How To Winnow Actives from Inactives: Introducing Molecular Orthogonal Sparse Bigrams (MOSBs) and Multiclass Winnow

Florian Nigsch and John B. O. Mitchell*

Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom

Received September 19, 2007

In the present paper we combine the Winnow algorithm and an advanced scheme for feature generation into a tool for multiclass classification. The Winnow algorithm, specifically designed in the late 1980s to work well with high-dimensional data, by design ignores most of the irrelevant features for the scoring of each single training/test case. To augment the pool of available molecular features we use the Winnow algorithm in conjunction with a process that creates additional features from a set of given ones. We adapt a technique formerly employed in text classification termed "orthogonal sparse bigrams" and extend the use of that method to the domain of cheminformatics. Using circular molecular fingerprints as initial features, we create "molecular orthogonal sparse bigrams" (MOSBs) and report their successful application to the task of classification of bioactive molecules. Additionally, we introduce a memory-efficient way of bagging individual classifiers, avoiding the need to hold the complete training data set in memory. To compare the performance of our method with published results, we use the Hert data set of 8293 active molecules in 11 classes. We compare our method to Random Forest and find that our method not only is comparable or better in classification accuracy (up to 50% higher in MCC [Matthews correlation coefficient], 98% higher in fraction of correct predictions) but also is quicker to train (by a factor between 2 and 18, depending on the feature generation), more memory efficient, and able to cope more easily with large data sets when we seeded the actives into a pool of 94290 inactive molecules. It is shown that this method can be used with different fingerprints.

1. INTRODUCTION

In modern drug discovery virtual screening has become common practice, if not a necessity. There are two distinct types of virtual screening: structure-based and ligand-based screening. Structure-based virtual screening is possible if structural information about the target protein is available.[1] If the chemical structure(s) of one or more actives are known, but not necessarily the structure of the target protein, ligand-based virtual screening can be employed. In this paper we will focus on ligand-based virtual screening, and henceforth we refer to virtual screening in the sense of the ligand-based approach if not otherwise specified.[2]

Virtual screening is used to select those molecules out of a virtual library that are most likely to yield the same activity as a given set of example molecules of known activity.[3−5] As the size of virtual libraries can range from hundreds to millions of compounds, how to make such a selection is of paramount importance. Molecules most similar to the example molecules are most likely to have the same properties ("similarity principle").[6] This concept of similarity, however, is difficult to define in practice because molecular similarity is heavily context-dependent: any of shape, molecular weight, number of unsaturated carbon atoms, solubility—to name but a few—are valid criteria to define a similarity measure with. To satisfy the requirements of different situations a number of different algorithms have been (more or less) successfully applied to the problem of virtual screening.[7]

To process a molecule by an algorithm, it has to be represented in a computer-understandable format by means of molecular descriptors.[8,9] A vast number of these descriptors have been developed, the suitability and performance of which in virtual screening experiments have been extensively studied.[10,11] Our present work focuses on the use of representations of molecules which encode the presence of structural features. Primarily we used circular fingerprints, in particular the MOLPRINT 2D implementation.[12] To benchmark our method against a standard method with standard descriptors, we also used the MACCS key fingerprints (see below).[13]

Given a set of molecules with suitable molecular descriptors, an algorithm "learns" the specificities of the molecules labeled as either active or inactive. This training step is then followed by the classification of unlabeled test molecules. A whole range of classification algorithms can be used for this purpose, including support vector machine (as a representative of the wider class of kernel methods),[14] Random Forest (ensemble of classification trees),[15] artificial neural network,[16] (naïve) Bayesian classifier,[17] binary kernel discrimination,[18] and linear discriminant analysis.[19]

In the present paper we use a linear-threshold algorithm known as Winnow that was invented by Nick Littlestone in 1988.[20] Winnow has particularly good performance in settings where the total number of features is large compared to the few important features required to successfully classify

* Corresponding author phone: +44 (0)1223 762 983; fax: +44 (0)-1223 763 076; e-mail: jbom1@cam.ac.uk.

10.1021/ci700350n CCC: $40.75
Published on Web 01/26/2008

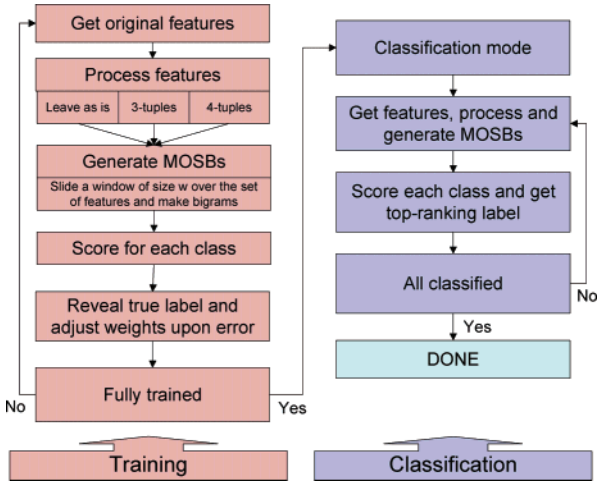

Figure 1. Schematic representation of the algorithm. The training step is shown on the left side, the classification step on the right side.

individual test cases, i.e., "when irrelevant attributes abound".[20] The Winnow algorithm is part of the class of "online" algorithms and, according to its inventor, has been "designed for efficiency in separating relevant from irrelevant attributes". Online algorithms do not need all data to be known from the outset, the input is processed piece by piece. Contrary to that, "offline" algorithms need all data to be accessible at all times, e.g., the random sampling procedure in Random Forest requires it to hold the whole data set in memory.[21] This fundamental difference has considerable implications on the tractability of problems where the underlying input data set gets large, i.e., the screening of large virtual libraries (>500 000 molecules) in combination with a considerable number of descriptors. Various modifications and extensions to the original Winnow algorithm have been developed, and the algorithm has been applied to tasks such as natural language processing (NLP), classification of text documents, and filtering of unsolicited commercial e-mail (spam).[22−25] To the best of our knowledge, we report the first application of this algorithm in cheminformatics.

Binary feature vectors for the description of molecules offer the possibility of mapping molecules into higher dimensional spaces by taking combinations of individual features. The resulting features may be more discriminating than the original ones, facilitating their classification through the use of suitable algorithms working in these augmented high-dimensional feature spaces. An obvious way to combine features would be their exhaustive enumeration. This approach, however, not only requires considerable computer time and memory but also is not necessarily the best solution. A more sophisticated way of generating additional features was proposed by Siefkes et al., yet in a completely different context.[23] They introduced a technique to create so-called "orthogonal sparse bigrams" (OSB) from the individual words of e-mails to build an effective filter against spam.

In this paper, we combine the Winnow algorithm with adapted feature generation to use molecular orthogonal sparse bigrams (MOSB) as features for the classification of bioactive substances (Figure 1). In the next section we will present the method, followed by a description of the data set and descriptors used. We then detail our results before moving on to a discussion of these and our conclusions.

2. METHODS

Algorithm. The Winnow algorithm holds an n-dimensional weight vector $w^c = (w_1^c, w_2^c, \ldots, w_N^c)$ for each class c, with w_i^c the weight of feature i in class c and N the size of F_{tot}, the set of all features in all classes. All w_i are initially set to 1. Every single training and test instance x is presented to the algorithm as a set of features $F_x = \{ f_i \}_{i=1}^{i=N_x}$, with $N_x \ll N$ the number of features present in that instance and $F_x \subset F_{tot}$. Each feature $f_i \in F_x$ is called an "active feature of instance x", and N_x is the number of active features of instance x. The score S_x^c for class c of instance x is then calculated as the sum of the weights of its active features

$$S_x^c = \sum_{j=1}^{N} \delta(f_j \in F_x) w_j^c$$

with $\delta(f_j \in F_x) = 1$ if $f_j \in F_x$ and 0 otherwise.

The learning procedure consists of a series of learning trials and is error-driven, i.e., the weight vectors are only updated if the predicted label for a given training instance is wrong.[20] A trial is made up of three steps: 1) algorithm receives instance; 2) prediction of label; and 3) algorithm receives true label ("reinforcement") and adjusts the weights accordingly. A prediction is considered a misclassification if the score is below the threshold for the true class or above the threshold for the wrong class(es). The threshold T_x for an instance x is the same as the number of active features N_x. In contrast to a fixed threshold, this has the advantage that the threshold varies as a function of the size of the molecule (i.e., number of structural features) and does not need to be optimized as in other methods, e.g., a naïve Bayesian classifier.

If $S_x^c < T_x$ for the correct class (false negative), then the weights of the active features in w^c are multiplied with a "promotion" factor $1 < p < p_{max}$. On the other hand, if $S_x^c > T_x$ for the wrong class (false positive), the weights of the active features in w^c are multiplied with a "demotion" factor $d_{min} < d < 1$. Instead of using fixed values for p and d, we calculate these values as a function of the misclassification difference $m_x = S_x^c - T_x$

$$p = 1 + \frac{p_{\max} - 1}{1 + e^{-|m_x|}}$$

$$d = 1 - \frac{1 - d_{\min}}{1 + e^{-|m_x|}}$$

with $p_{max} = 1.3$ and $d_{min} = 0.7$. For a false positive ($m_x > 0$) the weights of the active features are therefore lowered according to $w_j^{new} = d \cdot w_j$, whereas for a false negative ($m_x < 0$) the weights are increased according to $w_j^{new} = p \cdot w_j$.

We also use a threshold exclusion area ("thick threshold"): if the score of a correct prediction is within a certain ϵ area around the threshold, $|m_x| < \epsilon \cdot T_x$, it is equally considered as a mistake.[26] This means that the classifier not only learns upon misclassification but also if the classification was correct but too close to the threshold. The resulting classifier is more robust due to an increased separation between correct and incorrect predictions. The update rules of the algorithm are summarized in Table 1.

