Classifying Large Chemical Data Sets: Using A Regularized Potential Function Method

Hamse Y. Mussa,*,† Lezan Hawizy,† Florian Nigsch,‡ and Robert C. Glen†

Unilever Centre for Molecular Sciences Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom and CPC, LFP, Lead Discovery Informatics, Novartis Institutes for BioMedical Research, Inc., 250 Massachusetts Avenue Cambridge, Massachusetts 02139, United States

Received January 21, 2010

In recent years classifiers generated with kernel-based methods, such as support vector machines (SVM), Gaussian processes (GP), regularization networks (RN), and binary kernel discrimination (BKD) have been very popular in chemoinformatics data analysis. Aizerman et al. were the first to introduce the notion of employing kernel-based classifiers in the area of pattern recognition. Their original scheme, which they termed the potential function method (PFM), can basically be viewed as a kernel-based perceptron procedure and arguably subsumes the modern kernel-based algorithms. PFM can be computationally much cheaper than modern kernel-based classifiers; furthermore, PFM is far simpler conceptually and easier to implement than the SVM, GP, and RN algorithms. Unfortunately, unlike, e.g., SVM, GP, and RN, PFM is not endowed with both theoretical guarantees and practical strategies to safeguard it against generating overfitting classifiers. This is, in our opinion, the reason why this simple and elegant method has not been taken up in chemoinformatics. In this paper we empirically address this drawback: while maintaining its simplicity, we demonstrate that PFM combined with a simple regularization scheme may yield binary classifiers that can be, in practice, as efficient as classifiers obtained by employing state-of-the-art kernel-based methods. Using a realistic classification example, the augmented PFM was used to generate binary classifiers. Using a large chemical data set, the generalization ability of PFM classifiers were then compared with the prediction power of Laplacian-modified naive Bayesian (LmNB), Winnow (WN), and SVM classifiers.

INTRODUCTION

In drug discovery, virtual screening is often essential. Finding the required molecules by trial and error is not only time-consuming but also an expensive endeavor. A practical alternative that is cheap and fast is to build observational models (virtual screening) based on any available (and relevant) data about both the ligand and its interaction with the macromolecular target. An efficient computational approach for achieving this purpose is by employing machine learning (chemoinformatic techniques). Broadly speaking, virtual screening can often be reduced to two main schemes: phenomenological approaches, such as docking, and methods based on data analysis, such as similarity search approaches.[1−4] In the docking approach, one tries to find a ligand that complements the proposed binding site and hence induces (blocks) desirable (undesirable) activities of the target receptor efficiently. Similarity search techniques are based on the assumption that similar molecules have similar biological activity, i.e., molecules with similar physical and chemical properties are considered similar.[3,4]

Let $\mathbf{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ be a finite data sample that encapsulates our knowledge about the ligands and their interactions with the macromolecular target. Then, in both the similarity search and phenomenological approaches, the essence of machine learning is to infer the relationship between the quantities $\mathbf{x}_i$ and $\mathbf{y}_i$. In the ligand docking case, $\mathbf{y}_i$ denotes the effect that the i^{th} ligand has on the target, i.e., $\mathbf{y}_i$ (such as binding complementarity score) can be seen as the corresponding response value of $\mathbf{x}_i$. In the similarity search approach, $\mathbf{y}_i$ may denote how similar the i^{th} ligand is to a (set of) predefined ligand(s). However, in practice, for both approaches, $\mathbf{y}_i$ is taken as a class label. For example, in the two-class classification case, the label variable $\mathbf{y}_i$ is binary valued: $\mathbf{y}_i > \rho$ or $\mathbf{y}_i < \rho$, where ρ is a real-valued constant which is assumed to be zero in the rest of the paper; $\mathbf{y}_i > \rho$ if the ligand $\mathbf{x}_i$ belongs to class ω_1, i.e., the ligand is active against the predefined target, or $\mathbf{y}_i < \rho$ if the ligand $\mathbf{x}_i$ belongs to class ω_2—that is, the ligand is inactive against the target. (Note that in the labeling scenario, $\mathbf{y}_i$ is just a scalar.) $\mathbf{x}_i$ is an l-dimensional vector l elements of which represent the information we have about the l different relevant (physical, statistical, mathematical, or topological) properties of the i^{th} ligand. In the virtual screening literature, $\mathbf{x}_i$'s are often termed input, attribute, or descriptor vectors that mathematically inhabit an abstract l-dimensional space, $\mathscr{M}$, the input space (in this work $\mathscr{M}$ is assumed to be a Euclidean space).

Winnow (WN),[1] potential function method (PFM),[5] support vector machines (SVM),[6] random forest (RF),[7] rule induction (RI),[8] regularized network (RN),[9] Laplacian-modified naive Bayesian (LmNB),[10] binary kernel discrimination (BKD),[11,12] Multiperceptron Neural Networks,[13,14] and Gaussian process (GP),[15,16] to name but a few, are examples of machine learning algorithms that can be used to generate classifiers. Of special interest to us is the

* Address correspondence to hym21@cam.ac.uk.
† University of Cambridge.
‡ Novartis Institutes for BioMedical Research, Inc.

10.1021/ci100022u
Published on Web 12/15/2010

development of simple, fast, and accurate binary classifiers to analyze large chemical data sets. Currently popular kernel-based algorithms[6,15,17−19] can yield classifiers with impressive predictive power.[14,15,20] Unfortunately, these algorithms can become expensive computationally, as described below.

In this paper we revisit PFM, a very simple kernel-based method that was used in the past to recursively generate a binary classifier for pattern recognition.[5,21−23] This method, introduced by Aizerman et al.[5] over four decades ago (and preceding most of the above listed machine learning algorithms), is mathematically related to other kernel-based algorithms, such as SVM, GP, RN, and BKD. PFM can be computationally much cheaper than these methods. The following section gives a short overview of the basic idea behind the PFM algorithm. A crucial limitation of the algorithm is discussed, which we believe is the main reason why this simple method has been overlooked by the chemoinformatics community. We then develop how one may heuristically overcome the main drawback of the PFM approach while maintaining the simplicity of the method. The third section of the paper presents the generalization performance of binary classifiers that were built by using the augmented version of PFM. Also in that section we discuss how the new classifying technique compares against the popular SVM binary classifiers and two nonkernel binary classifying algorithms (WN and LmNB) that have recently been employed to classify the chemical data set used in this paper. The final section gives our concluding remarks.

Before we proceed and turn to the next section, we define the meaning of certain words and phrases that are frequently used in the paper: Learning here means supervised learning; kernel denotes a nonlinear reproducing kernel unless otherwise stated; decision function, discriminant function, and classifier are used interchangeably, and in the following discussions, kernel perceptron and PFM are used interchangeably. All variables, domains, functions, and spaces are real valued. Finally, it is assumed that the test (unseen) and training data sets are drawn from the same population.

POTENTIAL FUNCTION OVERVIEW

We state from the outset that in this paper we concentrate only on the PFM version in which the statistical properties of the pattern classes do not play a role in the derivation of the algorithm.[24,25] In the following brief overview of the PFM, we closely follow Aizerman et al. and Young and Calvert.[5,21] Aizerman et al. introduced their potential function concept in 1964. Subsequently they, with other researchers, studied and extensively modified the idea in a series of papers.[24−26] Let $\mathbf{x}$ be a training data point in $\mathscr{M}$ (the descriptor space) with a predefined class label. In the area of pattern classification, it has been known for some time that a complicated discriminant function $d(\mathbf{x})$ defined over $\mathscr{M}$—assuming that it exists—can be easily estimated in a feature space, $\mathscr{H}$ (known in the early days of pattern recognition research as the linearization space), as

$$d(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x}) \begin{cases} >\rho & \mathbf{x} \text{ belongs to } \omega_1 \\ <\rho & \mathbf{x} \text{ belongs to } \omega_2 \\ =0.0 & \text{defines the decision surface} \end{cases} \quad (1)$$

where $\mathbf{x} \in \mathscr{M}$, ρ is as defined before, $\{\phi_j(\mathbf{x})\}_{j=1}^{M}$ is a set of linearly independent basis functions defined over $\mathscr{M}$ that span an M-dimensional feature space $\mathscr{H}$, $M \leq \infty$, and $\alpha_j \in \mathscr{R}$ (real number). In simple terms, eq 1 says that each point $\mathbf{x} \in \mathscr{M}$ is assigned to a point in $\mathscr{H}$ with coordinates $\phi_j(\mathbf{x})$ ($j = 1, 2, ..., M$), where the coefficients α_j can be estimated by employed learning schemes developed for training conventional linear classifiers. (Note that if the data sample elements are linearly separable in $\mathscr{M}$, then $\phi_j(\mathbf{x})$ can be set to x_j, $\mathbf{x} = \{x_1, ..., x_j, ..., x_l\}$.)

