) for smaller training set sizes. On the other hand, the 2.5D descriptors appear to give better performance for larger training set sizes.

Finally, we observe that on these random train/test splits, the ranking performance of the RankSVM algorithm in terms of the ranking error is in general superior to SVR-based ranking. This is true for both data sets and for both descriptor representations. We also note that in terms of the other performance measures such as the rank correlation coefficients or NDCG, neither algorithm is a clear winner. This is not surprising given that the RankSVM algorithm optimizes the (relevance-weighted) ranking error, which takes into account the actual activity values of the compounds, whereas the Kendall τ and Spearman ρ rank correlation coefficients measure performance only with respect to the ordering or ranking of compounds implied by the activities. Morevoer, the ranking error measures the global performance of a ranking, whereas the NDCG focuses on local accuracy at the top of a ranking (see section 3.5). This is discussed further below.

4.3. Discussion. We observed in the virtual screening experiments above that the bipartite RankSVM algorithm outperformed SVM-based ranking in terms of the bipartite ranking error or AUC, which are the performance measures (approximately) optimized by the RankSVM algorithm in this setting. However, in terms of the other performance measures such as the average precision or the number of actives retrieved in the top portion of the ranking, the performance of the two algorithms was broadly similar. Similarly, in the QSAR ranking experiments, the RankSVM algorithm outperformed SVR-based ranking in terms of the (relevance-weighted) ranking error, which is the performance measure (approximately) optimized by the RankSVM algorithm in the real-valued labels setting. However, again, in terms of the other performance measures such as the Kendall

τ and Spearman ρ rank correlation coefficients or the NDCG, neither algorithm was a clear winner. This suggests the need for alternative ranking algorithms that could be used to optimize these other performance measures.

Indeed, as discussed in sections 2 and 3, performance measures such as the average precision and NDCG which focus on ranking accuracy at the top of a ranked list are of significant interest in IR applications, and there has been much effort recently in the machine learning and IR communities to develop algorithms that can optimize these criteria.[23,48−53] However, since these measures are defined in terms of the ranking of a whole list of objects (rather than pairs of objects), these algorithms generally require many different ranked lists, of documents ranked with respect to different queries, as training examples, together with a joint query-document feature representation to facilitate generalization across different queries. While we are not aware of any applications of these algorithms to drug discovery settings, this could potentially be a fruitful avenue to explore in the future. Using these algorithms in a drug discovery setting would require many different ranked lists, of compounds ranked with respect to different targets, as training examples, together with a joint target-compound representation. Such a representation was used implicitly in a different context recently by Geppert et al.,[37] through the use of joint target-ligand kernels.

Another approach is to develop machine learning algorithms that focus on ranking accuracy at the top of the list, but that require only preference examples between compounds with respect to a single target. One such example is the algorithm of Rudin,[26] which is a boosting style algorithm. Recently, we have developed a support vector style kernel-based ranking algorithm that focuses on accuracy at the top, and that has shown promising results in initial virtual screening experiments; this work will be reported elsewhere.

Another criterion that is often important in drug discovery settings is the diversity of the compounds returned at the top of a ranking. Ranking algorithms that can incorporate diversity constraints into the optimization criteria may therefore be of interest in such settings.[54]

Finally, in our experiments, we have used the Tanimoto kernel for binary fingerprint vectors and the Gaussian kernel for real-valued descriptor vectors, both of which are popular in the application of kernel-based methods in drug discovery.[9,12,37] However one can also use other kernel functions, such as molecular graph kernels and pharmacophore kernels that have been designed specifically for chemical applications,[55,56] in conjunction with the ranking methods we have proposed here. In particular, in cases where the library of chemical compounds to be ranked is known in advance, one can construct a chemical similarity graph over the compounds in the training set and test library using an arbitrary chemical similarity measure,[57,58] and then use graph-based ranking methods[21] that effectively construct a kernel function from such a graph.