Table 1. Update Rules for the Winnow Algorithm with Thick Threshold[a]

score > threshold	true class	within ϵ	update
T	T	T	p
T	T	F	−
T	F	(T)	d
T	F	(F)	d
F	T	(T)	p
F	T	(F)	p
F	F	T	d
F	F	F	−

[a] The values in parentheses are irrelevant, because in these cases the other two binary determinants prevail. T is logical *true*, F is logical *false*.

In our implementation, the algorithm processes the whole training set once during training. If more than one scorer is used (see next paragraph), then each training example is used as many times as bagged scorers are used. When running in classification mode we consider the label of the class with the highest-ranking score the predicted label.

Bagging of Scorers. Instead of scoring each class only once, one may adopt a consensus approach by combining multiple independent scores into a final one, i.e., by using an approach similar to the "bagging" (bootstrap aggregation) concept proposed by Breiman in 1996 that is also used in Random Forest.[21,27] Consequently, we use multiple clones of our basic classifier, each one holding its own set of weight vectors, resulting in more than one score per class that can then be consolidated into the final prediction. Our implementation of this concept in the present context follows.

For each class a number of J scorers (i.e., weight vectors) is used, each of which is trained individually. The sequence in which the training instances are presented has to be different for each single scorer to result in different weight vectors. To that end, we keep a FIFO (first in−first out) cache of the descriptors of J training instances. The J training instances are randomly dispatched to the J scorers which update their weights accordingly. Subsequently, the oldest instance in the cache is removed, and the next training case is put into the cache. The scores of all scorers for a given class are aggregated by fusion through an arithmetic average. Alternatively, a majority vote could be used to get the most likely class, but this approach has not been pursued here.

Instead of taking bootstrap samples from the total amount of data, our implementation only draws from a cache holding as many samples as there are scorers and keeps on changing the examples in that cache while progressing through the training data. That procedure avoids the memory requirements to hold all data in memory while still allowing the use of multiple instances of scorers for increased accuracy and stability in predictions.

Molecular Orthogonal Sparse Bigrams (MOSB). As a starting point for the feature generation process we mainly used circular fingerprints (MOLPRINT 2D), though to a lesser extent we also employed MACCS key fingerprints for benchmarking purposes (see below). The potential and use of circular fingerprints in virtual screening experiments have been demonstrated in the recent literature.[28] A raw feature in MOLPRINT 2D format can be described as the following regular expression (regex): "[0-9]{1,2};([0-9]{1,2}-[0-9]-[0-9]{1,2};)+". This corresponds to the central atom *C* followed by one or more strings of the form *L-N-T*, encoding the number *N* of atoms of type *T* at a distance of *L* chemical bonds (maximum 2) from *C*. The algorithm iterates over every heavy atom of the molecule, encoding its atomic environment into a string of the format previously described. We chose to additionally enumerate from these raw features, all 3-tuples of the form *L-N-T* or, otherwise, for each central atom *C*, all 4-tuples of the form *C-L-N-T*. The resulting tuples/raw features are lexicographically ordered, duplicates discarded so as to yield ordered sets of unique molecular features. For each of these sets of basic molecular features, we then created molecular orthogonal sparse bigrams: a window of size w is moved over this ordered sequence of features, creating bigrams by taking two out of the w features, under the condition that the newest one is always present.[23] Example: Let T be the sequence of features t_1 to t_N. With a window size of $w = 5$, the first set of created bigrams would consist of (t_1, t_5), (t_2, t_5), (t_3, t_5), and (t_4, t_5). This process is illustrated for an example molecule using circular fingerprints (Figure 2).

The sets of bigrams obtained from raw features, 3-tuples, and 4-tuples will be referred to in the remainder of this document by the identifiers TR (for raw features), T3 (for 3-tuples), and T4 (for 4-tuples), respectively. The features resulting from the process described above are orthogonal to each other in the sense that they all span different axes in feature space. In other words, they are orthogonal in the sense that no MOSB feature can be obtained by combining any other arbitrarily chosen MOSB features; that is, every single MOSB feature is nonredundant. Their sum encodes more information than the original fingerprint, yet their computation demands less time and memory than an exhaustive enumeration of all combinations of features. For illustration, we will consider a molecule that can be described by m features. An exhaustive enumeration of all combinations would yield $M_{ex} = 1/2 \cdot m \cdot (m - 1)$ unique pairs for that molecule. In contrast, from any window of size w, exactly $w - 1$ bigrams can be obtained. As the number of possible windows of size w that can be formed from m unique elements is $m - w + 1$, it follows that from a total of m unique features a maximum of $M_w = (m - w + 1) \cdot (w - 1)$ unique bigrams can be formed for that same molecule. The ratio M_{ex}/M_w is then

$$\frac{M_{ex}}{M_w} = \frac{m \cdot (m-1)}{2 \cdot (m-w+1) \cdot (w-1)}$$

This ratio is independent of the size of the data set being analyzed, as the number of total molecules for which features are generated would appear on both sides of the fraction and therefore has no influence. An analysis of our data set showed that the average number m of unique MOLPRINT 2D features per molecule was $m \approx 25$. A window size of $w = 5$ then yields $M_{ex} = 300$ and $M_w = 84$ unique features per molecule on average, resulting in a ratio $M_{ex}/M_w \approx 3.6$. This more than 3-fold decrease in features generated shows that the computational overhead is less if the exhaustive enumeration is avoided, and we also found that an exhaustive enumeration does not lead to more accurate classification results (see below).

Figures of Merit. To assess the accuracy of classification and hence the performance of our method we calculated the

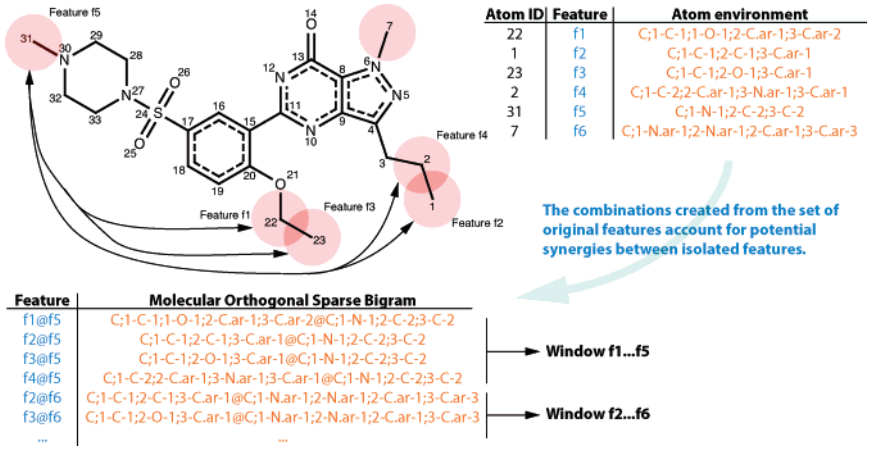

Figure 2. The process of creation of molecular orthogonal sparse bigrams (MOSBs). A window of a specified size w (here $w = 5$) is slid over the sequence of original features. By taking combinations of a fixed element with all others contained in this window another set of nonredundant features is created.

Table 2. Definition of True/False Positives/Negatives t_p^c, t_n^c, f_p^c, and f_n^c for Each Class from the c by c Confusion Matrix $Z = (z_{ij})$

figure	calculation
total number of instances classified	$N_{tot} = \Sigma_i \Sigma_j z_{ij}$
instances with true class c	$N_c = \Sigma_j z_{cj}$
instances with predicted class c	$N'_c = \Sigma_i z_{ic}$
true positives in class c	$t_p^c = z_{cc}$
false negatives in class c	$f_n^c = N_c - t_p^c$
false positives in class c	$f_p^c = N'_c - t_p^c$
true negatives in class c	$t_n^c = N_{tot} - (t_p^c + f_n^c + f_p^c)$

following figures of merit. The Matthews correlation coefficient (MCC) has been calculated according to

$$\mathrm{MCC} = \frac{t_p t_n - f_p f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}}$$

where t_p, t_n, f_p, and f_n stand for true/false positives/negatives, respectively. MCCs have been calculated for individual classes as well as across all classes by taking the sum of true/false positives/negatives across all classes. We also calculated the average percentage class correct (APCC) that is related to the total recall of positives $R_p = t_p/(t_p + f_n)$ and the total recall of negatives $R_n = t_n/(t_n + f_p)$ in the following way:

$$\mathrm{APCC} = \frac{1}{2}(R_p + R_n)$$

Numerical values for APCC are reported as real numbers between 0 and 1. For a binary classification, t_p, t_n, f_p, and f_n are readily obtained from a 2 by 2 confusion matrix. In multiclass classification with $c > 2$ classes, however, the confusion matrix is a c by c matrix $Z = (z_{ij})$, z_{ij} representing the number of times an instance of true class i is predicted to belong to class j. In this case, we calculated t_p, t_n, f_p, and f_n as shown in Table 2. This is of paramount importance for the rest of this paper as we are only considering multiclass classification here. With these definitions, it is possible to calculate all of the figures of merit previously mentioned for any given single class of a multiclass classification problem. However, problems arise when characterizing the accuracy of the method across all classes. It can easily be shown that, according to the definitions in Table 2, the sum of false positives $F_p = \Sigma_c f_p^c$ equals the sum of false negatives $F_n = \Sigma_c f_n^c$ (column- and row-wise marginals minus diagonal elements, respectively). Thus, the MCC defined as above is not an unbiased measure for the accuracy of a method for multiclass classification. We therefore employ another measure that we will call fraction of correct predictions (FCP)

$$\mathrm{FCP} = tr(Z)/N_{tot}$$

with $tr(Z)$ representing the trace of the confusion matrix. The FCP accounts for the fraction of all correct predictions (diagonal elements of the confusion matrix) with respect to the total number of predictions made.[29] The complement of the FCP is equivalent to the error rate ER=1-FCP.