Formally, eq 1 defines a binary classification rule: $d(\mathbf{x}) > \rho$ and $d(\mathbf{x}) < \rho$ indicate that $\mathbf{x}$ belongs to ω_1 and ω_2, respectively; $d(\mathbf{x}) = \rho$ is the discriminating boundary (decision surface) between the two classes in $\mathscr{M}$ (or in $\mathscr{H}$). In principle, M, the dimension of $\mathscr{H}$, can be infinite. However, in practice, it is impossible to use an infinite number of basis functions.[21,22] Even when M is finite, for a classification problem with many descriptors, computing $\phi_j(\mathbf{x})$ can become computationally too difficult.[22] Aizerman et al. circumvented this problem by cleverly recasting the above discriminant function in eq 1 into an equivalent form, such that d($\mathbf{x}$) can be expressed as a combination of (a finite set of) two variable functions $k(\mathbf{x}_i, \mathbf{x}; \tau)$

$$\mathrm{d}(\mathbf{x}) = \sum_{i=1}^{L} b_i k(\mathbf{x}_i, \mathbf{x}; \tau) \quad (2)$$

where each $k(\mathbf{x}_i, \mathbf{x}; \tau)$ is associated with a given training data point $\mathbf{x}_i$, τ is a tunable parameter, and L and b_i are as described below. These authors called $k(\mathbf{x}_i, \mathbf{x}; \tau)$ a potential function and stipulated that this function should be defined and bounded over $\mathscr{M}$. They argued $k(\mathbf{x}_i, \mathbf{x}; \tau)$ must be related to $\phi_j(\mathbf{x})$. Nowadays, $k(\mathbf{x}_i, \mathbf{x}; \tau)$ is actually what is known in kernel-based methods, such as SVMs, as the kernel function associated with $\mathscr{H}$. For more accessible discussions on the properties of the kernel function and its role in kernel-based algorithms, we refer the reader to refs 17, 19, and 21.

It was demonstrated that $d(\mathbf{x})$,[5,21,22,27] the PFM decision function, can be estimated iteratively using the training data set as follows:

$$d_i(\mathbf{x}) = d_{i-1}(\mathbf{x}) + b_i k(\mathbf{x}_i, \mathbf{x}; \tau) \quad (3)$$

with $d_i(\mathbf{x})$ being the i^{th} estimate of the decision function, $d_0(\mathbf{x})$ is arbitrarily set to a real value as an initial estimate of $d(\mathbf{x})$, and $b_i \in \mathscr{R}$ (real number set) that can be given as

$$b_i = \begin{cases} -1.0 & \text{if } d_{i-1}(\mathbf{x}) \geq \rho \text{ and } \mathbf{x} \text{ belongs to } \omega_2 \\ +1.0 & \text{if } d_{i-1}(\mathbf{x}) \leq \rho \text{ and } \mathbf{x} \text{ belongs to } \omega_1 \\ 0.0 & \text{otherwise} \end{cases} \quad (4)$$

with $i = \{1, 2, ..., L\}$, ρ is as defined before, L is the number of corrections to make during the estimation of the decision function phase: If the molecule represented by $\mathbf{x}$ belongs to class ω_1 but the estimated decision function $d_{i-1}(\mathbf{x})$ predicts the molecule does not belong to ω_1, then set b_i to +1; if the molecule belongs to ω_2 but the algorithm predicts it does not belong to ω_2, then set b_i to −1; and if the algorithm predicts correctly which class the molecule is a member of, then do nothing. This procedure is repeated until no more correction is required. This is basically applying a perceptron-like procedure to a generalized decision function, where $d_0(\mathbf{x})$ can be viewed as the bias term.

Aizerman et al. went further and alluded to the relationship between the convergence rate of the above training algorithm and the predicting power of the binary classifier it generates. They correctly noted that the number of corrections to make during the training phase (eqs 3 and 4) is inversely proportional to the square of the *infimum* value of the estimated decision function $d(\mathbf{x})$. The infimum value of $d(\mathbf{x})$, $\inf_{\mathbf{x}\in\mathscr{M}}|d(\mathbf{x})|$, is the distance (or the margin) between the separating hyperplane ($d(\mathbf{x}) = 0$) in $\mathscr{H}$ and the image of training datum ($\mathbf{x}$) nearest to the separating hyperplane. Here by image we mean the point, $\phi(\mathbf{x})$, in $\mathscr{H}$ that corresponds to the descriptor vector $\mathbf{x}$ in $\mathscr{M}$. (Recall that the separating hyperplane in $\mathscr{H}$ is equivalent to a nonlinear class partitioning/decision boundary in $\mathscr{M}$.) For accessible discussions on the relationship between the margin and the predicting power of kernel-based binary classifiers, see chapters 2 and 7 in refs 20 and 28, respectively. In simple terms, for a given kernel function $k(\mathbf{x}_i, \mathbf{x}; \tau)$, the kernel perceptron algorithm, as given in eqs 3 and 4, eventually converges to a decision function $d(\mathbf{x})$—assuming it exists—that classifies all the training data points.[5,28,29] However, this does not indicate/guarantee that the generated decision function $d(\mathbf{x})$ is the optimal classifier for the problem at hand.

In typical practical situations the data can be finite and noisy and the classes may overlap. Unfortunately, mapping the input spaces associated with these types of data points into high (sometimes infinite) dimensional feature space $\mathscr{H}$ and then employing the learning algorithm given in eqs 3 and 4 inevitably leads to overfitting classifiers.[21,24] Despite its extreme simplicity, both conceptually and computationally, the tendency of the PFM algorithm to yield overfitting classifiers (with respect to the available noisy and finite data set) is one of the reasons that PFM—or for that matter the kernel-based perceptron—has not been taken up in chemoinformatics. Fortunately, statistical learning theory provides some valuable insights that can help us address the aforementioned PFM limitation.[30,31,32] In brief: learning theory shows that it is the training data set size and the complexity of the functions (one of which is $d(\mathbf{x})$) in $\mathscr{H}$, rather than the size of the feature space $\mathscr{H}$, that actually matters in controlling the generalization ability of the estimated $d(\mathbf{x})$.[28,30−32] In other words, the level of complexity of $d(\mathbf{x})$ and its infimum value (which is as defined above) are related. A discriminant function is considered very complex if it is too flexible in relation to the data sample. In common terminology, controlling the complexity of $d(\mathbf{x})$ means striking an optimal compromise between the bias of $d(\mathbf{x})$ (very simple function) and the variance of $d(\mathbf{x})$ (very complex function) with respect to the available data sample. Basically the emphasis is on yielding a classifier that simultaneously reduces the number of misclassifications over both the training and the test data sets.

On the basis of statistical learning theory, the widely used kernel-based pattern class classifying algorithms are equipped with mechanisms that ideally control the complexity of $d(\mathbf{x})$ to safeguard $d(\mathbf{x})$ against overfitting the training data set. PFM is not endowed with similar safeguards.

In SVM, for instance, to address the overfitting issue, Cortes and Vapnik (CV)[28,30−33] introduced what they called soft-margin SVMs that are endowed with practical schemes that prevent SVM from generating overfitting classifiers when the training data points are noisy and finite or when the classes are overlapping. In CV's approach, the space $\mathscr{H}$ (where $d(\mathbf{x})$ lives) is explicitly controlled to a less complex space[30,32] for $d(\mathbf{x})$ to generalize well to unseen patterns. In simple language, the soft-margin SVM classifier is generated by inductively matching the size of $\mathscr{H}$ to the available finite amount of training data for the problem at hand.[20,30,32] In practice, this is achieved by optimizing a set of tunable parameters—the so-called slack variables ζ's and the margin (or penalty) parameter C—that are augmented to the standard SVMs.[15,20,28]

GP and RN methods avoid generating overfitting classifiers by invoking the principles of regularization theory.[15,16,20] These methods employ regularization parameters to control the complexity of $d(\mathbf{x})$ to construct a decision function that gives small error rates on both the training and the test data sets.