5. CONCLUSION

The problem of ranking chemical structures arises in a variety of drug discovery settings. For example, in virtual screening applications, where the goal is to identify active compounds from large databases of mostly inactive compounds, one often wants to rank compounds such that active compounds are ranked higher than inactive ones. Similarly, in QSAR ranking applications, where the goal is to identify compounds with higher activities, one wants to rank compounds such that compounds with greater activities are ranked higher than those with lower activities. With the growing scale of chemical databases, machine learning and data mining methods are increasingly used for such tasks; in particular, classification methods, such as those based on SVMs, are widely used for ranking tasks in virtual screening applications, while regression methods, such as SVR, are frequently used for ranking tasks in QSAR applications. In this study, we have proposed a new machine learning approach for such tasks that makes use of recent advances in ranking methods in machine learning. Our experiments with a variety of chemical data sets and descriptors demonstrate that these new ranking methods, which are designed to directly optimize ranking accuracy, lead to better ranking performance than previous machine learning approaches; in particular, we show that variants of a support vector ranking algorithm called RankSVM outperform both SVM-based methods for virtual screening and SVR-based methods for QSAR ranking. This suggests that ranking methods in machine learning could prove to be a powerful tool for prioritizing drug candidates, with the potential to help lower the costs associated with failed molecules. As discussed above, there are many exciting avenues for further research, including for example the application and development of ranking algorithms that focus on ranking accuracy at the top of a ranking or that incorporate diversity constraints into the ranking, as well as the use of graph-based ranking methods in conjunction with appropriate chemical similarity graphs.

ACKNOWLEDGMENT

We would like to thank Michael Collins for discussions related to gradient projection methods for optimization problems encountered in machine learning. This work was supported in part by National Science Foundation Grant DMS-0732334 (to S.A.) and by a DoD Era of Hope Scholar Award and a Mary Kay Ash Charitable Foundation Grant (to S.S.). Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF, DoD or the Mary Kay Ash Foundation.

REFERENCES AND NOTES

(1) Shekhar, C. In Silico Pharmacology: Computer-Aided Methods Could Transform Drug Development. *Chem. Biol.* **2008**, *15*, 413–414.
(2) Jorgensen, W. L. The Many Roles of Computation in Drug Discovery. *Science* **2004**, *303*, 1813–1818.
(3) Bajorath, J. *Chemoinformatics: Concepts, Methods, and Tools for Drug Discovery*; Humana Press: Totowa, NJ, 2004.
(4) Wold, S.; Sjöström, M.; Eriksson, L. PLS-Regression: A Basic Tool of Chemometrics. *Chemometr. Intell. Lab.* **2001**, *58*, 109–130.
(5) King, R. D.; Hirst, J. D.; Sternberg, M. J. E. New Approaches to QSAR: Neural Networks and Machine Learning. *Perspect. Drug Discovery* **1993**, *1*, 279–290.
(6) Peterson, K. L. Artificial Neural Networks and Their Use in Chemistry. In *Reviews in Computational Chemistry*; Lipkowitz, K. B., Boyd, D. B., Eds.; Wiley-VCH: New York, NY, 2000; Vol. 16, pp 53−140.
(7) Rogers, D.; Hopfinger, A. J. Application of Genetic Function Approximation to Quantitative Structure−Activity Relationships and Quantitative Structure−Property Relationships. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 854–866.
(8) Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J. C.; Sheridan, R. P.; Feuston, B. P. Random Forest: A Classification and Regression Tool

for Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 19471958.
(9) Lind, P.; Maltseva, T. Support Vector Machines for the Estimation of Aqueous Solubility. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1855–1859.
(10) Sutherland, J. J.; O'Brien, L. A.; Weaver, D. F. A Comparison of Methods for Modeling Quantitative Structure−Activity Relationships. *J. Med. Chem.* **2004**, *47*, 5541–5554.
(11) Warmuth, M. K.; Liao, J.; Rätsch, G.; Mathieson, M.; Putta, S.; Lemmen, C. Active Learning with Support Vector Machines in the Drug Discovery Process. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 667–673.
(12) Jorissen, R. N.; Gilson, M. K. Virtual Screening of Molecular Databases Using a Support Vector Machine. *J. Chem. Inf. Model.* **2005**, *45*, 549–561.
(13) Geppert, H.; Horvath, T.; Gärtner, T.; Wrobel, S.; Bajorath, J. Support Vector Machine-Based Ranking Significantly Improves the Effectiveness of Similarity Searching Using 2D Fingerprints and Multiple Reference Compounds. *J. Chem. Inf. Model.* **2008**, *48*, 742–746.
(14) Cohen, W. W.; Schapire, R. E.; Singer, Y. Learning to Order Things. *J. Artif. Intell. Res.* **1999**, *10*, 243–270.
(15) Freund, Y.; Iyer, R.; Schapire, R. E.; Singer, Y. An Efficient Boosting Algorithm for Combining Preferences. *J. Mach. Learn. Res.* **2003**, *4*, 933–969.
(16) Herbrich, R.; Graepel, T.; Obermayer, K. Large Margin Rank Boundaries for Ordinal Regression. In *Advances in Large Margin Classifiers*; Bartlett, P., Schölkopf, B., Schuurmans, D., Smola, A. J., Eds.; MIT Press: Cambridge, MA, 2000; pp 115−132.
(17) Joachims, T. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, 2002; ACM: New York, NY, 2002; pp 133−142.
(18) Crammer, K.; Singer, Y. Online Ranking by Projecting. *Neural Comput.* **2005**, *17*, 145–175.
(19) Agarwal, S.; Graepel, T.; Herbrich, R.; Har-Peled, S.; Roth, D. Generalization Bounds for the Area Under the ROC Curve. *J. Mach. Learn. Res.* **2005**, *6*, 393–425.
(20) Burges, C. J. C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; Hullender, G. Learning to Rank using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005; Raedt, L. D., Wrobel, S., Eds.; ACM: New York, NY, 2005; pp 89−96.
(21) Agarwal, S. Ranking on Graph Data. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006; Cohen, W. W., Moore, A., Eds.; ACM: New York, NY, 2006; pp 25−32.
(22) Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; Li, H. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning*, Corvalis, OR, 2007; Ghahramani, Z., Ed.; ACM: New York, NY, 2007; pp 129−136.
(23) Cossock, D.; Zhang, T. Statistical Analysis of Bayes Optimal Subset Ranking. *IEEE Trans. Inform. Theory* **2008**, *54*, 5140–5154.
(24) Clemencon, S.; Lugosi, G.; Vayatis, N. Ranking and Empirical Minimization of U-Statistics. *Ann. Stat.* **2008**, *36*, 844–874.
(25) Rudin, C.; Schapire, R. E. Margin-Based Ranking and an Equivalence between AdaBoost and Rank-Boost. *J. Mach. Learn. Res.* **2009**, *10*, 2193–2232.
(26) Rudin, C. The P-Norm Push: A Simple Convex Ranking Algorithm that Concentrates at the Top of the List. *J. Mach. Learn. Res.* **2009**, *10*, 2233–2271.
(27) Agarwal, S.; Sengupta, S. Ranking Genes by Relevance to a Disease. In *Proceedings of the 8th Annual International Conference on Computational Systems Bioinformatics*, Stanford, CA, 2009; 2009.
(28) Cortes, C.; Mohri, M. AUC Optimization vs Error Rate Minimization. In *Advances in Neural Information Processing Systems 16*; Thrun, S., Saul, L., Schölkopf, B., Eds.; MIT Press: Cambridge, MA, 2004.
(29) Rakotomamonjy, A. Optimizing Area Under ROC Curves with SVMs. In *Proceedings of the ECAI-2004 Workshop on ROC Analysis in AI*, Valencia, Spain, 2004; 2004.
(30) Agarwal, S.; Niyogi, P. Generalization Bounds for Ranking Algorithms via Algorithmic Stability. *J. Mach. Learn. Res.* **2009**, *10*, 441–474.
(31) Wassermann, A. M.; Geppert, H.; Bajorath, J. Searching for Target-Selective Compounds Using Different Combinations of Multi-Class Support Vector Machine Ranking Methods, Kernel Functions, and Fingerprint Descriptors. *J. Chem. Inf. Model.* **2009**, *49*, 582–592.
(32) Aronszajn, N. Theory of Reproducing Kernels. *Trans. Am. Math. Soc.* **1950**, *68*, 337–404.