A report of the FCP alone would hide much of the information contained in the whole confusion matrix; therefore, we report for any single experiment at least the overall MCC, overall APCC (both calculated from the sums of per-class true/false positives/negatives); FCP; per-class MCCs along with recall and precision of positives and numbers of true/false positives/negatives. Similarly, as the MCC (as well as the FCP) can appear misleadingly high if a substantial number of inactives are correctly predicted, we also report figures of merit for actives alone for experiments including a considerable number of inactives. For comparison of one particular set of parameters with another across all classes, we normally use the FCP, and sometimes the total number of false positives, as well. For some figures we employ ΔFP, referring to the difference in false positives with respect to the lowest value achieved.

Practical. Our implementation of the algorithm described above was done in C++. Unless stated otherwise, all calculations have been done on an iMac G5 (1.9 GHz) with 1 GB of physical memory. Data analysis, figure plotting, and Random Forest experiments (randomForest package version 4.5−18) have been performed with the freely available statistical framework R.[30,31]

3. DATA SETS AND DESCRIPTORS

We used a data set of 8293 compounds in 11 activity classes that has first been used by Hert et al. (Table 3).[10]

Table 3. Hert Data Set, Consisting of 8293 Actives in 11 Classes[a]

MDDR activity code	activity class	instances
06233	5HT3 antagonists	752
06235	5HT1A agonists	827
06245	5HT reuptake inhibitors	359
07701	D2 antagonists	395
31420	renin inhibitors	1130
31432	angiotensin II AT1 antagonists	943
37110	thrombin inhibitors	803
42731	substance P antagonists	1246
71523	HIV protease inhibitors	750
78331	cyclooxygenase inhibitors	636
78374	protein kinase C inhibitors	452
inact	not the above	94 290
total	−	102 583

[a] We augmented the data set with 94 290 inactives to yield a total of 102 583 molecules.

Additionally, a collection of 94 290 molecules not listed as active against any of the targets of the previously mentioned 8293 molecules has been used, yielding a data set totalling 102 583 molecules employed in a recent publication.[17] All molecules have been converted to Sybyl MOL2 format with the open source software OpenBabel 2.0.2 using the option to remove hydrogens.[32−34] We converted all molecules to MOLPRINT 2D fingerprints with the PERL scripts from http://www.molprint.com/, and MACCS key fingerprints have been calculated with the Molecular Operating Environment (MOE).[13,35]

4. RESULTS

Influence of Parameters. There are three parameters that can be varied: 1) size of the sliding window (w); 2) number of scorers per class (J); and 3) the separation parameter ϵ. To shed some light on the influence of these parameters on the performance of the classifier, we used the data set of only actives on which we carried out some extensive calculations. We randomly divided the 8293 molecules into 5 different training and test sets, using 80% of every single class for training and the remaining 20% for testing, corresponding to a 5-fold Monte Carlo cross-validation. By doing so we ensure that no single class is under- or overrepresented in either set and avoid skewed per-class results. Calculations have been carried out for every single split, and the results presented are averages of these 5 independent runs.

Features. We first investigated to what extent the set of features used influences the accuracy of classification. The influence of the other parameters being unclear at this point, we set $\epsilon = 0$ and used one scorer only ($J = 1$) with window size set to $w = 1$. Consistent with the expectation that T4 features would outperform T3 features, we find an FCP of 0.8077 (MCC = 0.7885) for 4-tuples compared to an FCP of 0.5769 (MCC = 0.5206) in the case of 3-tuples. The omission of the central atom therefore results in a relative decrease in classification accuracy of 40.0% in FCP (51.5% in MCC). The TR features outperform both 4-tuples as well as 3-tuples, yielding an average FCP of 0.9348 (MCC = 0.9283) which presents a relative increase of 15.7% in FCP relative to 4-tuples and 62.0% with respect to 3-tuples (Table 4).

Window Size. We varied the size of the window sliding over the set of features from 1 to 10, with inclusion of all individual features, i.e., monograms (Figure 3, Table S1). A larger window size results in more features that the algorithm is aware of, and as such better classification results are expected. Raw features at any window size perform better (in terms of FCP) than any other combination of features and window sizes (Figure 3a). For 3- and 4-tuples the classification accuracy increases up to a window of size $w \approx 6$ but levels off thereafter. For all sets of features, there is a window size that yields a higher FCP than exhaustive feature generation. Hence, we conclude that an exhaustive enumeration of features does not lead to the highest classification accuracy. This can be important when screening large databases where the enumeration of all possible features may become prohibitive because of space and time requirements. Whereas the FCP varies only slightly with window size, there is a pronounced difference in the number of false positives, especially so for T3 and T4 features (Figure 3b). The window sizes resulting in the highest FCP (lowest number of false positives) are 3, 7, and 10 for raw features (TR), 3-tuples (T3), and 4-tuples (T4), respectively.

Table 4. Classification of 5 Times 1663 Randomly Selected Molecules out of 8293 Actives[a]

tuples	MCC	APCC	FCP
T3	0.5206	0.5642	0.5769
T4	0.7885	0.8077	0.8077
TR	0.9283	0.9348	0.9348

[a] With a window size of $w = 1$ and different features; $\epsilon = 0$ and $J = 1$.

Number of Scorers. We varied the number of scorers J per class from 1 to 10 for a fixed window size of $w = 1$ and find that bagging of the classifier results in improved performance, notably for T3 and T4 features (Figure 4a and S1). In another experiment, we varied J from 1 to 5 for the optimum window size found earlier (Figures 4b and S2, Table S2). The change in classification accuracy is more pronounced in this case than for $w = 1$. For 5 scorers ($J = 5$), the optimum window size performs 21.4% (T3), 8.1% (T4), and 0.2% (TR) better in terms of FCP than $w = 1$.

Separation Parameter Epsilon ϵ. The parameter ϵ controls the separation between correct and incorrect classifications. For fixed window size w we varied ϵ systematically in steps of 0.025 from 0 to 0.4 (Figures 5 and S3, Table S3). For all sets of features the plot of ΔFP vs ϵ is qualitatively convex and has a minimum. The values of ϵ resulting in the lowest number of false positives (and also highest FCP) are 0.175, 0.1, and 0.1 for TR, T3, and T4 features, respectively (Table S3).

Classification Results for Actives Only and Total Data Set. The analysis of the influence of the parameters leads us to the conclusion that, within the parameter space that we examined, the best combination of parameters is as follows: $w = 3$ and $\epsilon = 0.175$ for TR features, $w = 7$ and $\epsilon = 0.1$ for T3 features, and $w = 10$ and $\epsilon = 0.1$ for T4 features (Table S4). We used these settings to classify 10 random splits (80% training, 20% test set) of both actives alone and actives seeded into the set of inactives (Table 5). In the case of actives alone the results are summarized in Figure 6a, for the total data set see Figure 6b. Detailed results can be found in Tables S5, S6, and S7 for actives alone and in Tables S8, S9, and S10 for the total data set.

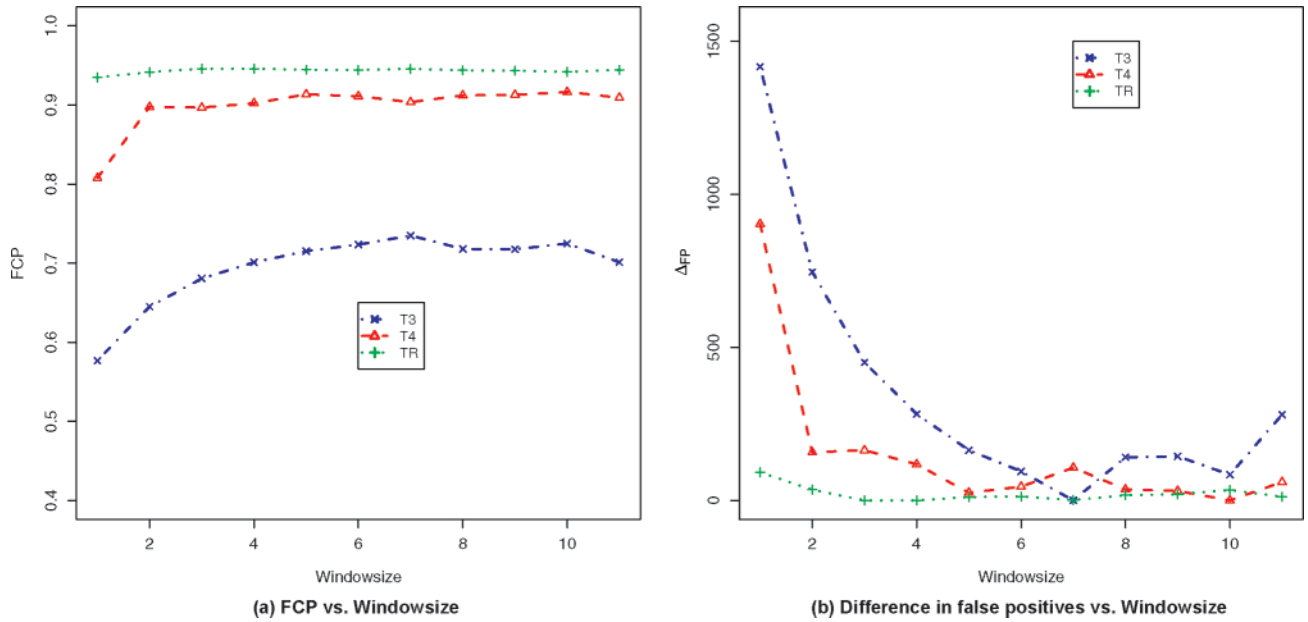

Figure 3. Influence of the size *w* of the sliding window on classification accuracy of the test set. The last point corresponding to an apparent window size of 11 is for exhaustive feature generation. (a) Variation of FCP with window size. (b) Relative increase in false positives with respect to the window size that yields the fewest of them, which is 3 for TR, 7 for T3, and 10 for T4 features.