Interestingly it was demonstrated that the regularization techniques and the concept on which CV's approach is based—the so-called structural risk minimization (SRM) principle—are at a fundamental level connected.[15,18,34,35] Thus it might be argued that if the appropriate kernel function and the tunable parameters, ζ's and C (in the case of SVM) or regularization parameters (in the case of GP and RN), are known and given, then the learning of an optimal decision function from the training data set reduces to solving the following equation for the c_i coefficients:

$$d(\mathbf{x}) = \sum_{i=1}^{N} c_i k(\mathbf{x}_i, \mathbf{x}; \tau) \tag{5}$$

which means the optimal discriminant function can be formally expressed in terms of the known N training data points (cf., eq 2). Even though the RN and GP methods and the SVM algorithm are at a fundamental level connected, in practice, the two approaches have some computational advantages over each other. In the SVM case, the solution of eq 5 yields a sparse coefficient vector $\mathbf{c}$, sparse in the sense that most of the c_i coefficients are zeros. Hence a small portion of the training data set and the corresponding coefficients are required to be retained in core memory in order to use them in the test (prediction) phase. This means, in the SVM classification algorithms, the computational complexity in the test phase might become less problematic.[32] However, solving eq 5 for the coefficient vector $\mathbf{c}$, in particular when the size of the data set is large, can become computationally prohibitive,[16,20,32] as one has to solve a quadratic optimization problem with linear constraints.[20,32] In GP and RN computational difficulties can arise in both the training and test phases.[16,20] Here, obtaining the coefficients requires solving N regularized linear equations, which does not guarantee the sparsity of the coefficient vector $\mathbf{c}$. Therefore in the test phase, all the training data points and their corresponding c_i coefficients are required to make classification predictions. Self-evidently, employing these types of kernel methods can become computationally infeasible when the training data set is large. For a more detailed discussion on the drawbacks that arose from solving eq 5 in the three methods that we have outlined and the research being undertaken to address them, the reader is referred to ref 16. (In the above discussion the technical details of how

the c_i coefficients are computed were omitted as they have been described widely in the literature, see refs 14, 16, 20, and 28.)

Finally we briefly comment on the BKD, which is another popular binary classifying tool that is based on the kernel concept and is easy to implement.[11,12] The central idea of BKD is to estimate two probability density functions, $p(\omega_1|\mathbf{x})$ and $p(\omega_2|\mathbf{x})$; if $p(\omega_1|\mathbf{x}) > p(\omega_2|\mathbf{x})$, $\mathbf{x}$ is assigned to class ω_1, otherwise $\mathbf{x}$ is assigned to ω_2. Unfortunately, in BKD, $p(\omega_1|\mathbf{x})$ and $p(\omega_2|\mathbf{x})$ are estimated via a nonparametric probability density estimator,[11] a slight variant version of eq 5, in which all the training points are required to make classification predictions; that is, BKD, too, can suffer a severe computational bottleneck in the test phase when the training data set is large.[14] Thus, like the GP and RN, we do not consider this method further in the rest of the paper. The reader desiring a more extensive treatment of this algorithm is referred to ref 37.

Spurred by: (i) the simplicity of PFM when it comes to estimating the b_i coefficients in eq 2, (ii) the fact that the generalization ability of $d(\mathbf{x})$ is connected to the complexity of $\mathscr{H}$, (iii) Aizerman et al.'s observation that the predicting power of $d(\mathbf{x})$ is related to $\inf_{\mathbf{x}\in\mathscr{M}}|d(\mathbf{x})|$, and (iv) the fact (see chapter 7 of ref 28) that PFM and kernel perceptrons are related, we decided to look into whether augmenting PFM with a simple regularizer can remedy the aforementioned limitation of PFM.

Naturally one wishes to employ a regularization scheme that on the one hand maintains the simplicity of PFM's learning algorithm, that is, eqs 3 and 4, while on the other hand improving the generalization ability of $d(\mathbf{x})$. A geometrical interpretation of eqs 3 and 4 indicates that PFM generates a classifier by successively changing the orientation of the decision function (hyperplane) in $\mathscr{H}$ until the algorithm converges.[5] (Recall that a converged kernel perceptron algorithm does not necessarily yield a separating hyperplane with a good predicting power.) Thus any scheme that chooses the orientation in which the discriminant function simultaneously reduces the number of misclassifications over both the training and test data sets can be considered as a valid regularizer. Fortunately, the early stopping criterion that is widely employed in training neural networks[14,15] can be used to achieve this goal. However, for the early stopping procedure to be of any use, besides the training data set, it requires a validation data set and a metric that measures how well the generated discriminant function is classifying the training and validation sets as the learning procedure proceeds. As we are specially interested in classifying binary chemoinfomatics data sets that are generally heavily biased toward one class (e.g., inactive molecules), Fmeasure[36] (eq 6) can be a good figure of merit for monitoring the performance of $d(\mathbf{x})$ during the training phase of R-PFM; R-PFM denotes regularized PFM.

From the coupling of the early stopping procedure with eqs 2−4, it is clear that R-PFM is, like a soft-margin SVM, a sparse kernel method in the sense that only the training data points that correspond to $b_i = \pm 1$ (see eq 4) are required to estimate the appropriate decision function. In other words, to estimate $d(\mathbf{x})$ only the distinct error-correcting training data points (henceforth N_{dp}) are required; L (number of corrections made during the training) $= \sum_i^{N_{\mathrm{dp}}} n_i$, where n_i denotes the number of times that $\mathbf{x}_i$ was misclassified[5,25] during the training phase.

There are various performance measuring metrics that can be chosen to evaluate the generalization ability of the classifiers generated via application of R-PFM. In virtual screening (which is one of our primary interests) it is often the case that the available data set is heavily biased toward the class of biologically inactive molecules. Thus, in this work the Fmeasure was deemed a highly relevant metric, which can be calculated according to

$$\text{Fmeasure} = \frac{2 \times R \times P}{R + P} \tag{6}$$

with P (precision) and R (recall) being defined, respectively, as follows

$$P = \frac{t_p}{(t_p + f_p)} \tag{7}$$

$$R = \frac{t_p}{(t_p + f_n)} \tag{8}$$

where t_p is the number of true positives; f_p is number of false positives; and f_n is the number of false negatives. It is often pointed out that Recall−Precision−Fmeasure measure is sensitive to class skewness. Stratified random splits of the data set can rectify this problem.

We follow the generally accepted practice that performance with relative error of about 25% or lower is considered good. Thus, in this work a classifier that yields Fmeasure values whose relative errors are higher than 26% was considered inefficient. Here the relative error in the Fmeasure value is defined as $|(1 - \text{Fmeasure value})| \times 100$. Also the Wilcoxon signed rank test[38] was used to compare the generalization abilities of R-PFM classifiers against the classification prediction performances of SVM, LmNB, and WN classifiers over the given data set.

Before we proceed to the results and the discussion section, in the following we briefly note some of the advantages (and disadvantages) of R-PFM over the soft-margin SVM method as pattern classification tool: (i) The SVM classifier is a practical realization of the powerful SRM approach of statistical learning theory coupled with the idea of reproducing kernels,[15,30] whereas the R-PFM scheme is based on a heuristically regularized empirical risk minimization (ERM) approach.[15] Recall that ERM is conceptually far too inferior to SRM. In the SRM approach an upper bound generalization error is minimized, whereas in the ERM scheme only the errors over the training data set are minimized.[15] Thus, in principle, SVM classifiers should have better generalization ability than R-PFM classifiers. (ii) In SVM, to generate a binary classifier, one has to solve a constrained quadratic optimization (quadratic programming) problem, whereas in R-PFM all that is required is a simple kernel perceptron procedure augmented with an early stopping criterion. (iii) Furthermore, in SVM, besides the kernel and margin parameters, N slack variables have to be tuned. This is done through repeatedly solving a constrained quadratic optimization problem, while in R-PFM only the kernel parameters have to be optimized via a repeated kernel perceptron procedure. (iv) Finally (and this a major advantage of R-PFM

over SVM), SVM are formulated for two-class classification problems, and direct extension of SVM to multiclass problems is not straightforward.[20,32,39] Contrast this with R-PFM where this sort of extension is a simple exercise. (Note that by direct extension of SVM to multiclass problems we mean constructing multidecision surfaces by considering all classes at once.)

In summary, the appropriate binary classifier is generated by estimating the optimal values of b_i's and τ via the combination of a generalized perceptron learning algorithm, regularization approach, and data sampling scheme. The typesetting, algorithm 1, illustrates a pseudocode of the proposed algorithm.

The scheme that is outlined in algorithm 1 assumes a multiple random three-way sample splits method.[15] Cross-validation, random subsampling, and bootstrapping, to name but a few, are examples of many alternative data sampling techniques. If one desires to employ the alternative resampling methods, then the steps described in the pseudocode should be accordingly modified.

