Novel Approaches for Small Biomolecule Classification and Structural Similarity Search

Emre Karakoc
SFU Lab for Computational Biology
8888 University Drive
Burnaby, Canada
ekarakoc@cs.sfu.ca

Artem Cherkasov
UBC Division of Infectious Diseases
2733 Heather Street
Vancouver, Canada
artc@interchange.ubc.ca

S. Cenk Sahinalp [*]
SFU Lab for Computational Biology
8888 University Drive
Burnaby, Canada
cenk@cs.sfu.ca

ABSTRACT

Structural similarity search among small molecules is a standard tool used in molecular classification and in-silico drug discovery. The effectiveness of this general approach depends on how well the following problems are addressed. The notion of similarity should be chosen for providing the highest level of discrimination of compounds with respect to the bioactivity of interest. The data structure for performing search should be very efficient as the molecular databases of interest include several millions of compounds.

In this paper we summarize the recent applications of k-nearest-neighbor search method for small molecule classification. The k-nn classification of small molecules is based on selecting the most relevant set of chemical descriptors which are then compared under standard Minkowski distance L_p. Here we describe how to computationally design the optimal *weighted* Minkowski distance wL_p for maximizing the discrimination between active and inactive compounds wrt bioactivities of interest. k-nn classification requires fast similarity search for predicting bioactivity of a new molecule. We then focus on construction of pruning based k-nn search data structures for any wL_p distance that minimizes similarity search time.

The accuracy achieved by k-nn classifier is better than the alternative LDA and MLR approaches and is comparable to the ANN methods. In terms of running time, k-nn classifier is considerably faster than the ANN approach especially when large data sets are used. Furthermore, k-nn classifier is capable of quantification of the level of bioactivity rather than returning a binary decision and can bring more insight to the nature of the activity via eliminating unrelated descriptors of the compounds with respect to the activity in question.

1. INTRODUCTION

Small chemical molecules (with molecular weights ≤ 500) are very important to the exploration of molecular and cellular functions. They also play key roles in treating diseases: almost all medicines available today are small molecules. One of the fundamental research challenges we are facing today is the identification of small molecules that play an active role in regulation of a given biological process or disease state.

Recent developments in the chemistry have given researchers the ability to efficiently synthesize and screen large number of related small molecules, a capability previously available only to pharmaceutical companies. As a result of these developments, public small chemical compound databases started to emerge with exponentially increasing number of compounds. The Molecular Libraries Roadmap of NIH, PubChem, is one of the largest public databases which aims to offer access to the large number of chemical compounds and their functions. Currently PubChem contains 100,000 compounds with known bioactivities and a total of 10 million unique small chemical compounds. This initiative is expected to lead new data mining and machine learning techniques to reveal the relationship between the structural information of chemical compounds and their bioactivity. It is anticipated that these projects will also facilitate the development of new drugs by providing early stage chemical compounds to validate new drug targets which could be then move into drug-development pipeline.

Structural similarity search among small molecules is one of the standard tools used in conventional *in silico* drug discovery as structural similarity usually implies similarity in physicochemical properties and/or biological activities [14]. Thus, it is common to query small molecules databases with a *probe* compound possessing desirable biological activity (bioactivity) to *discover* chemically similar database entries. It is also common to perform classification of a compound with an unknown bioactivity level through a similarity search among compounds whose bioactivity levels are known. Clustering small molecules with similar bioactivities can lead to better understanding of the underlying chemical and structural properties related to the bioactivity. It is known that certain structural components of small molecules can target other components via establishing stable chemical bonds. Identification of common characteristics of a given biological function significantly reduces the number of compounds need to be tested for a particular drug target.

This important *ligand-based* drug discovery methodology and classification approach are associated with the following two fundamental computational problems. (1) The notion of similarity used in search determines the molecules that are extracted from the database. A notion of similarity which has the highest level of bioactivity discrimination is

[*] Corresponding author

very desirable and needs to be determined computationally. (2) It is desirable to have efficient algorithms for structural and/or chemical similarity search as the molecular databases of interest include several millions of compounds and linear/brute force search may take significant amount of time (several days in certain large private databases).

In this summary, we describe frequently used computational methods for addressing these two fundamental research problems. The first part of the paper focuses on the common similarity/distance measures used in small chemical compound databases and their application for classification of compounds according to a given bioactivity. Typically small chemical compounds are represented as a vector of descriptors, that are extracted from the 2D/3D structure of chemical compounds. The most common measures of similarity/distance amongst sets of molecular descriptors are *Tanimoto coefficient*, standard L_1 and L_2 distances. These similarity/distance measures are global and independent from the bioactivity of the compounds. In order to capture the similarity between compounds with better accuracy with respect to a particular bioactivity more sophisticated measures must be used.

Similarity/distance measure itself is not enough for classification of new compounds; it must be trained in order to capture the bioactivity of interest. The most commonly used classification techniques of chemical compounds include Multiple Linear Regression (MLR) [6], Linear Discriminant Analysis (LDA) [12] and machine-learning techniques such as Support Vector Machines (SVM) [19] and Artificial Neural Networks(ANN) [21]. In this summary we focus on the k-nearest neighbor (k-nn) classification, which deduces the level of the bioactivity of a query molecule based on the number (and the bioactivity levels) of active elements among its k nearest neighbors with respect to a distance measure of choice. In order to determine the best similarity/distance measure, we introduce use of the (more general) weighted Minkowski distance of order 1, namely wL_1. For each bioactivity of interest, we determine real valued weights w_i of the wL_1 distance so as to maximize the discrimination between active and inactive compounds in a training set. We compute the *optimal* values for weights w_i via a linear optimization procedure [10].