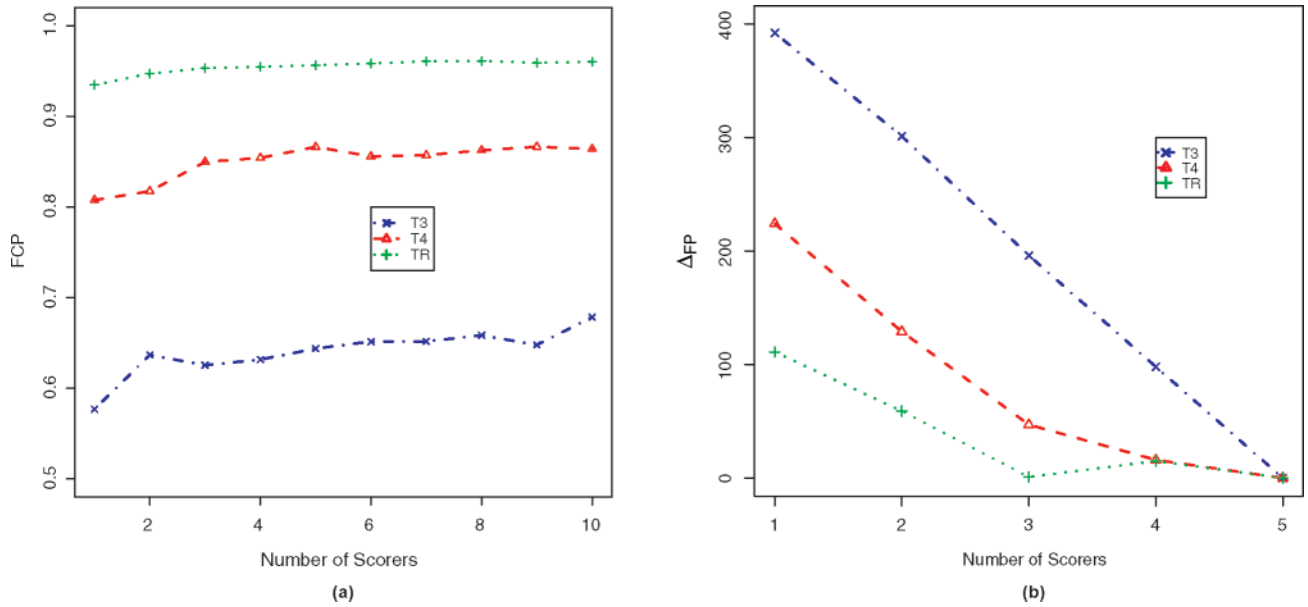

Figure 4. Influence of the number of scorers *J* on classification accuracy of the test set. (a) FCP as a function of *J*, window size is fixed to 1. (b) Relative increase in false positives with respect to the number of scorers that yields the fewest of them, which is 5 in all cases. Window size is 3 for TR features, 7 for T3 features, and 10 for T4 features.

In all experiments TR features consistently outperform the other two sets of features, followed by T4 and T3 features in that order. When only classifying 8293 active molecules, the FCPs (overall MCCs) are 0.9623 (0.9585), 0.9387 (0.9326), and 0.7833 (0.7617) for TR, T4, and T3 features, respectively. If the inactives are included in the classification the observed FCPs (overall MCCs) are 0.9899 (0.9890), 0.9839 (0.9824), and 0.9636 (0.9603) for TR, T4, and T3 features, respectively. The exclusion of the 94 290 inactives from the calculation of these figures of merit yields the following FCPs (overall MCCs): 0.8887 (0.9234), 0.8829 (0.8736), and 0.5592 (0.6601) for TR, T4, and T3 features, respectively. Activity classes most difficult to predict were found to be 06235 (5HT1A agonists), 06245 (5HT reuptake inhibitors), and 07701 (cyclooxygenase inhibitors), an observation in agreement with earlier studies.[10,17]

Comparison with Random Forest. To compare our methodology with a method that is commonly employed for classification of molecules we used Random Forest as a classifier in combination with MACCS key fingerprints.[13,21] The combination of these two arises from the fact that Random Forest is a well-documented and established technique (not only) in cheminformatics.[36,37] Similarly, MACCS key fingerprints are industry-standard chemical fingerprints readily obtained from molecular modeling packages such as MOE.[35] We could not use the MOLPRINT 2D fingerprints for these purposes because the data files would have become prohibitively large if our data set would have been converted

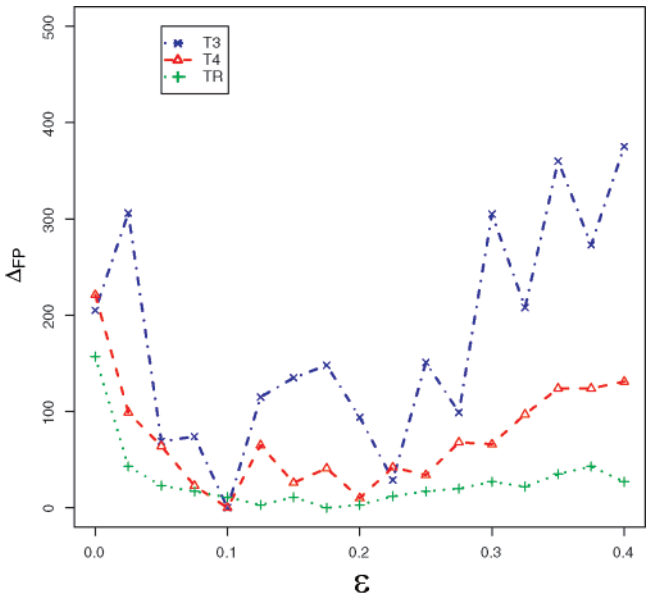

Figure 5. Influence of the separation parameter ϵ on classification accuracy of the test set. Relative increase in false positives (ΔFP) with respect to the value of ϵ that yields the fewest of them, which is 0.175 for TR features, and 0.1 for both T3 and T4 features; window size is 3 for TR features, 7 for T3 features, and 10 for T4 features.

Table 5. Consolidated Results for 10 Random 80%:20% (Training:Test) Splits of Our Data Set[a]

features	ϵ	w	J	MCC	APCC	FCP
Actives Only						
T3	0.100	7	5	0.7617	0.8808	0.7833
T4	0.100	10	5	0.9326	0.9663	0.9387
TR	0.175	3	5	0.9585	0.9792	0.9623
Total Data Set						
T3	0.100	7	1	0.9603	0.9801	0.9636
T4	0.100	10	1	0.9824	0.9912	0.9839
TR	0.175	3	1	0.9890	0.9945	0.9899
Only Actives within Total Data Set						
T3	0.100	7	1	0.6601	0.7790	0.5592
T4	0.100	10	1	0.8736	0.9112	0.8229
TR	0.175	3	1	0.9234	0.9442	0.8887

[a] "Actives Only" refers to 10 times 1663 molecules out of 8293 actives, "Total Data Set" is 10 times 20 519 out of 102 583 molecules. Results for the total data set are included when the "inactive" class is omitted for the calculation of the figures of merit ("Only Actives within Total Data Set").

into binary feature vectors, the subsequent handling of which in R would have been very difficult to impossible.[31]

For use with the randomForest() routine in R, the MACCS key fingerprints have been converted into binary vectors.[30] In the case of Winnow we used the raw ASCII files as exported from MOE and ran our method with window sizes of $w = 1$ and $w = 5$, and $J = 5$ in both cases. The results are summarized in Figure 7 and Table 7 for actives alone and in Figure 8 and Table 8 for the total data set. Table 9 shows a direct comparison between Random Forest and Winnow. For actives only randomForest() was called with its default parameters (variables tried at each split equal to the square root of the number of variables, and number of trees equal to 500). In the case of the total data set with ≈102 000 molecules we used 90 trees, as in a preliminary run further trees did not result in an increase of the classification accuracy of the forest rather than in a decrease (Figure S5).

In the case of only actives, the MCC is always lowest for Winnow with $w = 1$, whereas Winnow with $w = 5$ outperforms Random Forest for 8 out of 11 classes. This superiority of Winnow is mainly due to a considerably higher recall for classes where Random Forest only retrieves around 20% of actives, whereas differences in precision are less pronounced. For the MDDR activity classes 06245 (5HT reuptake inhibitors) and 07701 (cyclooxygenase inhibitors) the relative increase in recall is 340% and 159%, respectively. When inactives are included in the classification, Winnow with $w = 5$ outperforms Random Forest in 4 out of 12 classes in terms of MCC, with approximately equal performance in 4 other classes. In this setting, a window size of $w = 1$ yields a very poor classifier compared to both others. In terms of recall, Winnow with $w = 5$ achieves higher values in 6 out of 12 classes, which is only the case for 3 classes when considering precision.

An important difference between the two methods is requirements in terms of computer time. Training of the Winnow algorithm is quicker in all cases that we examined, being up to 18 times quicker to train when only actives are used with a window size of $w = 1$. When comparing the two almost equivalent situations in terms of accuracy (Random Forest and Winnow with $w = 5$ using MACCS keys), Winnow is 7.5 times quicker to train. The total time for both training and subsequent classification steps is 47 s compared to 308 s for Random Forest, i.e., Winnow is ≈6.6 times faster. Comparing the same two experiments using the total data set, Winnow is almost two times faster. The best classifier using TR features is ≈4 times faster in both cases, considering the sum for both steps.

5. DISCUSSION

Parameters for Winnow. The performance of our method is influenced by the following factors: 1) type of features; 2) size of the sliding window w; 3) separation parameter ϵ; 4) number of scorers J. The experiments we have carried out on the set of 8293 actives allow for several comments to be made about the individual and collective impact of these parameters on the performance of our method.