Table 1. The 12 995 Structures Classified According to Their Biological Activity

activity class	no. of active compounds
alpha1A	753
CA-II	1183
D2	1891
gamma-secretase	118
mGluR5	105
Src	143
tubilin	180
PDE4	549
PDE5	490
CDK4	43
CDK5	54
δ	1473
κ	1191
m1	764
m2	670
μ	1664
CDK1 cyclin B	133
CDK2 cyclin A	211
CCK1	639
CCK2	741

Algorithm 1 Pseudocode of R-PFM

Given N training labelled examples, a data set,
$S = < (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) >$, where $y_i \in \{-1, +1\}$
J : how many times to sample and partition the data set.
NE: Number of Epochs.
Choose an appropriate potential function, $k(\mathbf{x}_i, \mathbf{x}; \tau)$
for $j = 1, J$ **do**
(i) – Randomly partition S into training, validation and test datasets.
(ii) – Initialise $d_0(\mathbf{x}) \Leftarrow 0.0$.
(iii) – Set τ.
for $l = 1$ to NE **do**
– Build a classifier using the training dataset and the recursion given by eqs 3–4.
– Evaluate the obtained classifier employing the validation dataset.
end for
– Using the *early stopping criterion*, record the value of l (*'l-optimal-value'*) that yields the best Fmeasure value over the validation dataset for the given τ value.
– Go back to Step (ii) until the 'optimal value' of τ is obtained.
– The optimal τ, the recorded *'l-optimal-value'*, N_{dp} training data points and their associated N_{dp} coefficients constitute the estimated classifier.
– Using this classifier compute the Fmeasure value over the test set: $(\text{Fmeasure})_j$
end for
Output: The mean value of the chosen J Fmeasure values,

$$(\text{Fmeasure})_{mean} = \frac{1}{J}\sum_{j=1}^{J}(\text{Fmeasure})_j$$

Optional Outputs: $(\text{recall})_{mean}$ and $(\text{precision})_{mean}$.

RESULTS AND DISCUSSIONS

To demonstrate the performance or rather the efficiency of R-PFM as a binary classifying tool, we tested R-PFM classifiers on classifying a bioactivity data set that was taken from the WOMBAT 2007.1 database.[40]

Data Set and Attributes. The data set consisted of 12 995 compounds in 20 activity classes. This data set was compiled, studied, and analyzed by Nigsch et al.[1] They selected 20 targets of pharmaceutical interest from the WOMBAT database.[42] The targets included close homologues as well as very different target classes. Compounds were identified as active if the K_i or IC_{50} values assigned were equal to or below 10 μM against the biological target. The 12 995 compounds in the data set were active against one or more of the 20 targets. The complete list of the targets is given in Table 1. Columns one and two of the table show the activity class (target) and how many compounds (out of the 12 995) are active against that target. For full analysis of the data set, the reader is referred to ref 1. For each compound 165 attributes were calculated by MOE[41] from two-dimensional (2D) structures.

A random three-way sample splits method was employed. In each activity class the data set was randomly (J times, see algorithm 1) partitioned into three sets: training, validation, and test sets. On the basis of the work of Kearns,[42] 10 397 (80%) and 2598 (20%) compounds of the data sample set were allotted to become the training and validation and the test sets, respectively. Just under 15% of the 10 397 compounds was assigned to the validation set, and those remaining were assigned to the training set.

As Table 1 illustrates, in each activity class there are more inactive than active compounds. Thus a stratified data partitioning was used to account for the skewness. This guarantees that training, validation, and testing subsets will have approximately equal class distributions. In this regard, about 70% of the inactive and 70% of the active molecules were chosen as the training set, and 10% of the inactive and 10% of the active molecules were selected as the validation set. The test set consisted of the remaining 20% of the inactive and 20% of the active molecules. (Note that this data division is consistent with the data partitioning described in the preceding paragraph.)

RESULTS

It is extremely important to point out that here only the performance of the classifiers is being investigated given a set of molecular descriptors and that with a different set of descriptors the classifiers might be able to yield different classification results. In other words, this work is concerned with one of the two pattern recognition subproblems, making decisions on assigning a given descriptor vector to a class. We are not dealing with the other subproblem, i.e., the so-called feature extraction problem. The 165 attributes that were employed in this study might not be the optimal set of descriptors or features for the classification problem at hand.

Initially we randomly chose 14 activity classes: alpha1A, CDK2 cyclin A, CDK4, D2, CA-II, gamma-secretase, mGluR5, PDE4, PDE5, Src, tublin, μ, δ, and κ. For each one, five R-PFM classifiers were constructed using five different potential functions: radial basis function kernel; Lorentzian kernel; sinc kernel; tanh kernel; and polynomial kernel. After some preliminary classification trials on the 14 activity classes, it was found that R-PFM(R) and R-PFM(L) classifiers performed the best; in these preliminary trails J was set to 1, see algorithm 1. R-PFM(R) and R-PFM(L) denote R-PFM in which the potential functions were radial basis functions $e^{-\beta(\mathbf{x}-\mathbf{x}_i)^2}$ and Lorentzian kernel $1/[1 + \gamma(\mathbf{x} - \mathbf{x}_i)^2]$, respectively. ($\beta$ and γ denote τ, and $\mathbf{x}$, $\mathbf{x}_i$, and τ are as defined before.) Out of the two, R-PFM(R) and R-PFM(L), the performance of R-PFM(L) was marginally better.

Using R-PFM(L), 20 classifiers, one per activity class, were then constructed. Although in each activity class the size of the data set was significantly large, in some activity classes (as Table 1 shows) there were few active compounds. In these scenarios to account for the sparsity of active molecules over the input space, J (see algorithm 1) was set to 5 whenever the number of active molecules in the activity class was less than 600, i.e., the number of active compounds in the test data set was less than 120.[43] Thus, in the following analysis, the reported Fmeasure, recall, precision, N_{dp}, N_{sv} (as defined below), and CPU-time values are averages in the case of $J > 1$.

Figure 1 illustrates the generalization ability of the 20 R-PFM(L) classifiers with respect to the |(1 − Fmeasure value)| × 100 criterion. Clearly, the classifiers can be considered efficient in 10 out of the 20 activity classes, as the relative errors in the Fmeasure values were less than 26%. (In Figure 1, and in subsequent figures, SVM and PFM denote SVM-Light and R-PFM(L), respectively.)

For comparison, we classified the 20 activity classes using the SVM-Light package of Joachims.[44] For classification

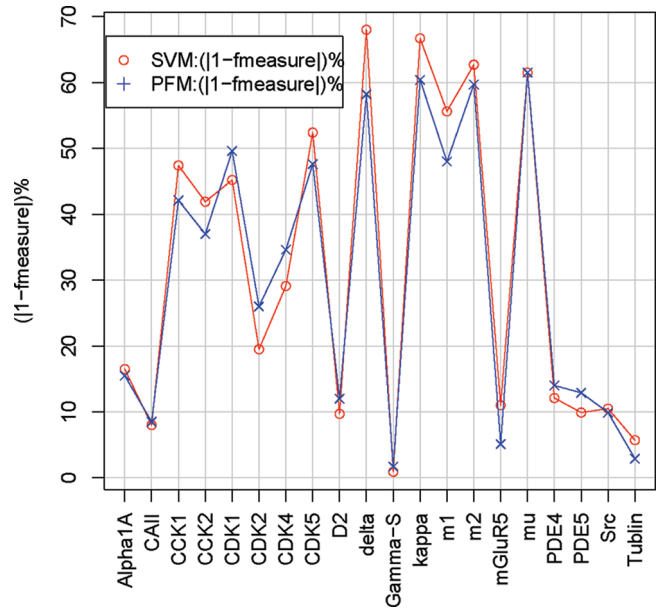

Figure 1. Figure illustrates the generalization ability of the 20 R-PFM(L) classifiers on the 12 995 WOMBAT structures with respect to the |(1 − Fmeasure value)| × 100 criterion. In the figure, CDK1, CDK2, and gamma-S denote CDK1/cyclin B, CDK2/cyclin A, and gamma-secretase, respectively; and SVM and PFM in the legends denote SVM-Light and R-PFM(L), respectively.

purposes, SVM-Light is arguably one of the best implementations of soft-margin SVMs at the moment. For each activity class, the three classifiers that were constructed as SVM-Light have three default kernel functions: polynomial, radial basis, and tanh kernels. Once again out of the three available kernels, the radial basis kernel was deemed suitable for the classification problem at hand.

The |(1 − Fmeasure value)| × 100 values yielded by the 20 SVM-Light classifiers on the test data sets are also given in Figure 1. Self-evidently the two sets of classifiers did very well to correctly classify the same 10 activity classes and similarly performed poorly (to a similar degree) in classifying the remaining 10 activity classes.