The second part of our summary focuses on the second problem, efficient data structures for fast chemical similarity search which is essential for k-nn classification. Space Covering Vantage Point (SCVP) tree is one of the well known data structures for performing fast nearest neighbor search in dimensional metric spaces via triangle inequality. In the SCVP tree construction, the vantage points in each level are chosen randomly until all search space is covered [15]. Clearly, it is desirable to minimize the number of vantage points that cover the search space as a better space utilization can be achieved, implying that more levels of the tree can be fitted in the available memory. In this summary we show how to approximate the minimum number of vantage points and thus obtain the optimum allocation through a simple polynomial time algorithm. The resulting data structure, which we call the deterministic multiple vantage point tree (DMVP tree) [10], when built in full, is guaranteed to have $O(\log \ell)$ levels, where ℓ is the size of the data set. If the maximum number of children of an internal node at level i is c_i, the query time guaranteed by our data structure is $O(\sum_{i=1}^{\log \ell} c_i)$. Because c_i is typically a small constant (approximately minimized for each level), the query time is only $O(\log \ell)$, a significant improvement over linear/brute force search.

Due to redundant representation of data items, the memory usage of the DMVP tree can be super-polynomial. In case the DMVP tree requires more memory than available, lower levels of the DMVP trees could be cut out. In this case, the pruning in the leaf nodes can be achieved by linear search. We also show how to obtain the optimum cut so as to minimize the expected query performance.

Our data structure is not only interesting for classification purposes; similarity search among small molecules under various notions of similarity is of independent interest. To the best of our knowledge, this is the first application of an efficient similarity search data structure to small molecule data collections.

2. COMMONLY USED SIMILARITY MEASURES FOR SMALL MOLECULES.

Given a notion of similarity among data points, it is usually possible to obtain a corresponding *distance* measure; searching for structurally most similar molecules to a query molecule in this context corresponds to searching for molecules with the smallest distance to the query molecule. The key premise of this approach is that the notion of a distance is mathematically well defined and algorithms for handling distance based classification, clustering and search are better understood. Under a given distance measure, the search for the most similar molecule to a query compound becomes the Nearest Neighbor Search (NN) problem. Thus, the above two problems in structural similarity search, i.e. classification and querying, can be mapped to corresponding problems in nearest neighbor search.

There are various ways to define the descriptors/parameters for the chemical structures stored in electronic collections conventionally used in the modern computer-aided drug discovery [2; 1].

Such parameters either (1) merely reflect the structural organization of molecules in qualitative manner, such as those used in the popular *structural fingerprints* (employed in NCBI's PubChem database), e.g. the existence of a doubly bonded Carbon pair, a three membered ring, an aromatic atom etc. [13] or (2) reflect various local and global physical-chemical molecular features (chemical descriptors) which are quantitative, such as atomic weight, aromaticity, hydrophobicity, the number of specific atoms, charge, density, etc. These descriptors serve as independent variables for modern *QSAR* (Quantitative Structure-Activity Relationship) tools including the structural similarity search engines in chemical compound databases.

Given an adequate set of descriptors, it is desirable to have a measure of similarity or alternatively a distance measure under which functionally related molecules have a high level of similarity or small distance, and non-related compounds have a low level of similarity or large distance. The most common measure of similarity amongst sets of molecular descriptors is the so called *Tanimoto coefficient* [17]. Given two descriptor sets (which can be organized in arrays) X and Y, the Tanimoto coefficient is defined to be the ratio of the number of descriptors that are identical in X and Y and the total number of descriptors available for X and Y. The Tanimoto coefficient is in the range $[0, 1]$; a value close to 1 implies similarity and a value close to 0 implies a dissimilarity among the two descriptor sets compared.

Often a collection of descriptors are represented as a bit-vector (e.g. structural fingerprints) where each one of the n possible descriptors is assigned a *dimension*, i.e. natural number between 1 and n (this is the representation used by PubChem and other databases). Let $B(x)$ represent the bit-vector corresponding to a molecule x and let $B(x)[i]$ represent its i^{th} dimension. Given two compounds x and y, the Tanimoto coefficient $T(x, y)$ is then defined as $T(x, y) = (\sum_{i=1}^{n}(B(x)[i] \wedge B(y)[i]))\ /\ (\sum_{i=1}^{n}(B(x)[i] \vee B(y)[i]))$. Although the Tanimoto coefficient provides a measure of *similarity*, it is possible to define a *Tanimoto distance measure* as $D_T(x, y) = 1 - T(x, y)$.

The Tanimoto coefficient is very popular mostly due to its simplicity. For real valued descriptor arrays (where each dimension has a real value) it is also quite common to use the Minkowski distance of order p, denoted L_p for measuring their similarity. Given two real valued n dimensional descriptor arrays X and Y, their Minkowski distance of order p, namely L_p, is defined as $L_p(X, Y) = (\sum_{i=1}^{n} |X[i] - Y[i]|^p)^{1/p}$. When comparing two structural fingerprints $B(x)$ and $B(y)$, the Minkowski distance of order 1 is equivalent to the well known Hamming distance (see for example [3]): $H(B(x), B(y)) = \sum_{i=1}^{n} |B(x)[i] - B(y)[i]|$.