Clearly, the type of features used has a major impact on the classification accuracy. In all of the experiments, the ranking of features according to their FCP (as well as total number of false positives) is in order of increasing accuracy: T3 < T4 < TR. These results agree with the expectation that a given method performs better when provided with input data of higher quality. The information content of the features provided also increases in the order T3 < T4 < TR, which we will discuss briefly. Every T3 feature consists of a string *L-N-T*, describing the number *N* of atoms of type *T* that can be found at a distance of *L* bonds from an—in the case of T3 features—unspecified atom. Contrary to that, a T4 feature includes the central atom in the 4-tuple of the form *C-L-N-T*. This leads to the major difference, that for a given molecule there will always be fewer unique T3 features than there will be T4 features. Unless a molecule is highly symmetric or has many chemically equivalent structural groups, the number of T4 features of a molecule will always be of the same order as the total

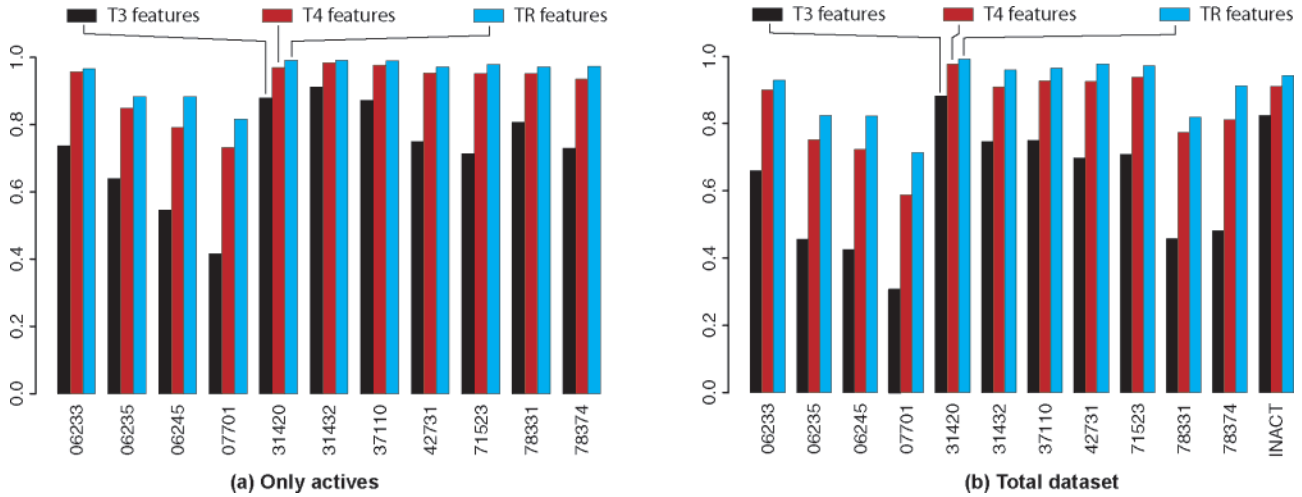

Figure 6. Classification of 10 random splits where 80% of each class is used for training and 20% for testing. Averaged test set MCCs for all classes using the three different sets of features and optimum parameters. Numbers below the bars correspond to the class numbers in Tables S5−S7. (a) Only 8293 actives. (b) Total data set of 102 583 molecules.

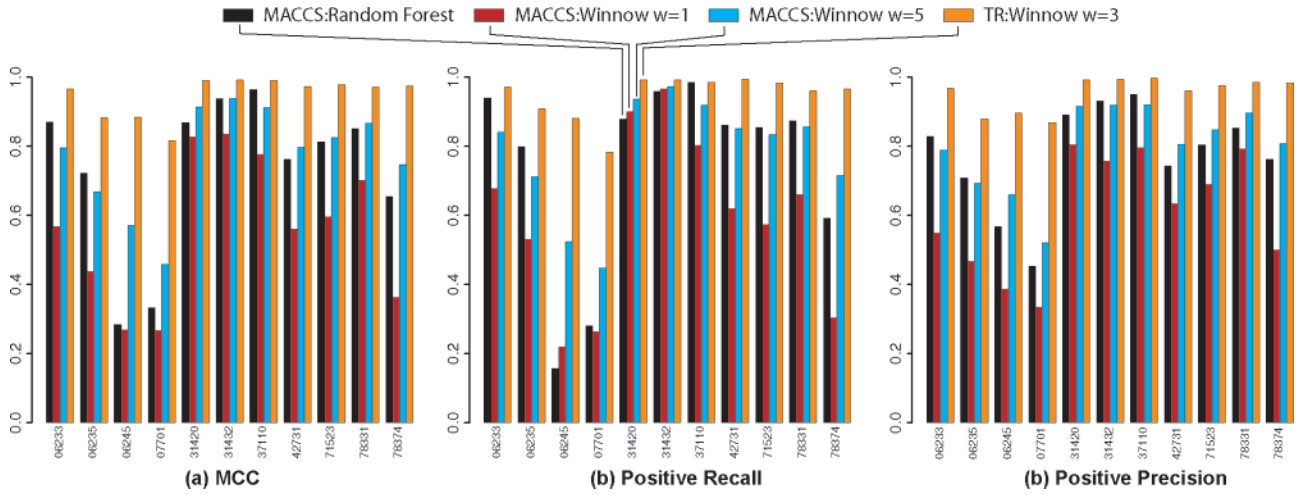

Figure 7. Comparison of Random Forest and Winnow for 10 random 80%:20% (training:test) splits of the 8293 actives alone. Figures reported are averages of the 10 test sets. (a) MCC; (b) positive recall; (c) positive precision.

Table 6. Comparison between Random Forest and Winnow Using MACCS Key Fingerprints and Circular Fingerprints[a]

method	features	MCC	APCC	FCP	T_{tr} [sec]	T_{te} [sec]
			Actives Only			
Random Forest	MACCS	0.7971	0.8985	0.8155	306	2
Winnow ($w = 1$)	MACCS	0.6242	0.8121	0.6583	17 (18x)	2 (1x)
Winnow ($w = 5$)	MACCS	0.8094	0.9047	0.8267	41 (7.5x)	6 (0.3x)
Winnow ($w = 3$)	TR	0.9585	0.9792	0.9623	69 (4.4x)	4 (0.5x)
			Total Data Set			
Random Forest	MACCS	0.9482	0.9741	0.9525	1070	–
Winnow ($w = 1$)	MACCS	0.9207	0.9604	0.9273	193 (5x)	20
Winnow ($w = 5$)	MACCS	0.9454	0.9727	0.9499	495 (1.9x)	73
Winnow ($w = 3$)	TR	0.9890	0.9945	0.9899	196 (4.6x)	35
			Only Actives within Total Data Set			
Random Forest	MACCS	0.6212	0.7234	0.4474		
Winnow ($w = 1$)	MACCS	0.3024	0.5669	0.1342		
Winnow ($w = 5$)	MACCS	0.5925	0.7110	0.4225		
Winnow ($w = 3$)	TR	0.9234	0.9442	0.8887		

[a] Results are for a 10-fold random split (80%:20% training:test) of the total data set including typical times for the training (T_{tr}) and classification (T_{te}) steps; values in parentheses are the conversion factors when going from Winnow to Random Forest. For the total data set the conversion factors refer to total time incurred in both training and testing steps.

number of heavy atoms of that molecule. Contrasting with that, the number of T3 features will be smaller because the same number of neighboring atoms is a condition that is more easily met if the central atom is not specified. The problem is similar to considering, for example, how many red cars are parked next to a blue car (equivalent to T4 features) compared with how many red cars are parked next to any car (equivalent to T3 features). In the former case where

Table 7. Classification of 10 Times 1663 Randomly Selected Molecules out of 8293 Total Actives (80%:20% Training:Test), Using MACCS Key Fingerprints and Different Classification Algorithms[a]

	Random Forest			Winnow, $w = 1$			Winnow, $w = 5$		
class	R_p	P_p	MCC	R_p	P_p	MCC	R_p	P_p	MCC
06233	**0.9404**	**0.8285**	**0.8703**	0.6767	0.5492	0.5684	0.8411	0.7879	0.7958
06235	**0.7994**	**0.7081**	**0.7232**	0.5313	0.4667	0.4379	0.7114	0.6931	0.6685
06245	0.1569	0.5678	0.2836	0.2197	0.3863	0.2677	**0.5243**	**0.6591**	**0.5714**
07701	0.2810	0.4531	0.3322	0.2627	0.3350	0.2657	**0.4467**	**0.5199**	**0.4581**
31420	0.8796	0.8919	0.8679	0.9004	0.8047	0.8263	**0.9372**	**0.9157**	**0.9146**
31432	0.9587	**0.9316**	0.9379	0.9661	0.7576	0.8352	**0.9730**	0.9195	**0.9387**
37110	**0.9857**	**0.9497**	**0.9640**	0.8017	0.7958	0.7770	0.9198	0.9210	0.9118
42731	**0.8612**	0.7434	0.7621	0.6193	0.6340	0.5611	0.8519	**0.8064**	**0.7974**
71523	**0.8553**	0.8049	0.8123	0.5724	0.6897	0.5951	0.8346	**0.8481**	**0.8257**
78331	**0.8742**	0.8522	0.8516	0.6599	0.7925	0.7024	0.8569	**0.8969**	**0.8666**
78374	0.5912	0.7631	0.6554	0.3033	0.4991	0.3622	**0.7154**	**0.8077**	**0.7471**
ALL	0.8155	0.8155	0.7971	0.6583	0.6583	0.6242	**0.8267**	**0.8267**	**0.8094**

[a] The highest values are highlighted in bold. R_p positive recall, P_p positive precision, MCC Matthews correlation coefficient.