Similar patterns appear in the other figures of merit as the three panels in Figure 2 illustrate. The agreement between the classification performance of the two sets of R-PFM(L) and SVM-Light classifiers is very good indeed.

Table 2 gives the numerical values of the three performance measures. Table 2 also shows the values of γ and β that were used to generate the appropriate classifiers. Note that in the case of $J > 1$, unlike the Fmeasure, recall, precision, N_{dp} and N_{sv}, the values of β and γ are not mean values. Instead the reported β and γ values denote the values to which the tunable parameters were set to obtain the optimal binary SVM-Light and R-PFM(L) classifiers when j is 1, i.e., in the first data partitioning case, see algorithm 1. (In fact in most cases, we observed that the optimal values of β and γ in the remaining four data partitionings were not significantly different from the reported parameter values.)

In all SVM-Light calculations ζ's and C were set to 0.1 and 700, respectively, whereas in the case of R-PFM(L) ρ and $d_0(\mathbf{x})$ were both set to 0.0 (see eqs 3 and 4).

Remarkably Table 2 shows that the number of distinct training data points N_{dp} (column three) and support vectors N_{sv} (column eight) required in R-PFM(L) and SVM-Light to generate the corresponding binary classifiers, respectively,

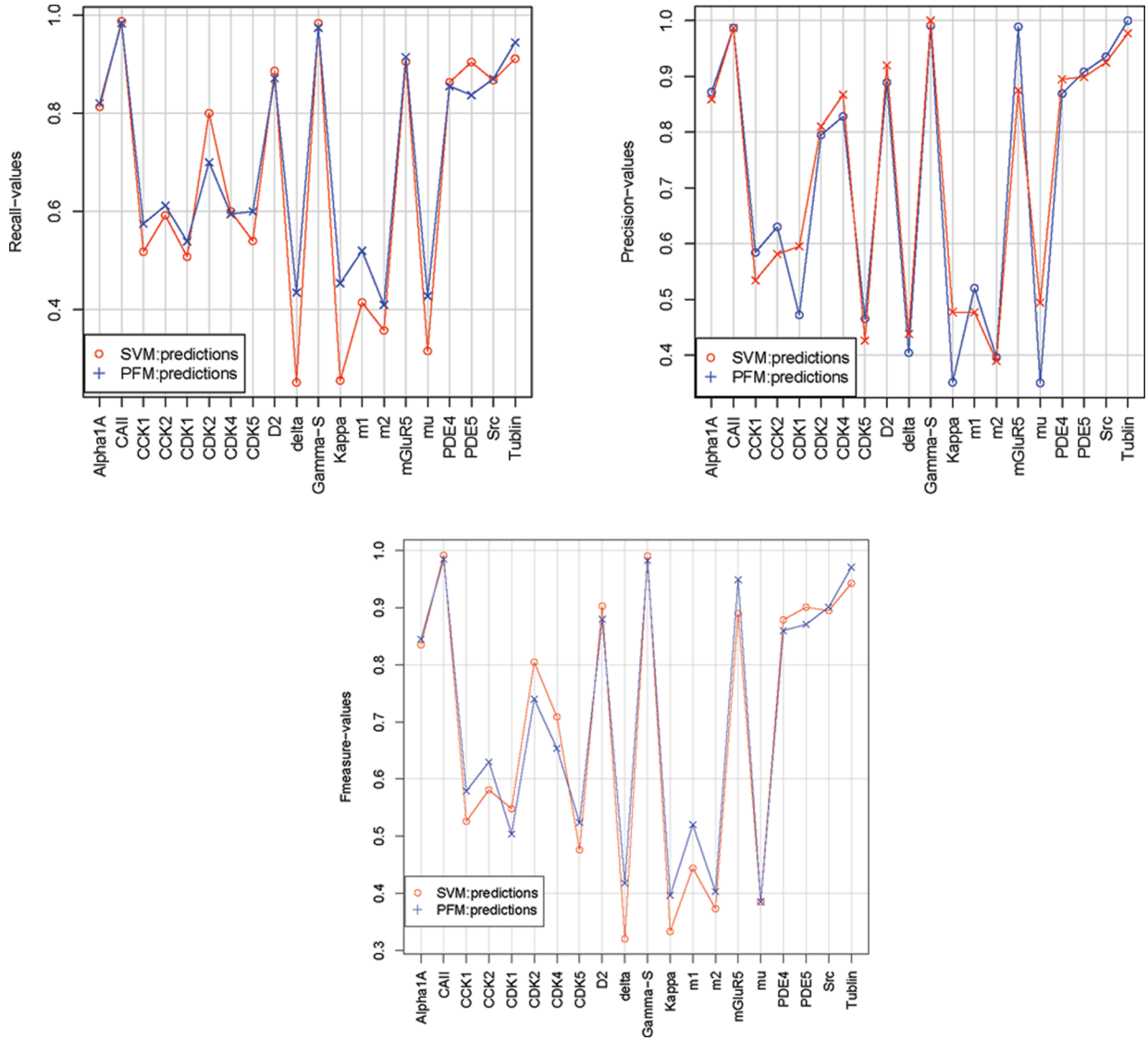

Figure 2. The top left and right and bottom panels, respectively, denote the recall, precision and Fmeasure values yielded by the R-PFM(L) and SVM-Light classifiers over the 20 test activity class data sets. In the three panels, CDK1, CDK2, and gamma-S denote CDK1/cyclin B, CDK2/cyclin A, and gamma-secretase, respectively.

were very similar, except that in a few cases $N_{sv} \approx 2 \times N_{dp}$. We observed similar findings when R-PFM(L) and SVM-Light were used to generate binary classifiers using some UCI benchmark data sets, such as the Mask data set.[45] These results, in general, support our arguments given in the preceding section. We suggested that a closer look into how R-PFM works when coupled with a judiciously chosen regularization technique may allow R-PFM to mimic (in performance at least) soft-margin SVM even though R-PFM is conceptually different and inferior to SVM as explained earlier.

We then investigated whether the R-PFM(L) and SVM classifiers were classifying the same set of compounds in each activity class; that is, do R-PFM(L) and SVM-Light classifiers assign correctly (or incorrectly) compound **x** to the appropriate activity class or not? In order to answer this question, we concentrated on the 10 activity classes in which the R-PFM(L) and SVM-Light classifiers performed well. (Recall that in each activity class, the size of the test data set is 2598 if $J = 1$ and 12 990 in the case of $J = 5$.)

Column four of Table 3 shows the cardinality of the set $A \cap B$, $\mathcal{N}(A \cap B)$, i.e., the number of compounds in a certain activity class that both R-PFM(L) and SVM-Light methods misclassify. A and B denote the set of molecules (out of the number of test molecules in that particular activity class) that were misclassified by the R-PFM(L) and SVM-Light classifiers, respectively. And in the table, $\mathcal{N}(A)$ and $\mathcal{N}(B)$ denote the number of molecules in A and B, respectively. As expected the values of $\mathcal{N}(A)$, $\mathcal{N}(B)$, and $\mathcal{N}(A \cap B)$ are small compared to the size of the corresponding test data set; however, an intriguing observation is the correlation between the cardinalities of the sets and the number of support vectors (N_{sv}) and distinct data points (N_{dp}); in particular, the correlation between $\mathcal{N}(A)$ and $\mathcal{N}(A \cap B)$ and N_{dp} barring the activity class where there was no misclassification. Why this is the case may warrant further investigation. It is worth noting that in the case of $J = 5$, $\mathcal{N}(A)$ (and $\mathcal{N}(B)$) denotes the mean value over the aggregate number of misclassifications out of the 12 990 test molecules for a given class activity.