In order to capture the similarity between compounds with respect to a particular bioactivity, it is possible to assign a relative importance to each structural descriptor in the form of a weight $w_i \in [0, 1]$. The resulting *weighted* Minkowski distance of order 1 can then be defined for two descriptor arrays X and Y as $wL_1(X, Y) = \sum_{i=1}^{n} w_i \cdot |X[i] - Y[i]|$.[1]

3. CLASSIFICATION METHODS FOR SMALL MOLECULES.

The descriptor arrays described above can be used for classification of compounds according to a given bioactivity.

One of the most popular classification techniques is the MLR (Multiple Linear Regression) [6] method which quantifies the activity level of a descriptor array X as: $Activity(X) = c + \sum_{i=1}^{n} \sigma_i \cdot X[i]$ where c is a constant. If $Activity(X) \geq t$ for a (user specified) threshold value t then it is likely that the molecule is active with respect to the bioactivity of interest. Notice that the MLR classifier is described by a planar separator in the multi-dimensional descriptor array space; those points on one side of the separator are classified as active and those on the other side are classified as inactive. The most widely used optimization criteria for determining the coefficients (which we used in our experiments) is the partial least squares criteria [7], which suggests to minimize the sum of the squares of differences between actual and predicted activity levels of the compounds in a training set. The separator plane which satisfies this criteria is NP-hard to compute deterministically but can be approximated through genetic algorithms, local search heuristics, etc.

Another popular statistical classification method is Linear Discriminant Analysis(LDA) [12]. Given a set of descriptor arrays, LDA computes a linear projection of the descriptor array space into a Euclidean space with 2 or 3 dimensions (i.e. each descriptor array is mapped to a point in the 2/3-D Euclidean space). The projection aims to maximize the ratio of between-class variance and within-class variance. The projection of descriptor arrays to points in the Euclidean space is followed by the computation of a line/plane which best separates the active and inactive compounds, i.e. maximizes the accuracy of the classifier. For a given query compound with unknown activity, its class is then simply determined by checking to which subspace its projection falls into; clearly this can be performed very fast.

It is also possible to perform compound classification via well known machine-learning techniques such as SVM (Support Vectors Machines) [19] and, more commonly, ANN (Artificial Neural Networks)[21].

Although k-nn classification is a conceptually simple approach and is applied to solve several chemistry and biology problems, it was not considered for small molecule classification until recently [20; 9]. For each predefined number of variables, it seeks to optimize (i) the number of nearest neighbor k used to estimate the activity of each compound (ideal case is where k=1) (ii) selection of variables from the original set of all descriptors. The compounds are then compared under the standard (unweighted) L_1 or L_2 distance.

4. DISTANCE MEASURES FOR SMALL MOLECULES AND DISTANCE BASED CLASSIFICATION

Given a chemical compound s, its descriptor array S is defined to be an n dimensional vector in which each dimension i, denoted by $S[i]$, is a real value corresponding to the descriptor associated with dimension i. For a given bioactivity, it is of significant interest to come up with a distance measure $D(S, R)$ between pairs of descriptor arrays S and R that correspond to the similarity in the bioactivity levels of the corresponding compounds s and r: if the bioactivity levels are similar, the distance must be small and vice versa. Such a distance measure could be very useful in the classification of *new* chemical compounds in terms of the bioactivity of interest: the bioactivity level of the new compound is likely to be identical to the bioactivity level of its closest neighbors.

Metric distances are particular interesting candidates for fast similarity search, due to the availability of efficient data structures. A distance measure D forms a metric if the following conditions are satisfied. (i) $D(S, S) = 0$ for all S and $D(S, R) \geq 0$ for all S and R (non-negativity). (ii) $D(S, R) = D(R, S)$ (symmetry). (iii) $D(S, R) \leq D(S, Q) + D(Q, R)$ (triangle inequality). Metric distance of interest include the Hamming distance, Euclidean distance and the Tanimoto distance.

The commonly used QSAR approach estimates the level of bioactivity of a compound via a linear combination of its descriptors. In *distance based* compound classification, it is natural to consider a distance between two descriptor arrays which is a linear combination of the differences in each one of the dimensions.

More specifically one can define $D(S, R) = \sum_{i=1}^{n} w_i \cdot |S[i] - R[i]|$ where w_i, the weight of the dimension i is a real value in the range $[0, 1]$. It is easy to show that this distance, which is usually called the weighted Minkowski distance of

[1] To the best of our knowledge all recent studies in this direction show how to assign *binary values* to weights w_i i.e. how to choose the specific descriptors that are most relevant for the application of interest (e.g. [20; 9]). As will become clear later in the paper, we show how to compute optimal *real valued weights* so as to improve the predictive power of our classifier.

order 1 forms a metric.

In this paper we focus on classification of biomolecules according to binary bioactivities. The biomolecular data sets available usually do not specify the level of bioactivity of interest but rather provide whether a compound is active or inactive. Thus we only perform a binary classification of compounds for each bioactivity, although our methods are general to provide a real valued level of bioactivity.

Our classification method for a given bioactivity first computes a distance measure for a training data set which *separates* the subset of active compounds from those that are inactive. Given a training set of descriptor arrays $T = \{T_1, T_2, ..., T_\ell\}$ (each of which belonging to a compound) we determine the distance measure D, more specifically compute the associated weights w_i, through a combinatorial optimization approach.

Given the training set T, let $T^A = \{T_1^A, T_2^A, ..., T_m^A\}$ denote its subset of active compounds and $T^I = \{T_1^I, T_2^I, ..., T_{\ell-m}^I\}$ denote its subset of inactive compounds. Clearly $T = T^I \cup T^A$.