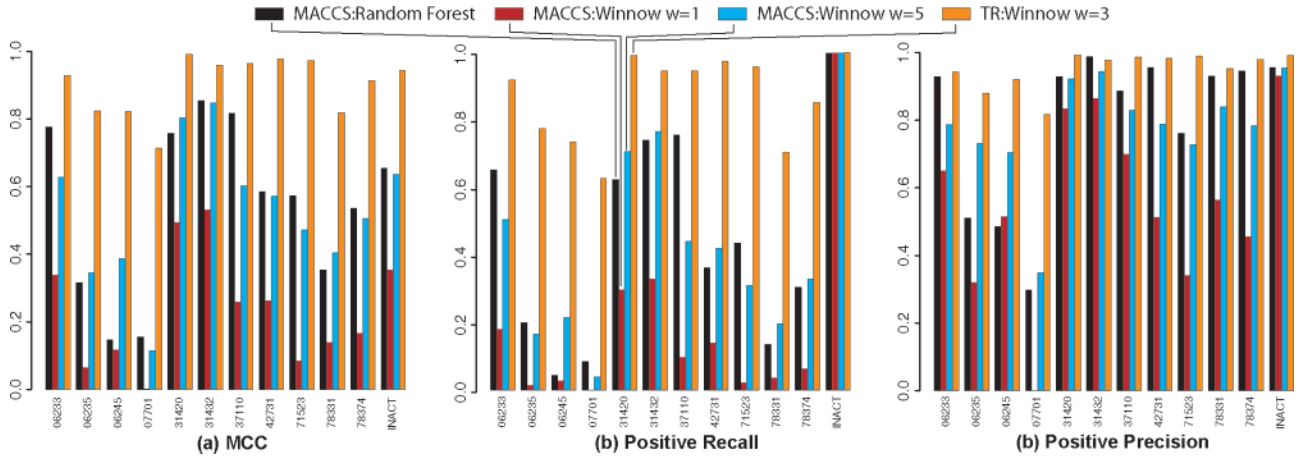

Figure 8. Comparison of Random Forest and Winnow for 10 random 80%:20% (training:test) splits of the total 102 583 molecules. Figures reported are averages of the 10 test sets. (a) MCC; (b) positive recall; (c) positive precision.

Table 8. Classification of 10 Times 20 519 Randomly Selected Molecules out of the Total Data Set (80%:20% Training:Test), Using MACCS Key Fingerprints and Different Classification Algorithms[a]

	Random Forest			Winnow, $w = 1$			Winnow, $w = 5$		
class	R_p	P_p	MCC	R_p	P_p	MCC	R_p	P_p	MCC
06233	**0.6517**	**0.9292**	**0.7768**	0.1783	0.6498	0.3381	0.5036	0.7867	0.6274
06235	**0.2000**	0.5108	0.3163	0.0133	0.3188	0.0636	0.1652	**0.7307**	**0.3453**
06245	0.0444	0.4848	0.1460	0.0264	0.5135	0.1158	**0.2139**	**0.7032**	**0.3868**
07701	**0.0835**	0.2973	**0.1559**	0.0000	0.0000	0.0016	0.0381	**0.3488**	0.1141
31420	0.6226	**0.9287**	0.7583	0.2956	0.8350	0.4939	**0.7049**	0.9224	**0.8045**
31432	0.7392	**0.9887**	**0.8538**	0.3280	0.8647	0.5303	**0.7646**	0.9438	0.8483
37110	**0.7547**	**0.8869**	**0.8168**	0.0969	0.6996	0.2586	0.4398	0.8300	0.6021
42731	0.3624	**0.9557**	**0.5859**	0.1376	0.5134	0.2615	**0.4196**	0.7893	0.5720
71523	**0.4340**	**0.7623**	**0.5730**	0.0220	0.3402	0.0850	0.3093	0.7284	0.4724
78331	0.1352	0.9301	0.3534	0.0344	0.5641	0.1382	**0.1961**	0.8395	**0.4043**
78374	0.3044	**0.9454**	**0.5355**	0.0615	0.4553	0.1662	**0.3275**	0.7842	0.5055
Inact	0.9971	**0.9562**	**0.6540**	**0.9973**	0.9311	0.3526	0.9965	0.9548	0.6356
Act	0.4474	0.8675	0.6212	0.1342	0.6903	0.3024	0.4225	0.8362	0.5925
ALL	0.9525	0.9525	0.9482	0.9273	0.9273	0.9207	0.9499	0.9499	0.9454

[a] The highest values are highlighted in bold. R_p positive recall, P_p positive precision, MCC Matthews correlation coefficient.

the condition is more strict, each red car parked next to a car of any other color is considered a different arrangement, whereas in the latter case with the less strict condition, the arrangement of a red car next to any other car is considered the same arrangement. Consequently, the number of T3 features is always less than that of T4 features. As for the raw features, TR, they are yet more specific and thus contain more information than any of the other types of features. To once again resort to the example of a parked car, the equivalent of TR features encodes very specifically for every single car how many cars of which specific color are parked in its immediate neighborhood of up to two cars away. Unless a specific pattern is enforced, the likelihood of having the same arrangement for a considerable number of cars is very low. As a consequence of the foregoing reasons, the information content of the features follows the series T3 <

Table 9. Classification of 10 Times 20 519 Randomly Selected Molecules out of the Total Data Set (80%:20% Training:Test), Using MACCS Key Fingerprints for Random Forest and TR Features for Winnow[a]

	Random Forest, MACCS			Winnow, TR		
class	R_p	P_p	MCC	R_p	P_p	MCC
06233	0.6517	0.9292	0.7768	0.9179	0.9422	0.9295
06235	0.2000	0.5108	0.3163	0.7735	0.8807	0.8240
06245	0.0444	0.4848	0.1460	0.7347	0.9200	0.8216
07701	0.0835	0.2973	0.1559	0.6253	0.8179	0.7142
31420	0.6226	0.9287	0.7583	0.9903	0.9933	0.9917
31432	0.7391	0.9887	0.8538	0.9439	0.9775	0.9602
37110	0.7547	0.8869	0.8168	0.9441	0.9870	0.9651
42731	0.3624	0.9557	0.5859	0.9736	0.9830	0.9780
71523	0.4340	0.7623	0.5730	0.9553	0.9903	0.9725
78331	0.1352	0.9301	0.3534	0.7047	0.9525	0.8183
78374	0.3044	0.9454	0.5355	0.8516	0.9798	0.9131
Inact	0.9971	0.9562	0.6540	0.9988	0.9923	0.9440
Act	0.4474	0.8675	0.6212	0.8887	0.9605	0.9233
ALL	0.9525	0.9525	0.9482	0.9899	0.9899	0.9890

[a] R_p positive recall, P_p positive precision, MCC Matthews correlation coefficient. All but one number are higher for Winnow+TR.

T4 < TR, as does the number of unique features of a given type. Our results confirm the theoretical expectation that for all measures of overall performance (MCC, APCC, FCP), we find in order of increasing performance T3 < T4 < TR (Table 4).

Given this series, it is understood that the *relative increase* in information due to inclusion of combinations of individual features should follow the opposite series TR < T4 < T3. In other words, a combination of two T3 features will contain more information with respect to the two individual features than a combination of TR features will with respect to two individual TR features. Hence, the additional features created by means of the sliding window process described earlier will have more influence on the classification performance according to the series TR < T4 < T3. There is a pronounced increase in classification performance for T3 features when the window size w goes from 1 (FCP=0.5769) to 2 (FCP=0.6451), corresponding to a relative increase in FCP of 11.8% (Figure 3a). Further increase of the sliding window results in only a slight increase which levels off at $w \approx 7$. In the case of T4 features, the step from $w = 1$ to $w = 2$ entails a change in FCP from 0.8077 to 0.8976 (corresponding to a relative increase of 11.1%), changes thereafter are less important than for T3 features and level off earlier at $w \approx 5$. Even less pronounced is the effect on TR features changing from FCP 0.9348 to FCP 0.9417 (less than 1% relative change) when w increases from 1 to 2. Compared to the optimum window size ($w_{opt}^{TR} = 3$, $w_{opt}^{T3} = 7$, $w_{opt}^{T4} = 10$) the increase in FCP is 27.4%, 13.5%, and 1.2% for T3, T4, and TR features, respectively.

In summary, we find that the order of features in increasing classification performance is T3 < T4 < TR, whereas the effect of the additional combinations of individual features is opposed to that series. The values of w that yield the best results are 3, 7, and 10 for TR, T3, and T4 features, respectively.

Ensembles of predictors, or in other words the consolidation of multiple predictions into a single one, has proven useful with other methods.[27] In the case of Random Forest, the outcome of many individual decision trees is aggregated into one.[21,30] Similarly, ensembles of artificial neural networks have been implemented in the same manner.[38,39] The general principle is that more than one independent prediction is obtained for any given test case, allowing the algorithm to make a consensus choice. This can readily be implemented by taking an average (arithmetic or weighted by some distance information obtained at the same time) of the individual outcomes. In the case of classification, another possibility is to take a majority vote of all the predictors. As laid out in the description of the algorithm, we chose to take an arithmetic average of the scores of individual predictors.

Our results for a varying number of scorers J, with window size w fixed at 1, confirm that in general two or more scorers per class perform better than an individual one (Figures 4 and S1). T3 features gain most through multiple scorers, followed by T4 and TR features, in that order. This confirms once again our hypothesis of information content and relative capabilities of improvement through other means, such as combination of features and/or bagging. An investigation with $w \neq 1$, namely $w = 3$, $w = 7$, and $w = 10$ for TR, T3, and T4 features, respectively, leads to the same conclusions but provides the additional information that the effect of more scorers is more pronounced if a feature-optimized window size is used (Figures 4b and S2). By switching from $J = 1$ to $J = 5$, the total number of false positives drops by ≈100, 200, and 400 for TR, T3, and T4 features, respectively, in a classification of 5 times 1663 randomly chosen molecules.