(33) Cucker, F.; Smale, S. On the Mathematical Foundations of Learning. *Bull. Am. Math. Soc.* **2002**, *39*, 1–49.
(34) Bertsekas, D. *Nonlinear Programming*, 2nd ed.; Athena Scientific: Nashua, NH, 1999.
(35) Burges, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discovery* **1998**, *2*, 121–167.
(36) Wilton, D.; Willett, P. Comparison of Ranking Methods for Virtual Screening in Lead-Discovery Programs. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 469–474.
(37) Geppert, H.; Humrich, J.; Stumpfe, D.; Gärtner, T.; Bajorath, J. Ligand Prediction from Protein Sequence and Small Molecule Information Using Support Vector Machines and Fingerprint Descriptors. *J. Chem. Inf. Model.* **2009**, *49*, 767–779.
(38) Bender, A.; Mussa, H. Y.; Glen, R. C.; Reiling, S. Molecular Similarity Searching Using Atom Environments, Information-Based Feature Selection, and a Naïve Bayesian Classifier. *J. Chem. Inf. Comp. Sci.* **2004**, *44*, 170–178.
(39) MOLPRINT 2D. http://www.molprint.com (accessed July 1, 2009).
(40) OpenBabel. http://openbabel.org (accessed July 2, 2009)).
(41) Hanley, J. A.; McNeil, B. J. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology* **1982**, *143*, 29–36.
(42) Smola, A. J.; Schölkopf, B. A Tutorial on Support Vector Regression. *Stat. Comput.* **2004**, *14*, 199–222.
(43) Kendall, M. A New Measure of Rank Correlation. *Biometrika* **1938**, *30*, 81–89.
(44) Spearman, C. The Proof and Measurement of Association Between two Things. *Am. J. Psychol.* **1904**, *15*, 72–101.
(45) Järvelin, K.; Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inform. Syst.* **2002**, *20*, 422–446.
(46) Joachims, T. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods—Support Vector Learning*; Schölkopf, B., Burges, C. J. C., Smola, A. J., Eds.; MIT Press: Cambridge, MA, 1999; Chapter 11, pp 169−184.
(47) SVMlight. http://svmlight.joachims.org (accessed May 20, 2009).
(48) Burges, C. J. C.; Ragno, R.; Le, Q. V. Learning to Rank with Non-Smooth Cost Functions. In *Advances in Neural Information Processing Systems 19*; Schölkopf, B., Platt, J., Hoffman, T., Eds.; MIT Press: Cambridge, MA, 2007; pp 193−200.
(49) Yue, Y.; Finley, T.; Radlinski, F.; Joachims, T. A Support Vector Method for Optimizing Average Precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, 2007; Kraaij, W., de Vries, A. P., Clarke, C. L. A., Fuhr, N., Kando, N., Eds.; ACM: New York, NY, 2007; pp 271−278.
(50) Xu, J.; Li, H. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, 2007; Kraaij, W., de Vries, A. P., Clarke, C. L. A., Fuhr, N., Kando, N., Eds.; ACM: New York, NY, 2007; pp 391−398.
(51) Chapelle, O.; Le, Q.; Smola, A. Large Margin Optimization of Ranking Measures. In *Proceedings of the NIPS-2007 Workshop on Machine Learning for Web Search*, Whistler, Canada, 2007; 2007.
(52) Taylor, M.; Guiver, J.; Robertson, S.; Minka, T. SoftRank: Optimizing Non-Smooth Rank Metrics. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining*, Palo Alto, CA; Najork, M., Broder, A. Z., Chakrabarti, S., Eds.; ACM: New York, NY, 2008; pp 77−86.
(53) Chakrabarti, S.; Khanna, R.; Sawant, U.; Bhattacharyya, C. Structured Learning for Non-Smooth Ranking Losses. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, 2008; Li, Y., Liu, B., Sarawagi, S., Eds.; ACM: New York, NY, 2008; pp 88−96.
(54) Radlinski, F.; Kleinberg, R.; Joachims, T. Learning Diverse Rankings with Multi-Armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008; Cohen, W. W., McCallum, A., Roweis, S. T., Eds.; ACM: New York, NY, 2008; pp 784−791.
(55) Mahé, P.; Ueda, N.; Akutsu, T.; Perret, J.-L.; Vert, J.-P. Graph Kernels for Molecular Structure-Activity Relationship Analysis with Support Vector Machines. *J. Chem. Inf. Model.* **2005**, *45*, 939–951.
(56) Mahé, P.; Ralaivola, L.; Stoven, V.; Vert, J.-P. The Pharmacophore Kernel for Virtual Screening with Support Vector Machines. *J. Chem. Inf. Model.* **2006**, *46*, 2003–2014.
(57) Willett, P. Chemical Similarity Searching. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 983–996.
(58) Nikolova, N.; Jaworska, J. Approaches to Measure Chemical Similarity—A Review. *QSAR Comb. Sci.* **2003**, *22*, 1006–1026.

CI9003865