We obtain a linear program for determining each w_i as follows. The objective function of the linear program which is to be minimized is

$$f(T) = (\sum_{h=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{n} w_i \cdot |T_h^A[i] - T_j^A[i]|)/m^2 \qquad (1)$$

$$+ (\sum_{h=1}^{\ell-m}\sum_{j=1}^{\ell-n}\sum_{i=1}^{n} w_i \cdot |T_h^I[i] - T_j^I[i]|)/(\ell - m)^2 \qquad (2)$$

$$- (\sum_{h=1}^{m}\sum_{j=1}^{\ell-m}\sum_{i=1}^{n} w_i \cdot |T_h^A[i] - T_j^I[i]|)/(m \cdot (\ell - m)) \qquad (3)$$

subject to the following conditions

$$\forall T_h^A \in T^A \quad (\sum_{j=1}^{m}\sum_{i=1}^{n} w_i \cdot |T_h^A[i] - T_j^A[i]|)/m^2$$

$$\leq (\sum_{j=1}^{\ell-m}\sum_{i=1}^{n} w_i \cdot |T_h^A[i] - T_j^I[i]|)/(m \cdot (\ell - m)) \qquad (4)$$

$$\forall i \quad 0 \leq w_i \leq 1 \quad \& \quad \sum_{i=1}^{n} w_i \leq C \qquad (5)$$

where C is a user defined constant.

The objective function $f(T)$ has three components: Component (1) is the average distance among active compounds and component (2) is the average distance among the inactive compounds; their sum provides the *within-class* average distance. Component (3), on the other hand, is the average distance between an active compound and an inactive one; thus it stands for the *between-class* average distance. As a result our linear programming formulation aims to maximize the difference between the average between-class distance and the average within-class distance. The distance measure obtained will *separate* the typical active compound from the typical inactive compound, while *clustering* all active compounds and all inactive compounds as much as possible.

There are three types of constraints on the weights w_i in our linear programming formulation. Constraint (4) ensures that the average distance among active compounds is no more than the average distance between active and inactive compounds.[2] Constraints (5) impose bounds on the values of weights w_i and their sum.[3]

5. EFFICIENT DATA STRUCTURES FOR K-NN SEARCH

A distance measure defined as above can be used for the classification of compounds with unknown levels of bioactivity as the bioactivity level of a compound is likely to be similar to the bioactivity levels of compounds within its close proximity. Our *k-nn classifier* estimates the (binary) bioactivity of a given compound by (1) either taking the majority of the bioactivities of its k-nearest compounds w.r.t. the distance measure or by (2) checking whether sum of the binary bioactivity levels of the k-nearest neighbors normalized by their distances to the compound is above a threshold value. Under each approach, it is possible to select the value of k which maximizes the accuracy of the estimator, i.e. the ratio of the sum of true positives and true negatives to the size of the training data set.

Once the method of classification is determined, it is desirable to construct an efficient data structure for performing k-nn search. In the remainder of the paper we focus on constructing an efficient k-nn search data structure for the metric distance we developed and provide some experimental results.

Efficient data structures for k-nn search

Typical similarity search methods for large collections of data elements usually perform iterative partitioning of the data set into smaller subsets so as to perform efficient querying by pruning - which is achieved at each iteration by checking out into which partition the query falls[16; 18]. The pruning strategy can be made particularly effective on data collections where similarity is measured with respect to a metric distance. The partitions in such a metric space are usually achieved with respect to simply defined planar cuts; given a query element, it is quite simple to check into which side of the planar cut it falls.

Given a set of data elements $X = \{X_1, \ldots, X_\ell\}$ in a metric space with distance D, similarity search for a query element Y can be posed in two flavors. (1) Range query: retrieve all items whose distance to Y is at most some user defined R.

[2] A more stringent set of constraints can be imposed on active compounds such that the distance between a given active compound T_h^A and any other active compound is no more than the distance between T_h^A and any inactive compound. Such a set of constraints can, in principle, can separate active and inactive compounds into tighter clusters. Unfortunately, the number such constraints, $m^2 \cdot (\ell - m)$, turns out to be impractical, even for the most advanced linear program solvers.

[3] The number of descriptors related to a specific bioactivity is usually no more than a few, thus it is desirable to simplify the distance measure by limiting the number of non-zero weights. The final constraint aims to achieve this by imposing an upper bound on the sum of the weights. Although this constraint does not guarantee to upper bound the number of non-zero weights, in practice, the number of non-zero weights obtained is no more than $2C$.

(2) k-nn query: retrieve the $k \geq 1$ items whose distances to Y are as small as possible.

Efficient data structures for performing nearest neighbor search in high dimensional metric spaces usually exploit the triangle property satisfied by the metric distance measure. One particularly efficient similarity search tool for performing range queries is the Vantage Point (VP) trees [16; 18]. In a VP tree, efficient similarity search in a large data set is achieved through iterative pruning. Traditionally, a vantage point tree is defined as a binary tree that recursively partitions a data set into two equal size subsets according to a randomly selected vantage point X_v as follows. Let M is the median distance among the distances of the data elements to X_v. The *inner partition* consists of the elements Y such that $D(X_v, Y) < M$ and the *outer partition* consists of the elements Z such that $D(X_v, Z) \geq M$. The two subsets are further partitioned via the iterative application of the above procedure until each subset includes a single data element.

For a given query element Y, the set of data elements X_i for which $D(Y, X_i) \leq R$ for the search radius R can be computed as follows. Let X_v be the vantage point chosen for the entire data set and let M be the median distance among the distances of the data elements to X_v. If $D(X_v, Y) + R \geq M$ then recursively search the *outer partition*. If $D(X_v, Y) - R < M$ then recursively search the *inner partition*. If both conditions are satisfied then both partitions must be searched implying that no pruning has been achieved. The correctness of the search routine follows from the triangle inequality.