We also compared the classification efficiency of R-PFM(L) classifiers against the predicting powers of LmNB and WN classifiers on the same 20 activity classes. As Figure 3 (bottom panel) illustrates the Fmeasure values predicted by the three sets of classifiers agree well. Once again, the

Table 2. Comparison between the Predicting Powers of R-PFM(L) and SVM-Light Classifiers on 12 995 WOMBAT Structures[a]

	R-PFM					SVM-Light				
activity class	γ	N_{dp}	R	P	Fmeasure	β	N_{sv}	R	P	Fmeasure
Alpha1A	68.0	498	0.820	0.872	0.845	0.40	1036	0.813	0.859	0.835
CA II	3.33	177	0.983	0.987	0.985	−	228	0.996	0.987	0.992
CCK1	2.80	591	0.575	0.584	0.579	−	800	0.518	0.534	0.526
CCK2	1.51	787	0.612	0.649	0.630	0.04	836	0.592	0.581	0.581
CDK1/cyclin B	0.28	215	0.539	0.472	0.504	−	276	0.508	0.595	0.548
CDK2/cyclin A	39.00	186	0.700	0.795	0.740	−	271	0.800	0.810	0.805
CDK4	2.90	51	0.595	0.828	0.654	0.40	118	0.600	0.867	0.709
CDK5	0.28	90	0.600	0.4650	0.524	0.04	115	0.540	0.426	0.476
D2	49.80	829	0.871	0.889	0.880	−	1575	0.886	0.920	0.903
δ	1.030	2353	0.435	0.404	0.418	−	2722	0.252	0.438	0.320
gamma-secretase	98.00	57	0.974	0.991	0.983	−	130	0.983	1.000	0.991
κ	0.38	2022	0.454	0.351	0.396	−	2251	0.256	0.477	0.333
m1	1.05	1050	0.520	0.520	0.520	−	1055	0.415	0.477	0.444
m2	1.08	894	0.410	0.396	0.403	−	1107	0.358	0.390	0.373
mGluR5	2.33	41	0.914	0.989	0.949	−	118	0.905	0.875	0.890
μ	0.63	2468	0.428	0.350	0.385	−	3448	0.316	0.494	0.385
PDE4	30.30	410	0.0.855	0.869	0.860	−	626	0.863	0.895	0.879
PDE5	120.00	302	0.0.837	0.908	0.871	−	438	0.904	0.899	0.901
Src	93.00	86	0.870	0.935	0.901	−	221	0.867	0.925	0.895
tublin	6.30	91	0.944	1.000	0.971	−	170	0.911	0.977	0.943

[a] R, P, Fmeasure, N_{dp}, N_{sv}, β, and γ are as defined in the main text.

Table 3. Misclassifications by R-PFM(L) and SVM-Light[a]

activity class	$\mathcal{N}$(A)	$\mathcal{N}$(B)	$\mathcal{N}$(A ∩ B)	N_{dp}
mGluR5	2	12	2	42
gamma-secretase	2	0	0	59
Src	5	5	3	89
tublin	6	4	3	91
CA II	7	4	0	174
CDK2	21	16	11	208
PDE5	23	19	13	281
PDE4	29	25	14	375
Alpha1A	48	48	36	498
D2	119	72	57	829

[a] $\mathcal{N}$(A) and $\mathcal{N}$(B) denote how many molecules out of the test molecules for a given activity class (recall in the case of $J = 1$ the size of the test set is 2598, while in the case of $J = 5$ the size of the test set is 12 990) were misclassified by R-PFM(L) and SVM-Light, respectively. $\mathcal{N}$(A ∩ B) denotes the number of identical molecules (in a given activity class) that were assigned to the wrong class activity by both R-PFM(L) and SVM-Light, and N_{dp} is as defined in the main text.

methods performed well on the same 10 activity classes. However, in the activity class CDK4, the WN classifier predicted an Fmeasure value of 0.748, which is equivalent to 25.2% relative error in the Fmeasure value. The R-PFM(L) and LmNB classifiers predicted Fmeasure values of 0.654 and 0.701 (i.e., 34.6 and 29.9% relative errors in the Fmeasure values), respectively. This means the WN classifier performed well on 11 activity classes. In Figure 3, the top left panel indicates that the recall values yielded by LmNB are on average higher than their corresponding R-PFM(L) and WN recall values. However, as the top right panel in the figure shows, the precision values yielded by the three classifying approaches are (on average) in good agreement. (The numerical values of recall, precision, and Fmeasure are given in columns 2−10 of Table 4.)

Finally the performances of the R-PFM(L) classifiers over the 20 class activities were compared against the generalization performance of the SVM-Light, LmNB, and WN classifiers using Wilcoxon signed rank test statistics, where the null hypothesis was that the two sets of the chosen measure of merit values came from identical populations. In other words, the Fmeasure (recall or precision) values yielded by the R-PFM(L) vs SVM-Light; R-PFM(L) vs LmNB; and R-PFM(L) vs WN classifiers came from identical populations. For a paired test, the p-values that were returned by the Wilcox test function in R [46] are shown in Table 5. Thus, at 0.01 significance level we conclude that the generalization performance of SVM-Light and R-PFM binary classifiers over the given data set is not statistically significantly different in all the three measures of merit. We can also infer from the table that the prediction performance of R-PFM(L) and LmNB is not statistically different in two out of the three chosen metrics. Similar inference can be made in the case of R-PFM(L) and WN. These statistical results are self-evidently in line with the analysis given in the preceding paragraphs. Table 5 also gives Wilcoxon signed rank test statistics of SVM-Light vs R-PFM(L); SVM-Light vs LmNB; and SVM-Light vs WN. The results shown in the bottom half of the table are effectively similar to the results in the upper half of the table. Recall that in this work the crucial metric is the Fmeasure. The better performance of the WN algorithm (with respect to the Fmeasure metric) over SVM-Light and R-PFM(L) methods can arguably be attributed to the fact that the features used in the WN method were highly optimized and especially tailored for this algorithm.[1]

Remark: The results for LmNB and WN classifiers were reproduced from ref 1. In that reference, a Matthew correlation coefficient in lieu of Fmeasure was employed. Another point worth noting is that in this work real-valued attributes were used, while in ref 1 the attributes were binary.

Computational Aspects of R-PFM(L): In all the reported classifiers $d(\mathbf{x})$, the l-optimal (see algorithm 1) values were in the range [3,10].

As described in the preceding section, R-PFM(L) is an extremely simple method to understand and implement. However, as eq 4 indicates, the computational efficiency of