The parameter ϵ controls the separation between correct and incorrect classifications. This is equivalent to forcing the classifier to make more mistakes during training, thereby forcing the mistake-driven learning procedure to update the weight vectors more frequently. The weight vectors are updated not only if the score is above the threshold for the wrong class, or below the threshold for the right class, but also when a correct prediction is too close to the threshold. The results we obtain by varying ϵ from 0 to 0.4 in steps of 0.025 at fixed w show that there is an optimum value for ϵ in all cases (Figures 5 and S3, Table S3). In the case of T3 features, the FCP increases with ϵ up to a value of ≈0.1 after which there is a decline in classification accuracy; the same behavior is reproduced in a smoother fashion for T4 features. An interesting fact to note here that ties in with the discussion of the types of different features earlier on is the stability of FCP with respect to changes in ϵ. TR features seem to be most resilient to changes in ϵ, whereas the other features, in particular T3 features, undergo important changes upon variation of ϵ. The relative change in false positives with respect to the minimum value shows that there is an optimum value for ϵ which is close to 0.1 in all cases. The optimum value for ϵ is 0.1 for T3 and T4 features and 0.175 for TR features, and changes to either side of this value result in an increase in the total number of false positives for all features (Figure 5).

The nature of the influence on the classification performance can once more be explained by the properties of the different features used. We have shown earlier on that the number of unique features follows the series T3 < T4 < TR. Furthermore, we make the assumption that molecules of a given class are more likely to share a common subset of features than any number of class-wise unrelated molecules. This assumption can be seen as an interpretation of the "molecular similarity principle" in words that suit the purpose of the present discussion. Accordingly, for a given

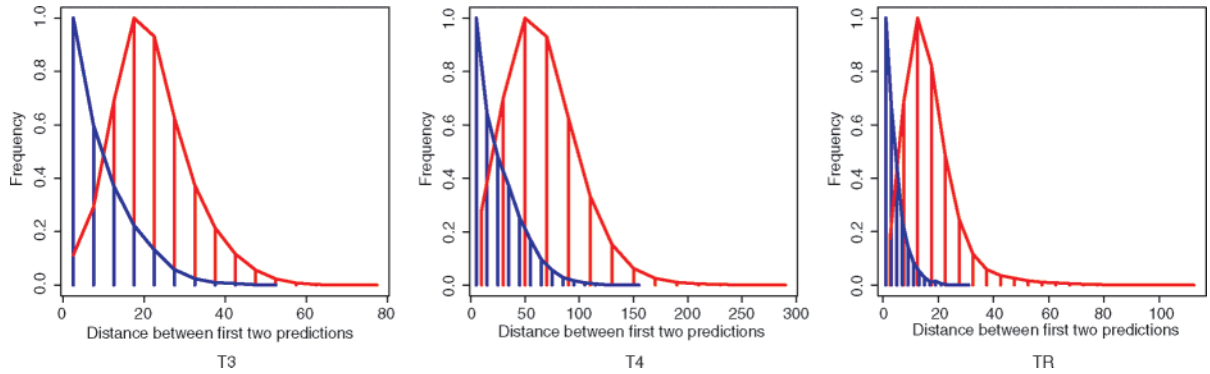

Figure 9. Analysis of test set predictions: separation between the two most likely predictions for different features. The Gaussian-shaped curves (red) are for correct predictions, single-tailed curves (blue) are for incorrect predictions; both are normalized to unit height. The separation is always better for correct predictions, and resolution R_S for the peaks follows the series T3 (0.44) < T4 (0.48) < TR (0.67).

activity class, the number of features available to distinguish it from the other classes varies in the order T3 < T4 < TR. Furthermore, the probability of a significant overlap of features used to describe the members of different classes is higher the smaller the number of total features available. The problem of attributing class-specific features is hence complicated, as is the inverse problem of determining class membership of a molecule given its set of features. The algorithm being forced to more frequently update its weights if $\epsilon > 0$ is now more likely to erroneously update the weights of the "wrong" features if the total pool of features per class is diminished. In the case of fewer total features, any single one of them is prone to more updates in either direction (promotion or demotion). These supposedly wrong adjustments to the weights result in the observed jumps in classification accuracy.

We performed an analysis of the separation between correct and incorrect predictions for the results obtained from a 10-fold random split of the whole data set (≈102 000 molecules, resulting in ≈204 000 predictions) with default parameters of $w = 1$, $\epsilon = 0.1$, and $J = 1$ (Figure 9). This figure shows a histogram (normalized to unit height) of the difference between the scores of the two most likely predictions, d_{12}, for different types of features. The curves with tails only extending to the right side correspond to incorrectly classified test cases (first-ranking class is not true class), whereas the Gaussian-shaped curves correspond to the test cases that have been correctly classified (first-ranking class is true class). To note from these curves are the following characteristics: 1) on average, d_{12} is larger for correct predictions and smaller for incorrect ones, which corresponds to a good separation of correct/incorrect predictions; 2) the further the tail of the blue curve extends to the right into the red curve, the lower is the "resolution" of the predictions. Accordingly, the greater the overlap of the two curves, the lower the accuracy of that method is expected to be. The resolution between two peaks can be computed using a commonly employed definition of resolution R_S between two peaks A and B

$$R_S = \frac{2\Delta Z}{W_A + W_B}$$

with ΔZ being the distance of the maximum values of the peaks, and $W_{A,B}$ the width at the base of peaks A and B. By estimation from the graph one finds in order of increasing resolution R_S the by now expected series: T3 (0.44) < T4 (0.48) < TR (0.67) (Table 10).

Table 10. Resolution R_S between Correct and Incorrect Predictions for Different Features in a Classification of 10 Times 20 519 Randomly Selected Molecules out of the Total Data Set (80%:20% Training:Test) with Values Estimated from Figure 9

features	ΔZ	W_A	W_B	R_S
T3	20	30	60	0.44
T4	60	75	175	0.48
TR	20	10	50	0.67

All of the aforementioned points confirm one of the main characteristics of the Winnow algorithm: the algorithm has been proven in theory and in practice to perform well in situations where there is an abundant number of features, but very few of them are important in each case.[20] This important characteristic is at the heart of the different performances that we see in our experiments. As the number of features follows the series T3 < T4 < TR, the classification accuracy equally does so, too. The scenario is the same for an increasing number of features with increasing size of the window (w) being slid over the set of molecular features: the larger w, the larger the number of features which in turn results in better performance in cases where individual features alone are either not enough in number (especially T3 features) and/or not discriminating enough with respect to the classes a molecule belongs to. Yet, the classification accuracy does not increase *ad infinitum* with more features, as shown by the inferior performance in all cases when an exhaustive feature enumeration is used (Figure 3b).

Comparison to Random Forest. We wanted the benchmarking of our method against Random Forest to be as unbiased as possible. Doing that, we hit a number of obstacles, some of which are inherent to the methods being compared, and others are implementation-dependent problems which should not be interpreted as drawbacks of the given method.

The main difficulty encountered when comparing our method to Random Forest was the inherent difference that the latter is an algorithm requiring all data to be held in memory at all times. In contrast, our implementation of the bagging Winnow algorithm only needs as many training examples as there are independent scorers J, which we incorporated using a FIFO cache (see Methods). Using MACCS key fingerprints and the total data set, the data matrix has dimensions 102 583 × 167, resulting from 166

Table 11. Memory Requirements for the Winnow Algorithm When the Total Data Set of 102 583 Molecules Is Used, Illustrating the Scalability of the Method[a]

features	window size	no. of weights	memory per class per scorer
MACCS	1	≈160	640 bytes
MACCS	5	≈5500	21.5 kB
T3	7	≈11 000	43.0 kB
T4	10	≈360 000	1.4 MB
TR	3	≈1 900 000	7.2 MB

[a] The number of weights can be adjusted through the fingerprint employed and the size of the sliding window. The reported numbers of weights are obtained from the algorithm after completion of the training step.

binary MACCS features plus one variable for the class label. The raw data in text format is 37.9 MB, read into R this increases to 129.5 MB (obtained via object.size() in R). Considering the ≈5500 MOSBs from MACCS keys that are created when using a window size of $w = 5$ this corresponds to a 33-fold (i.e., 5500/166) increase in data (Table 11). The resulting files would reach sizes of 1.25 GB in raw text which are too large to use with randomForest in R. The use of the inherently much more numerous MOLPRINT 2D features results in a prohibitive increase in data for "offline" methods such as Random Forest that are required to hold all data in memory. Thus, we could not use those latter features with Random Forest.

As opposed to Random Forest which has to keep all data (additionally to the structures of the trees) in memory for its bootstrap sampling procedure, our method only needs to hold as many training examples in a FIFO cache (several MBs) as there are scorers J in use. Furthermore, a weight vector for each class in every scorer has to be accounted for. Memory requirements for this vary with the number of features that are created, ranging from several kilobytes to several megabytes, and can straightforwardly be adjusted according to specific needs (Table 11). What puts the bagging Winnow algorithm in an inherently favorable position for the use with MOSBs is that the online learning allows the data to be created on-the-fly, i.e., there is no need to hold all MOSBs of all examples in memory at all times. The MOSBs of any single training or test example are created from its original fingerprint when it is first seen and discarded when needed no further.

When using MACCS key fingerprints and 8293 actives only, Random Forest (FCP=0.8155) performs better than Winnow with $w = 1$ (FCP=0.6583) and equally well with respect to Winnow with $w = 5$ (FCP=0.8267). For classes where Winnow with $w = 5$ has a higher MCC than Random Forest, this can be ascribed to a significantly higher recall for Winnow (Figure 7b). Classification accuracy for Random Forest (FCP=0.9525) is slighty higher with respect to Winnow with $w = 5$ (FCP=0.9499) when the total data set is used.