A natural extension to the traditional vantage point trees is what we call the *Space Covering VP trees* (SCVP trees) first described by Sahinalp et al [15]. At each level of the SCVP trees, multiple vantage points are chosen so as to increase the chance of inclusion of the query region in one of the inner partition of the vantage points. This can be achieved by selecting vantage points in a way that the union of the inner partitions of these vantage points cover the entire data set. In other words, each data element is included in at least one of the inner partitions of a vantage point. Thus a SCVP tree has multiple branches at each internal node, each representing a vantage point and its inner partition. If a query element is not close to any of the vantage points at a given level, it is deduced that there are no similar items to it in the data set. The original SCVP trees chose the vantage points at each level randomly. Although this approach can perform quite well for certain data collections, it can also result in poor space utilization. The SCVP trees introduce some redundancy in the representation of the data elements: clearly each data element may be included in more than one inner partition and thus need to be represented in more than one subtree. Thus the memory requirements of the SCVP tree can be fairly large.

Clearly it is desirable to *cover* the entire data collection by the fewest number of (inner partitions of) vantage points. However, the problem of minimizing the number of vantage points for this purpose turns out to be an NP-hard problem under all distance measures of interest (i.e. weighted Minkowski distance of any order p, wL_p)[10]. Nevertheless it is possible to approximate the minimum number of vantage points in any metric space through a simple polynomial time algorithm as we show later. As a result we obtain a data structure that deterministically picks the vantage points (whose inner partitions cover the entire data set) which results in almost optimal redundancy; we call this data structure *Deterministic Multiple Vantage Point* tree (DMVP tree) [10].

An $O(\log \ell)$ approximation to the optimal vantage point selection

The variant of the optimal vantage point selection problem (OVPS) for which we establish NP-hardness assumes a fixed radius r for each neighborhood around a vantage point. One can think of two natural variants of the OVPS problem: (1) each neighborhood includes a fixed number of points (e.g. $\ell/2$ points as per the original VP Tree construction), (2) each neighborhood has at least ℓ/k and at most ℓ/k' points for some $k \geq k'$. It is not difficult to show that these variants are NP-hard as well.

In the remainder of the paper we focus on variant (2) of the OVPS problem and describe a polynomial time $O(\log \ell)$ approximation algorithm for solving it. Such a solution will also imply an $O(\log \ell)$ approximation algorithm for variant (1) by setting $k = k'$. The approximation algorithm is achieved by reducing the OVPS problem to the weighted set cover problem as follows.

Consider each point X_i in S. We construct the following ℓ sets for X_i named $X_i^1, X_i^2, \ldots, X_i^\ell$. X_i^1 consists of only X_i. X_i^2 consists of X_i and its nearest neighbor. In general, X_i^j consists of X_i and its $j-1$ nearest neighbors. Let the cost of X_i^j be j.

Now given sets X_i^j, for all $1 \leq i \leq \ell$ and $k \leq j \leq k'$, each with cost j, if we can compute the minimum cost collection of sets such that each $X_h \in S$ is in at least one such set, we would get a solution to the variant (2) of the OVPS problem. This problem is equivalent to the weighted set cover problem for which a simple greedy algorithm provides an $O(\log \ell)$ approximation (e.g. [5]). The greedy algorithm works iteratively: each iteration simply picks a set where the cost-per-uncovered-element is minimum possible. The algorithm terminates when all elements are covered.

Optimal fitting of the multiple vantage point tree in the memory

Although the deterministic multiple vantage point tree improves the memory usage of the randomized space covering vantage point tree, it is still possible that the tree may not fit in the main memory. If this is indeed the case, we try to place a connected subtree (which includes the root) to the memory. The search again is performed starting with the root. When an internal node whose children are not represented in the memory is reached, the search is done in a brute force manner on the set of points represented by that node.

Clearly it is of interest to obtain the *best* subtree for optimizing the query performance of the data structure. For that we use the following $0-1$ programming formulation [10].

Given a Multiple Vantage Point tree T and a node i, let S_i be the number of points in the neighborhood represented by i. During a search, when a node j is reached, its children $i, i+1, \ldots$ are considered for further search in linear order; i.e. we first check whether the query fits in the neighborhood of i, then we check $i+1$ and so on until a suitable vantage point $i+h$ is found. Let S'_{i+h} be the number of points in the neighborhood represented by node $i+h$ which are not in the neighborhoods represented by $i, i+1, \ldots, i+h-1$.

Our $0-1$ programming formulation sets the *probability* that node $i+h$ is reached during a search to S'_{i+h}/ℓ. If the children of the node $i+h$ are not placed in the memory, i.e. if node $i+h$ is on the *cut-set*, the time needed for performing a search on the neighborhood represented by this node is S_{i+h}. Thus the expected contribution of node $i+h$ to the query time is $S_{i+h} \cdot S'_{i+h}/\ell$.

Let b_i be a binary variable, which takes the value 1 if vertex i is in the cut-set and is 0 otherwise. Our goal is to minimize the expected running time of the brute-force search performed for each query; i.e. our objective function is $f(T) = \sum_{\forall i} b_i S_i S'_i$ subject to the following constraints.

For any pair of consecutive sibling nodes i and $i+1$, we must have $b_i = b_{i+1}$.

We should not exceed the memory M dedicated to the cut-set; thus $\sum_{\forall i} b_i S_i \leq M$.

Finally, at least one node in every path from the root to a leaf in T must include one vertex in the cut-set. Thus for any such path P we have $\sum_{i \in P} b_i = 1$.