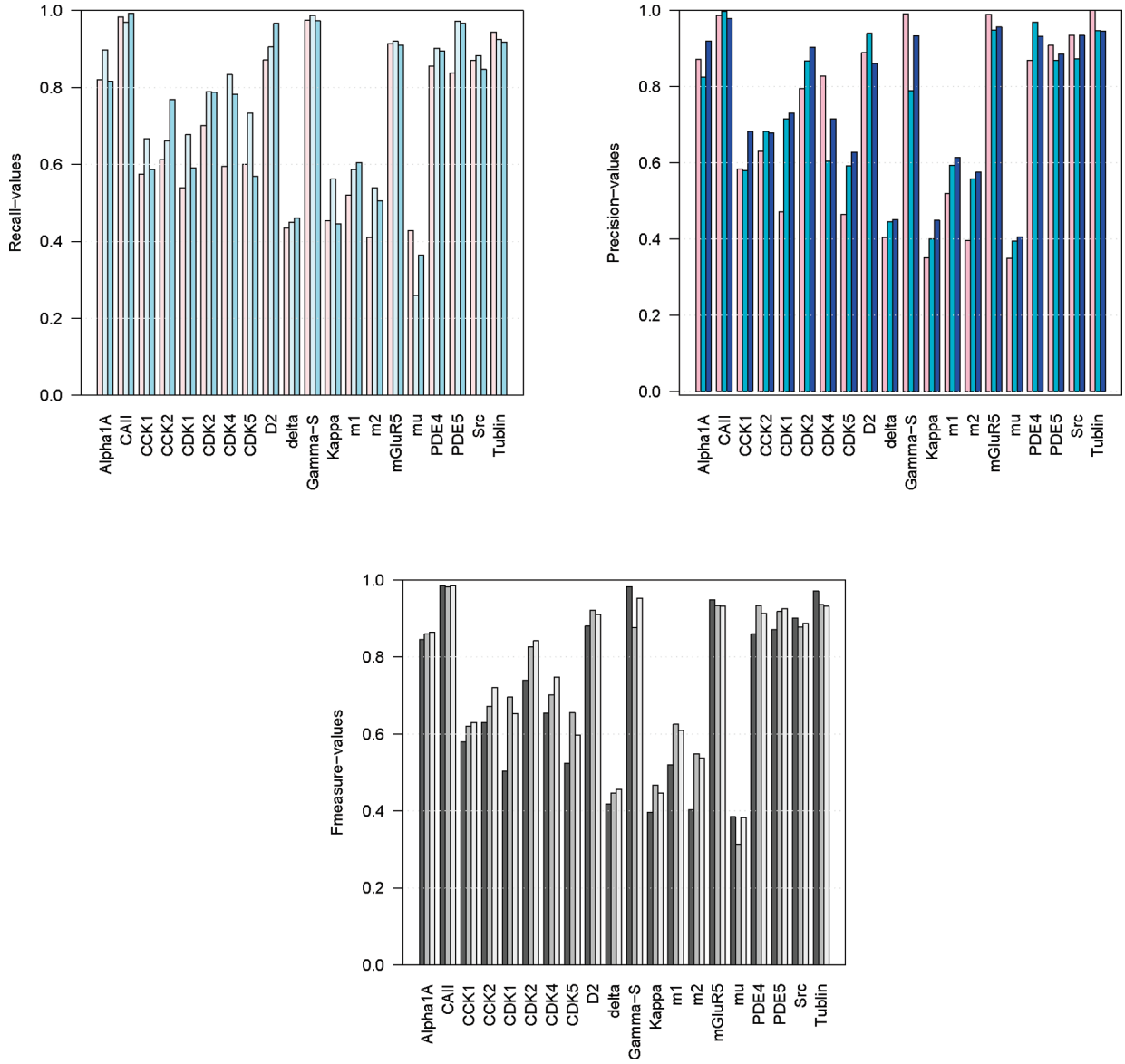

Figure 3. In the top left panel, misty-rose, light-cyan, and light-blue bars denote the recall values yielded by R-PFM(L), LmNB, and WN classifiers, respectively; and the top right panel, pink, cyan, and blue bars denote the precision values obtained with the application of R-PFM(L), LmNB, and WN classifiers, respectively. The bottom panel shows the Fmeasure values yielded by the three classifying algorithms: R-PFM(L), LmNB, and WN, which are denoted by the dark- and light-gray bars, respectively. In the three panels, CDK1, CDK2, and gamma-S denote CDK1/cyclin B, CDK2/cyclin A, and gamma-secretase, respectively.

R-PFM is dependent on the values of N_{dp} and l where the latter (as described above) denotes the size of the input vector. In other words, in classification problems where the values of N_{dp} and l are small, R-PFM(L) can be fast and memory efficient. In this work when N_{dp} was less than 2000, R-PFM(L) was extremely fast. Figure 4 shows that the classification procedure required between 0.05 and 3 min. The figure also demonstrates that when the value of N_{dp} was much higher than 2000, the CPU-time requirements rocketed. For, instance, the algorithm required 9 min in the worst case, which was classifying the μ activity class where the value of N_{dp} was 2468.

The SVM-Light method which was run on the same computer showed similar behavior, see Figure 4, and again, the most CPU-time-consuming classification problem was classifying the μ activity class; it took 17 min.

The R-PFM(L) algorithm was coded in FORTRAN90. The results presented were computed on a 1.60 GHz Intel Celeron M PC. We emphasize that the CPU-time requirements that we noted in the previous paragraphs do not include CPU-time overheads that were spent on obtaining the optimal values of γ, β, C, and slack variables. Thus, the reported CPU-time figures may heavily favor the SVM-Light algorithm, as SVM-Light has more tunable parameters than R-PFM(L).

CONCLUSION

We investigated whether the simple potential function method (PFM) of Aizerman et al. can become a useful classification tool in chemoinformatics provided that it is employed wisely. From the results presented one may cautiously argue that even when the available data set is not linearly separable in a high-dimensional feature space $\mathcal{K}$ (which is the case in many practical and realistic classification problems), a kernel-based perceptron can yield binary classifiers with generalization ability similar to that of the

Table 4. Comparison between the Predicting Powers of R-PFM(L), SVM-Light, LmNB, and WN Classifiers of 12 995 WOMBAT Structures[a]

	R-PFM(L)			LmNB			WN		
activity class	*R*	*P*	Fmeasure	*R*	*P*	Fmeasure	*R*	*P*	Fmeasure
Alpha1A	0.820	0.872	0.845	0.898	0.825	0.860	0.816	0.920	0.865
CA II	0.983	0.987	0.985	0.969	0.998	0.983	0.992	0.979	0.985
CCK1	0.575	0.584	0.579	0.667	0.580	0.620	0.586	0.682	0.630
CCK2	0.612	0.649	0.630	0.661	0.683	0.672	0.769	0.679	0.721
CDK1/cyclin B	0.539	0.472	0.504	0.677	0.716	0.696	0.590	0.731	0.653
CDK2/cyclin A	0.700	0.795	0.740	0.789	0.868	0.827	0.788	0.903	0.842
CDK4	0.595	0.828	0.654	0.833	0.605	0.701	0.782	0.716	0.748
CDK5	0.600	0.465	0.524	0.733	0.592	0.655	0.569	0.628	0.597
D2	0.871	0.889	0.880	0.905	0.940	0.922	0.966	0.861	0.910
δ	0.435	0.404	0.418	0.450	0.445	0.447	0.461	0.451	0.456
gamma-secretase	0.974	0.991	0.983	0.987	0.789	0.877	0.973	0.933	0.953
κ	0.454	0.351	0.396	0.562	0.400	0.467	0.446	0.449	0.447
m1	0.520	0.520	0.520	0.594	0.602	0.625	0.625	0.604	0.614
m2	0.410	0.396	0.403	0.539	0.558	0.548	0.505	0.575	0.538
mGluR5	0.914	0.989	0.949	0.920	0.949	0.934	0.910	0.956	0.932
μ	0.428	0.350	0.385	0.259	0.395	0.313	0.364	0.405	0.383
PDE4	0.855	0.869	0.860	0.901	0.969	0.934	0.893	0.932	0.913
PDE5	0.837	0.908	0.871	0.972	0.869	0.918	0.967	0.886	0.925
Src	0.870	0.935	0.901	0.883	0.873	0.878	0.847	0.934	0.888
tublin	0.944	1.000	0.971	0.925	0.947	0.936	0.918	0.946	0.932

[a] *R*, *P*, LmNB, WN, R-PFM, and Fmeasure are as defined in the main text.

Table 5. Wilcoxon Signed Rank Test on the Prediction Performances of R-PFM(L) vs SVM-Light, R-PFM(L) vs LmNB, and R-PFM(L) vs WN and SVM-Light vs R-PFM(L), SVM-Light vs LmNB, and SVM-Light vs WN over 20 Class Activities[a]

performance measure	*R*	*P*	Fmeasure	*R*	*P*	Fmeasure	*R*	*P*	Fmeasure
R-PFM(L) vs		SVM-Light			LmNB			WN	
p-values	0.0793	0.8249	0.1589	0.00482	0.330	0.0304	0.0349	0.0532	0.00238
SSDA.01	no	no	no	yes	no	no	no	no	yes
SVM-Light vs		R-PFM(L)			LmNB			WN	
p-values	0.0793	0.8249	0.1589	0.000483	0.514	0.0100	0.000708	0.126	0.00182
SSDA.01	no	no	no	yes	no	no	yes	no	yes

[a] SSDA.01 denotes statistically significantly different at 0.01 significance level, and *R*, *P*, and Fmeasure are as defined in the main text.

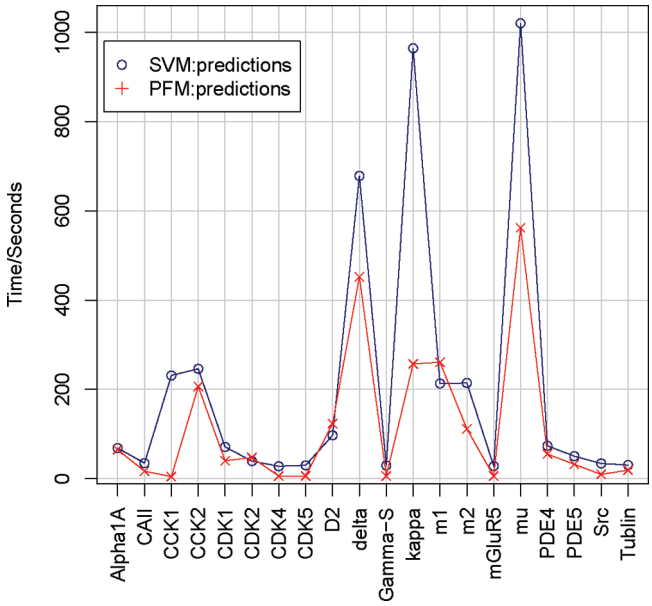

Figure 4. The CPU-time it takes R-PFM(L) and SVM-Light to classify each class activity. The CPU-time for gamma-secretase, mGluR5, Src, tubilin, PDE4, PDE5, CDK4, and CDK5 are averages as described in the text. CDK1, CDK2 and gamma-S denote CDK1/cyclin B, CDK2/cyclin A, and gamma-secretase, respectively. In the legend, SVM and PFM denote SVM-Light and R-PFM(L), respectively.

state-of-the art (and far more complex) kernel-based classifiers, such as soft-margin support vector machines (SVM) classifiers. Fortunately, however, the regularized PFM (R-PFM) method is conceptually simpler to understand and far easier to implement than its soft-margin SVM counterpart. For a given and known potential (kernel) function, R-PFM has far fewer tunable parameters than the soft-margin SVM algorithm. This aspect is important because, in general, the number of times the learning procedure has to be repeated increases with the number of tunable parameters in the machine learning algorithm. There is another major advantage that R-PFM has over the soft-margin SVM approach as a classifying tool: SVMs are designed for binary classifications, and their extension to multiclass problems is not straightforward, while R-PFM is very easily extensible to multiclass problems.