The use of raw circular fingerprints (TR features) changes performance of the Winnow algorithm considerably, resulting in higher values for all figures of merit in all classes. In terms of FCP for the total data set, Winnow with TR features achieves a value of 0.9899 compared to 0.9525 for Random Forest (≈4% increase). If, however, the bias introduced through the large number of (correctly classified) inactives is accounted for, the relative increase in FCP is as big as 98.6% (0.8887 for Winnow, 0.4474 for Random Forest). Relative increases in positive recall range from 1555% for the MDDR activity class 06245 (5HT reuptake inhibitors) to 25% for class 37110 (thrombin inhibitors). Equally, there are significant increases observed in precision, in particular 75% for class 07701 (D2 antagonists) and ≈90% for class 06245 (5HT reuptake inhibitors).

The training procedure allows the algorithm to "learn" the important features of the activity classes provided. The resulting weights of each feature allow an identification of those features the algorithm learns as the most important ones for a given activity class. If this is combined with fingerprints that can easily be related to molecular structure, then it should be possible to use this information to help in drug discovery programs. Some preliminary work in our laboratory suggests such a possibility (manuscript in preparation).

6. CONCLUSIONS

The algorithmic framework we developed, multiclass Winnow in combination with an advanced feature generation scheme to create molecular orthogonal sparse bigrams (MOSBs) from molecular fingerprints, was successfully applied to the classification of bioactive substances. In a similar way as in Random Forest, we implemented the bagging (bootstrap aggregation) of multiple individual classifiers, resulting in a more accurate and robust predictor. Our implementation, however, does not need to hold all training data in memory due to a different sampling procedure, which reduces the number of examples to be held in memory to the number of different instances of the classifier being used (usually up to 10). This memory-efficient cloning of individual scorers in combination with an on-the-fly generation of MOSBs ties in naturally with the design of the online learning procedure of the algorithm and allows for large data sets to be handled with relative ease.

A comparison of our method with Random Forest using the Hert data set and MACCS key fingerprints shows that Winnow is an equal or better classifier, depending on the number of MOSBs used. The combination of Winnow with circular fingerprints excels in all experiments that we performed; additionally, the Winnow algorithm was quicker to train than Random Forest for all combinations of data sets and descriptors that we examined.

ACKNOWLEDGMENT

The authors thank Unilever for funding. F.N. thanks Dr. Hamse Mussa for many helpful discussions.

Supporting Information Available: Additional tables and figures referenced in the text. This material is available free of charge via the Internet at http://pubs.acs.org/.

REFERENCES AND NOTES

(1) Verdonk, M. L.; Berdini, V.; Hartshorn, M. J.; Mooij, W. T. M.; Murray, C. W.; Taylor, R. D.; Watson, P. Virtual screening using protein-ligand docking: avoiding artificial enrichment. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 793−806.

(2) Hert, J.; Willett, P.; Wilton, D. J.; Acklin, P.; Azzaoui, K.; Jacoby, E.; Schuffenhauer, A. New methods for ligand-based virtual screening: use of data fusion and machine learning to enhance the effectiveness of similarity searching. *J. Chem. Inf. Model.* **2006**, *46*, 462−470.

(3) Kubinyi, H. Molecular similarity. 1. Chemical structure and biological action. *Pharm. Unserer Zeit* **1998**, *27*, 92−106.
(4) Kubinyi, H. Molecular similarity. 2. The structural basis of drug design. *Pharm. Unserer Zeit* **1998**, *27*, 158−172.
(5) Johnson, M.; Lajiness, M.; Maggiora, G. Molecular similarity: a basis for designing drug screening programs. *Prog. Clin. Biol. Res.* **1989**, *291*, 167−171.
(6) Willett, P. Similarity-based approaches to virtual screening. *Biochem. Soc. Trans.* **2003**, *31*, 603−606.
(7) Bruce, C. L.; Melville, J. L.; Pickett, S. D.; Hirst, J. D. Contemporary QSAR classifiers compared. *J. Chem. Inf. Model.* **2007**, *47*, 219−227.
(8) Raevsky, O. A. Physicochemical descriptors in property-based drug design. *Mini Rev. Med. Chem.* **2004**, *4*, 1041−1052.
(9) Senese, C. L.; Duca, J.; Pan, D.; Hopfinger, A. J.; Tseng, Y. J. 4D-fingerprints, universal QSAR and QSPR descriptors. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1526−1539.
(10) Hert, J.; Willett, P.; Wilton, D. J.; Acklin, P.; Azzaoui, K.; Jacoby, E.; Schuffenhauer, A. Comparison of topological descriptors for similarity-based virtual screening using multiple bioactive reference structures. *Org. Biomol. Chem.* **2004**, *2*, 3256−3266.
(11) Bender, A.; Glen, R. C. A discussion of measures of enrichment in virtual screening: comparing the information content of descriptors with increasing levels of sophistication. *J. Chem. Inf. Model.* **2005**, *45*, 1369−1375.
(12) 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. Comput. Sci.* **2004**, *44*, 170−178.
(13) Elsevier MDL, 2440 Camino Ramon, Suite 300, San Ramon, CA 94583, U.S.A.
(14) Mahé, P.; Ralaivola, L.; Stoven, V.; Vert, J. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model.* **2006**, *46*, 2003−2014.
(15) Cannon, E. O.; Bender, A.; Palmer, D. S.; Mitchell, J. B. O. Chemoinformatics-based classification of prohibited substances employed for doping in sport. *J. Chem. Inf. Model.* **2006**, *46*, 2369−2380.
(16) Molnar, L.; Keseru, G. M. A neural network based virtual screening of cytochrome P450 3A4 inhibitors. *Bioorg. Med. Chem. Lett.* **2002**, *12*, 419−421.
(17) Cannon, E. O.; Amini, A.; Bender, A.; Sternberg, M. J. E.; Muggleton, S. H.; Glen, R. C.; Mitchell, J. B. O. Support vector inductive logic programming outperforms the naive Bayes classifier and inductive logic programming for the classification of bioactive chemical compounds. *J. Comput.-Aided. Mol. Des.* **2007**, *21*, 269−280.
(18) Wilton, D. J.; Harrison, R. F.; Willett, P.; Delaney, J.; Lawson, K.; Mullier, G. Virtual screening using binary kernel discrimination: analysis of pesticide data. *J. Chem. Inf. Model.* **2006**, *46*, 471−477.
(19) Arodz, T.; Yuen, D. A.; Dudek, A. Z. Ensemble of linear models for predicting drug properties. *J. Chem. Inf. Model.* **2006**, *46*, 416−423.
(20) Littlestone, N. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning* **1988**, *2*, 285−318.
(21) Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, 5−32.
(22) Mesterharm, C. A Multi-class Linear Learning Algorithm Related to Winnow. In *Adv. Neural Inf. Processing Syst.* **2000**, *12*, 519−525.
(23) Siefkes, C.; Assis, F.; Chhabra, S.; Yerazunis, W. S. Combining Winnow and Orthogonal Sparse Bigrams for Incremental Spam Filtering. In *Proceedings of the European Conference on Principle and Practice of Knowledge Discovery in Databases*; 2004; pp 410−421.
(24) Zhang, T. Regularized Winnow Methods. In *Advances in Neural Information Processing Systems*; 2001; pp 703−709.
(25) Zhang, T.; Damerau, F.; Johnson, D. Text Chunking based on a Generalization of Winnow. *J. Mach. Learn. Res.* **2002**, *2*, 615−637.
(26) Dagan, I.; Karov, Y.; Roth, D. Mistake-driven learning in text-categorization. In *Empirical Methods in Natural Language Processing*; 1997; pp 55−63.
(27) Breiman, L. Bagging Predictors. *Machine Learning* **1996**, *24*, 123−140.
(28) Glen, R. C.; Bender, A.; Arnby, C. H.; Carlsson, L.; Boyer, S.; Smith, J. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME. *IDrugs* **2006**, *9*, 199−204.
(29) Baldi, P.; Brunak, S.; Chauvin, Y.; Andersen, C. A.; Nielsen, H. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* **2000**, *16*, 412−424.
(30) Liaw, A.; Wiener, M. Classification and Regression by randomForest. *R News* **2002**, *2*, 18−22.
(31) R Development Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2007.
(32) Tripos, 1699 South Hanley Road, St. Louis, MO 63144-2319, U.S.A.
(33) Guha, R.; Howard, M. T.; Hutchison, G. R.; Murray-Rust, P.; Rzepa, H.; Steinbeck, C.; Wegner, J.; Willighagen, E. L. The Blue Obelisk-interoperability in chemical informatics. *J. Chem. Inf. Model.* **2006**, *46*, 991−998.
(34) *The Open Babel Package, version 2.0.2*. http://openbabel.sourceforge.net/ (accessed 02/08/2007).
(35) Chemical Computing Group, Suite 910 - 1010 Sherbrooke St. W Montreal, Quebec, Canada H3A 2R7. http://www.chemComp.com (accessed May 17, 2007).
(36) Ehrman, T. M.; Barlow, D. J.; Hylands, P. J. Virtual screening of Chinese herbs with Random Forest. *J. Chem. Inf. Model.* **2007**, *47*, 264−278.
(37) Zhang, Q.; Aires-de-Sousa, J. Random forest prediction of mutagenicity from empirical physicochemical descriptors. *J. Chem. Inf. Model.* **2007**, *47*, 1−8.
(38) Agrafiotis, D. K.; Cedeño, W.; Lobanov, V. S. On the use of neural network ensembles in QSAR and QSPR. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 903−911.
(39) Bakker, B.; Heskes, T. Clustering ensembles of neural network models. *Neural Netw.* **2003**, *16*, 261−269.

CI700350N