A $0-1$ assignment to b_i's that minimize the objective function will minimize the expected query time while fitting the data structure in the main memory.

6. PRELIMINARY EXPERIMENTS

In this section we aim to provide some insight into the comparative performance of our k-nn classifier, both in terms of accuracy and efficiency. We applied our classifier to five types of bioactivities [11]: (i) being antibiotic, (ii) being a bacterial metabolite, (iii) being a human metabolite, (iv) being a drug, and (v) being drug-like.

The first data set we used is the complete small molecule collection from [4], which includes 520 antibiotics, 562 bacterial metabolites, 958 drugs, 1202 drug-like compounds, and an additional 1104 human metabolites. The total number of the compounds in the data set is 4346. Each compound in the data set is represented with a descriptor array of 62 dimensions, which is a combination of 30 inductive QSAR descriptors [4] and 32 physicochemical properties such as molecular weight, number of specific atoms (O, N, S), acidity, density, etc. This data set was used for testing the classification accuracy of k-nn approach. A second data set which enriches the first data set by the addition of 20000 additional drug like compounds was later used for testing performance of DMVP tree.

For each bioactivity, a wL_1 distance is determined to establish a *model* for compound classification w.r.t. this bioactivity using our k-nn method. Note that the descriptors of each compound are normalized according to the observed maximum and minimum values in the data set in order to remove the bias to parameters with larger values.

The comparative results of the four classification methods, namely k-nn, LDA,MLR and ANN are provided in Table 1. For each bioactivity, we provide the sensitivity, specificity and accuracy obtained by each classifier. We demonstrate the performance of our k-nn classifier only for $k=1$; i.e. given a query compound, our classifier returns the bioactivity of its nearest neighbor in the training data set. It is possible to set $k>1$, however it requires determining the best k value, as well as the best method for assigning the bioactivity of the query compound such as majority rule or distance weighted majority rule. In order to keep our classifier simple, we set $k=1$.

We constructed the wL_1 measure for three different values of C - the upper bound on the sum of weights, i.e., $\sum_{i=1}^{n} w_i \leq C$. Setting $C=\infty$ removes the restriction on the sum of weights and thus computes the wL_1 distance that achieves the best classification. We also set C to 3 and 10 to restrict the number of non-zero weights, with the aim of focusing only on the C most relevant descriptors to the bioactivity of interest. As the resulting non-zero weights turned out to be equal to or very close to 1, these two classifiers are quite similar to those described in recent papers (e.g. [20; 9]) that focus on determining the most relevant descriptors for modeling a bioactivity of interest.

We used MOE(Molecular Operating Environment) PLS module for MLR classification and SNNS (Stuttgart Neural Network Simulator) with default parameters (52 nodes and 420 connection network) for ANN classification. LDA classification is performed through the use of standard C libraries for matrix operations.

For each bioactivity, a *training data set* comprising of 70 percent of both the active and the inactive compounds are formed via random selection. The remaining compounds are used as the *test data set*. Each training data set is used for building the four classifiers corresponding to the related bioactivity and the test data is used for the evaluating their performance.

For each bioactivity/classifier pair we report the following test results: The number of true positives (T_P), the number of true negatives (T_N), the number of false positives (F_P), the number of false negatives (F_N), sensitivity (T_P/(T_P+F_N)), specificity (T_N/(T_N+F_P)), accuracy ((T_N+T_P)/(T_P+T_N+F_P+F_N)), positive predictive value (T_P/(T_P+F_P)), negative predictive value (T_N/(T_N+F_N)).

Our similarity search data structure for computing the nearest neighbor of the query compound is quite efficient, especially when compared to brute force search. We tested our data structure under the wL_1 distance computed for each of the five bioactivities, on both of the data sets. The crucial parameter that determines the performance of our data structure is the pruning it achieves for any given query compound. Thus we determined the percentage of compounds pruned in the second training data set (the first training data set enriched with 20000 drug like compounds), averaged over all compounds in the test data set. On a 32GB Sun Fire V40Z server (with 2.4 Ghz AMD 64bit Opteron processor) the respective pruning ratios are as follows. We achieved (i) 84.4% pruning for being antibiotic, (ii) 84.5% pruning for being bacterial metabolite, (iii) 86.1% pruning for being human metabolite, (iv) 81.7% pruning for being drug, and (v) 81% pruning for being drug-like. This is significant improvement over brute force search.

As a result our k-nn classifier turns out to be very fast. On the first data set, the running time of our k-nn classifier averaged over all 4346 compounds (training+test data sets) and all five bioactivities is 0.3 milliseconds on the above server. In contrast the ANN classifier requires 39.7 milliseconds on the same data set. On the second data set (which simply has additional 20000 compounds in the data structure) the running time of our k-nn classifier increases only to 1.3 milliseconds (again averaged over the 4346 compounds from the first data set and five bioactivities), still 30 times better than the ANN trained over a much smaller set.