ACKNOWLEDGMENT

It is a pleasure to acknowledge Unilever for research support. We also would like to thank Dr. J. Townsend for useful discussions.

REFERENCES AND NOTES

(1) Nigsch, F.; Bender, A.; Jenkins, J. L.; Mitchell, J. Ligand-Target Prediction Using Winnow and Naive Bayesian Algorithms and the

Implications of Overall Performance Statistics. *J. Chem. Inf. Model.* **2008**, *48*, 2313–2325.
(2) Bender, A.; Mussa, H. Y.; Glen, R. C.; Reiling, S. Molecular similarity searching using atom environments, information-based feature selection, and a naive Bayesian classifier. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 170–178.
(3) Kotani, T.; Higashiura, K. Rapid evaluation of molecular shape similarity index using pairwise calculation of the nearest atomic distances. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 58–63.
(4) Rueda, M.; Bottegoni, G.; Abagyan, R. Consistent Improvement of Cross-Docking Results Using Binding Site Ensembles Generated with Elastic Network Normal Modes. *J. Chem. Inf. Comput. Sci.* **2009**, *49*, 716–725.
(5) Aizerman, M.; Braverman, E. M; Rozonoer, L. Theoretical Foundations of the Potential Function Method in Pattern Recognition. *Avtom. Telemekh.* **1964**, *25*, 917–936.
(6) Vapnik, V. N. *The Nature of Statistical Learning Theory*, 2nd ed.; Springer-Verlag: Berlin, Germany, 1995.
(7) Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.
(8) Mitchell, T. M. *Machine Learning*, 2nd ed.; McGraw-Hill International: New York, 1997.
(9) Evgeniou, T.; Pontil, M.; Poggio, T. Regularization Networks and Support Vector Machines. *Adv. Comp. Math.* **2000**, *13*, 1–50.
(10) Bender, A.; Jenkins, J. L.; Glick, M.; Deng, Z.; Nettles, H. J.; Davies, W. J. "Bayes Affinity Fingerprints" Improve Retrieval Rates in Virtual Screening and Define Orthogonal Bioactivity Space: When Are Multitarget Drugs a Feasible Concept. *J. Chem. Inf. Model.* **2006**, *46*, 2445–2456.
(11) Harper, G.; Bradshaw, J.; Gittins, J. C.; Green, D. V. S.; Leach, A. R. Prediction of biological activity for high-throughput screening using binary kernel discrimination. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1295–1300.
(12) Wilton, D. J.; Harrison, R. F.; Willett, P.; Dalaney, J.; Lawson, K.; Mullier, G. Virtual Screening Using Binary Kernel Discrimination: Analysis of Pesticide Data. *J. Chem. Inf. Model.* **2006**, *46*, 471–477.
(13) Schurmann, J. *Classification: A Unified View of Statistical and Neural Approaches*, 1st ed.; John Wiley and Sons: New York, 1996.
(14) Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, 1999.
(15) Bishop, C. M. *Pattern Recognition and Machine Learning*, 1st ed.; Springer-Verlag: New York, 2006.
(16) Tresp, V. Scaling kernel-based systems to large data sets. *Data Min. Knowl. Discov.* **2001**, *5*, 197–211.
(17) Aronszajn, N. Theory of reproducing kernels. *Trans. Amer. Math. Soc.* **1950**, *68*, 337–404.
(18) Wahba, G. *Spline models for observational data (CBMS-NSF Regional Conference Series in Applied Mathematics)*, 1st ed.; SIAM: Philadelphia, PA, 1990.
(19) Saitoh, S. *Theory of Reproducing Kernels and its Applications (Pitman research notes in mathematics series)*, 1st ed.; Longman Scientific and Technical: Harlow, Essex, U.K., 1988; Vol. 189.
(20) Abe, S. *Support vector machines for pattern classification*, 1st ed.; Springer-Verlag: New York, 2005.
(21) Young, T.; Calvert, T. W. *Classification, Estimation and Pattern Recognition*, 1st ed.; American Elsevier Pub. Co.: New York, 1974.
(22) Meisel, W. S. *Computer-Oriented Approaches to Pattern Recognition*, 1st ed.; Academic Press: New York, 1972.
(23) Freund, Y.; Schapire, R. E. Large margin classification using the perceptron algorithm. *Machine Learning* **1999**, *37*, 277–296.
(24) Braverman, E. M. On the potential function method. *Avtom. Telemekh.* **1965**, *26*, 2205–2213.
(25) Aizerman, M.; Braverman, E. M.; Rozonoer, L. The probability problem of pattern recognition learning and method of potential functions. *Avtom. Telemekh.* **1964**, *25*, 1307–1323.
(26) Braverman, E. M.; Pyatnitski, E. S. Estimation of the rate of convergence of algorithms based on the potential function method. *Avtom. Telemekh.* **1966**, *27*, 95–112.
(27) Fukunaga, K. *Introduction to Statistical Pattern Recognition*, 2nd ed.; Academic Press: London, U.K., 1990.
(28) Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*, 1st ed.; Cambridge University Press: Cambridge, U.K., 2004.
(29) Ikeda, K. *Convergence Theorem for Kemel Perceptron: In Computational Intelligence for the E-Age*; Proceedings of the 9th International Conference on Neural Information, Singapore, November 18−22, 2002; Wang, L, Rajapakse, J. C., Fukushima, K., Lee, S, Yao, X., Eds; Wiley-IEEE Press: Hoboken, NJ, 2002; Vol. 1, pp 163−166.
(30) Cortes, C.; Vapnik, V. N. Support vector networks. *Mach. Learn.* **1995**, *20*, 273–297.
(31) Hofmann, T.; Schölkopf, B.; Smola, A. J. Kernel methods in machine learning. *Ann. Stat.* **2008**, *36*, 1171–1220.
(32) Burges, C. J. *Simplified support vector decision rules*; Proceedings of the 13th International Conference on Machine Learning,Bari, Italy, July 3−6, 1996; Saitta, L., Ed.; Morgan Kaufmann: Burlington, MA, 1996; pp 71−77.
(33) Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167.
(34) Girosi, F. An equivalence between sparse approximation and support vector machines. *Neural Comput.* **1998**, *10*, 1455–1480.
(35) Smola, A. J.; Scholkopf, B.; Muller, K. R. The connection between regularization operators and support vector kernels. *Neural Networks* **1998**, *11*, 637–649.
(36) van Rijsbergen, C. J. *Information Retrieval*, 2nd ed.; Butterworths: London, U.K., 1979.
(37) Duda, R. O.; Hart, P. E. *Pattern Classification and Scene Analysis*, 1st ed.; Wiley: New York, 1973.
(38) Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin* **1945**, *1*, 80–83.
(39) Joachims, T.; Finley, T.; Yu, C. J. Cutting-Plane Training of Structural SVMs. *Mach. Learn.* **2009**, *77*, 27–59.
(40) Olah, M.; Mracec, M.; Ostopovici, L.; Rad, R.; Bora, A.; Hadaruga, N.; Olah, I.; Banda, M.; Simon, Z.; Oprea, T. I. WOMBAT: World of Molecular Bioactivity. In *Cheminformatics in Drug Discovery*; Oprea, T. I., Ed.; Wiley-VCH: Weinheim, Germany, 2004; pp 223−239.
(41) *Molecular Operating Environment (MOE)*; Chemical Computing Group: Montreal, Quebec, Canada; http://www.chemComp.com. Accessed January 10, 2010.
(42) Kearns, M. A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split. *Neural Comput.* **1997**, *9*, 1143–1161.
(43) Ripley, B. D. *Pattern Pattern Recognition and Neural Networks*, 1st ed.; Cambridge University Press: Cambridge, U.K., 1996; pp 66−89.
(44) Joachims, T. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*; Schölkopf, B., Burges, C., Smola, A. J., Eds.; MIT-Press: Cambridge, MA, 1999; pp 169−184.
(45) University of California Irvine Machine Learning Repository; University of California, Irvine: Irvine, CA; http://archive.ics.uci.edu/ml/. Accessed March 4, 2008.
(46) R Development Core Team . *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2008; http://www.R-project.org. Accessed April−July, 2010.

CI100022U