Model		T_P	T_N	F_P	F_N	SPEC	SENS	ACCUR	PPV	NPV
Antibacterial Model, C=∞	Train	269	2610	69	95	.97	.74	.95	.8	.96
	Test	117	1119	28	39	.98	.75	.95	.81	.97
Antibacterial Model, C=10	Train	224	2538	141	140	.95	.62	.91	.61	.95
	Test	92	1085	62	64	.95	.59	.90	.60	.94
Antibacterial Model, C=3	Train	201	2526	153	163	.94	.55	.90	.57	.94
	Test	75	1074	73	81	.94	.48	.88	.51	.93
Antibacterial Model, LDA	Train	364	0	2679	0	0.00	1.00	0.12	0.12	-
	Test	156	0	1147	0	0.00	1.00	0.12	0.12	-
Antibacterial Model, MLR	Train	194	564	2115	170	0.21	0.53	0.25	0.08	0.77
	Test	61	1129	18	95	0.98	0.39	0.91	0.77	0.92
Antibacterial Model, ANN	Train	294	2651	27	70	0.99	0.81	0.97	0.92	0.97
	Test	129	1132	16	27	0.99	0.83	0.97	0.89	0.98
Bacterial Metabolite Model, C=∞	Train	311	2537	112	83	.96	.79	.94	.74	.97
	Test	135	1091	44	33	.96	.80	.94	.75	.97
Bacterial Metabolite Model, C=10	Train	220	2436	213	174	.92	.56	.87	.51	.93
	Test	98	1038	97	70	.91	.58	.87	.50	.94
Bacterial Metabolite Model, C=3	Train	152	2376	273	242	.90	.39	.83	.36	.90
	Test	80	1018	117	88	.90	.48	.84	.41	.92
Bacterial Metabolite Model, LDA	Train	240	2587	62	154	0.98	0.61	0.93	0.79	0.94
	Test	90	1088	47	78	0.96	0.54	0.90	0.66	0.93
Bacterial Metabolite Model, MLR	Train	301	2525	124	93	0.95	0.76	0.93	0.71	0.96
	Test	119	1073	62	49	0.95	0.71	0.91	0.66	0.96
Bacterial Metabolite Model, ANN	Train	338	2597	52	55	0.98	0.86	0.96	0.87	0.98
	Test	159	1076	59	10	0.95	0.94	0.95	0.73	0.99
Drug Model, C=∞	Train	474	2158	214	197	.91	.71	.86	.69	.92
	Test	204	928	88	83	.91	.71	.87	.70	.92
Drug Model, C=10	Train	349	2072	300	322	.87	.52	.80	.54	.87
	Test	151	861	155	136	.85	.53	.78	.49	.86
Drug Model, C=3	Train	305	2026	346	366	.85	.45	.77	.47	.85
	Test	126	846	170	161	.83	.44	.75	.43	.84
Drug Model, LDA	Train	0	2372	0	671	1.00	0.00	0.78	-	0.78
	Test	0	1014	2	287	0.99	0.00	0.78	0.00	0.78
Drug Model, MLR	Train	279	2234	138	392	0.94	0.42	0.83	0.67	0.85
	Test	109	951	65	178	0.94	0.38	0.81	0.63	0.84
Drug Model, ANN	Train	489	2178	194	182	0.92	0.73	0.88	0.72	0.92
	Test	177	978	39	110	0.96	0.62	0.89	0.82	0.90
Druglike Model, C=∞	Train	674	2043	158	168	.93	.80	.89	.81	.92
	Test	281	866	77	79	.92	.78	.88	.0.78	.92
Druglike Model, C=10	Train	560	1959	242	282	.89	.67	.83	.70	.87
	Test	239	842	101	121	.89	.66	.83	.70	.87
Druglike Model, C=3	Train	467	1813	388	375	.82	.55	.75	.55	.83
	Test	197	275	168	163	.82	.55	.75	.54	.83
Druglike Model, LDA	Train	683	1917	284	159	0.87	0.81	0.85	0.71	0.92
	Test	295	801	142	65	0.85	0.82	0.84	0.68	0.92
Druglike Model, MLR	Train	665	1951	250	177	0.89	0.79	0.86	0.73	0.92
	Test	282	812	131	78	0.86	0.78	0.84	0.68	0.91
Druglike Model, ANN	Train	734	2086	114	107	0.95	0.87	0.93	0.87	0.95
	Test	334	891	52	27	0.94	0.93	0.94	0.87	0.97
Human Metabolite Model, C=∞	Train	773	2270	0	0	1.00	1.00	1.00	1.00	1.00
	Test	331	972	0	0	1.00	1.00	1.00	1.00	1.00
Human Metabolite Model, C=10	Train	772	2266	4	1	.99	.99	.99	.99	.99
	Test	330	972	0	1	1.00	0.99	.99	1.00	.99
Human Metabolite Model, C=3	Train	772	2270	0	1	1.00	0.99	.99	1.00	.99
	Test	330	972	0	1	1.00	0.99	.99	1.00	.99
Human Metabolite Model, LDA	Train	773	2270	0	0	1.00	1.00	1.00	1.00	1.00
	Test	331	972	0	0	1.00	1.00	1.00	1.00	1.00
Human Metabolite Model, MLR	Train	773	2270	0	0	1.00	1.00	1.00	1.00	1.00
	Test	331	972	0	0	1.00	1.00	1.00	1.00	1.00
Human Metabolite Model, ANN	Train	773	2270	-0	0	1.00	1.00	1.00	1.00	1.00
	Test	331	972	0	0	1.00	1.00	1.00	1.00	1.00

Table 1: Binary classification of the bioactivities of the test set according to four classification methods: k-nn, LDA, MLR, ANN.

7. CONCLUSION

We have demonstrated that our k-nn classifier with respect to wL_1 distance obtains better accuracy than the LDA and MLR, sometimes significantly so. It is comparable to the ANN classifier in terms of accuracy and is superior in the sense that it is capable of determining a real valued level of bioactivity rather than giving a simple YES or NO answer. k-nn approach also provides insights into the level of bioactivity or the importance of the descriptors with respect to bioactivity. Analysis of the relative weights of the descriptors for the 5 different bioactivity model demonstrated certain characteristics of these activities. Our models verify that bacterial metabolites and antimicrobial drugs are significantly overlapping which can be attributed to their similar origin. We also observe that human metabolites display distinctive properties compared to the other 4 bioactivities. Another important observation is that QSAR models for drugs and human metabolites are dominated by few descriptors that are correspondingly favored by the drug developers and natural evolution. The distribution of the values for these descriptors may be an important factor for the overlaps among different bioactivities used in our experiments. Overall results of the k-nn classification method bring more insight into the nature and structural dominants of the studied classes of small molecules and if necessary, can help rationalizing the design and discovery of novel antimicrobials and human therapeutics with metabolite-like chemical profiles [11].

Our classifier is faster compared to alternative approaches, thanks to the DMVP tree data structure we develop for fast similarity search. Our DMVP tree data structure improves the existing vantage point tree data structures in multiple ways. It provides a deterministic selection of the optimal vantage points in each level as well as providing the optimal cut of the tree so as to fit it in the available memory. Our data structure can be applied to any metric distance including the wL_p distance for any p and the Tanimoto distance. It performs very well in practice, achieving fast similarity search and classification.

8. REFERENCES

[1] Adamson, G. W., Cowell, J., Lynch, M. F., McLure, A. H. W., Town, W. G., Yapp, A. M. (1973) Strategic Considerations in the Design of a Screening System for Substructure Searches of Chemical Structure Files, *J. Chem. Doc*, **13**, 153-157.

[2] Brown, R. D. (1997) Descriptors for Diversity Analysis, *Persp. Drug Discovery Des.*, **7/8**, 31-49.

[3] Chen, X., Reynolds, C. H. (2002) Performance of Similarity Measures in 2D Fragment-Based Similarity Searching: Comparison of Structural Descriptors and Similarity Coefficients, *J. Chem. Inf. & Comp. Sci.*, **42**, 1407-1414.

[4] Cherkasov, A. (2005) Inductive Descriptors. 10 Successful Years in QSAR, *Curr. Computer-Aided Drug Des.*, **1**, 21-42.

[5] Chvatal, V. (1979) A Greedy Heuristic for the Set Covering Problem, *Math. of Operations Research*, **4**, 233-235.

[6] Cramer, R.D.,Bunce, J.D., and Patterson, D.E. (1988) Crossvalidation, Bootstrapping, and Partial Least Squares Compared with Multiple Regression in Conventional QSAR Studies, *Quant. Struct.-Act. Relat.* **7**, 18-25.

[7] Geladi P.,and Kowalski B. R. (1986) Partial Least-Squares Regression: A Tutorial, *Analytica Chimica Acta*, **185**, 1-17.

[8] Good, A. C., So, S. S., Richards W. G. (1993) Structure - Activity relationships from Molecular similarity Matrices, *J. Medicinal Chemistry*, **36**, 433-438.

[9] Itskowitz, P., and Tropsha, A. (2005) Kappa Nearest neighbors QSAR modeling as a variational problem: theory and applications, *J. Chem. Inf. Model.*, **45(3)**, 777-85.

[10] Karakoc, E., Cherkasov A., and Sahinalp S.C. (2006) Distance based ALgorithms for Small Biomolecule Classification and Structural Similarity Search, *ISMB'06 Intelligent Sytems for Molecular Biology*, May 2006.

[11] Karakoc, E., Sahinalp S.C., and Cherkasov A. (2006) Comparative QSAR- and Fragments Distribution Analysis of Drugs, Druglikes, Metabolic Substances and Antimicrobial Compounds, *Journal of Chemical Information and Modelling*, **46(5)**, 2167-2182.

[12] Livingstone, D. J. (1995) Data analysis for chemists. Applications to QSAR and chemical product design, *Oxford Univ. Press*, 239.

[13] MACCS II Manual, MDL Information Systems, Inc 14600 Catalina Street, San Leandro, CA 94577 USA.

[14] Maggiora, G. M., Johnson, M. A. (1990) Concepts and Applications of Molecular Similarity, *Wiley*, New York.

[15] Sahinalp, S. C., Tasan, M., Macker, J., Ozsoyoglu Z. M.(2003) Distance-Based Indexing for String Proximity Search, *Proc. IEEE Int. Conf. on Data Eng.*, **19**, 135-138.

[16] Uhlmann, J. K. (1991) Satisfying general proximity/similarity queries with metric trees, *Inf. Proc. Lett.*, **4**, 175-179.

[17] Willett, P., Banard, J. M., and Downs, G. M. (1998) Chemical Similarity Searching, *J. Chem. Inf. & Comp.. Sci.*, **38 (6)**, 983 -996.

[18] Yianilos, P. N. (1993) Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces, *Proc. ACM-SIAM Symp. on Discr. Alg.*, **1**, 311-321.

[19] Zernov, V. V., Balakin, K. V., Ivaschenko, A. A., Savchuk, N. P., Pletnev, I. V. (2003) Drug Discovery Using Support Vector Machines. The Case Studies of Drug-likeness, Agrochemical-likeness, and Enzyme Inhibition Predictions, *J. Chem. Inf. & Comp. Sci.*, **43(6)**,2048-2056.

[20] Zheng, W. and Tropsha, A. (2000) Novel Variable selection quantitative structure-property relationship approach based on the k-nearest neighbor principle, *J. Chem. Inf. & Comp. Sci.*, **40**, 185.

[21] Zupan, J., Gasteiger, J. (1999) Neural Networks in Chemistry and Drug Design, 2nd ed., *Wiley*, New York.