CHEMICAL COMPOUND CLASSIFICATION ENSEMBLE

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Ya Zhu

In Partial Fulfillment
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

December 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Chemical Compound Classification Ensemble

By

Ya Zhu

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Changhui Yan
Chair

Dr. Juan Li

Dr. Jun Kong

Dr. Limin Zhang

Approved:

December 3, 2013
Date

Dr. Brian Slator
Department Chair

ABSTRACT

In this paper, we try to address this two-class classification problem using global and local similarity between compounds. The global similarity measures the overall structural resemblance between two compounds. Local similarity is computed based on the occurrences of common sub-structures between compounds. We built several classification models based on global and local similarity. To improve the classification result, we used an ensemble of those models to predict the function compounds in NCI cancer data sets.

We predict whether a compound can inhibit cancer cell growth or not, obtaining AUC higher than 80% for five datasets. We compare our results with other state-of-the-art methods. Our classification result is the best in all five datasets.

Our results show that local similarity is more useful than global similarity in predicting compound function. An ensemble method integrating global and local similarity achieves much better performance than single predicting models.

ACKNOWLEDGMENTS

I would like to thank a number of people for their support of the development of this work. First of all, I thank my advisor Dr. Changhui Yan and the other members of my committee, Dr. Juan Li, Dr. Jun Kong, and Dr. Limin Zhang. Finally I thank my family for encouraging and supporting all my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Researchers have observed that chemical compounds with similar structures often have similar functions. This observation motivated researchers to develop computational methods that can predict the function of new chemical compounds by comparing the compound's structure with other compounds whose function is known. The similarity could be measured based on global and local information. For methods using the global information, a value between 0 and 1 is used to show the overall structure similarity between two compounds. The higher the value, the more similar they are. 1 means these two compounds are identical. Methods using local information compute the similarity between two compounds based on the occurrence of common substructures in the compounds. For each substructure, 1 means it occurs in a compound, and otherwise. In the comparison of compound structures, many substructures could be considered. Thus, similarity measured by local information can be described as a matrix of 0/1.

We addressed the problem of predicting compound function using a two-class classification approach with classification methods, *such as* SVM, Random Forest, *etc.* Each combination of feature set and classification algorithm will output a prediction, a discriminate value, for a new compound. A discriminate value is a real number between 0 and 1. The higher the discriminate value, the more likely this new compound may has a certain function.

We used an ensemble method to integrate the discriminate values output by the single classification methods. Our ensemble result is better than previous works.

1.1. Chemical Compound Similarity

Scientists try to find the relationship between the chemical and/or 3D structure of a molecule and its biological activity, this is structure activity relationships (SAR). To discover SAR, they use global structural similarity and maximum common sub-structure, a type of local similarity.

When global information is considered, scientists usually use graph kernels and SVM to compare the compound structure. When local information is considered, they investigate the occurrence pattern of some maximum common sub-graph (MCS) in the compound structure and use classification method like SVM to predict function. MCS is widely used in graph mining. Table 1 is an overview of classification algorithms and the two kinds of information used in chemical compound function prediction.

Table 1. Methods and features

Methods	Local similarities	Global similarities
SVM	MCSs	Graph kernels
Random Forest	MCSs	
Elasticnet	MCSs	
Generalized Linear Model	MCSs	
Artificial Neural Network	MCSs	
k-Nearest Neighbor	MCSs	
Gradient Boosted Regression Trees	MCSs	

1.2. Applications

In silico research, in which biological process is simulated or evaluated on a computer, has the potential to speedup drug discovery while reducing the need for expensive lab work and clinical trials. Using *in silico* method, researchers can produce and screen drug candidates more effectively. For example, researchers using the protein docking algorithm EADock, found potential inhibitors to an enzyme associated with cancer activity in 2010 50% of the molecules were later shown to be active inhibitors *in vitro*.

The aim of *in silico* screening is to identify molecules of novel chemical structure that bind to the macromolecular target of interest.

1.3. Graph Kernels

The kernel of SVM estimates the global similarity of two instances. The kernel function is a weighting function used in nonparametric function estimation. It gives the weights of the nearby data points in making an estimate. In practice, kernel functions are piecewise continuous, bounded, symmetric around zero, concave at zero, real valued, and for convenience often integrate to one. They can be probability density functions. They maybe in a range such as [0,1] or $[0,+\infty)$.

Kernel functions must be continuous, symmetric, and most preferably should have a positive (semi-) definite Gram matrix. Kernels which are said to satisfy the Mercer's theorem are positive semi-definite, meaning their kernel matrices has no non-negative Eigen values. The use of a positive definite kernel insures that the optimization problem will be convex and solution will be unique.

A graph kernel is a kernel function that take graphs as input. A graph G is described by a finite set of vertices V and a finite set of edges E. We use a labeled undirected graph to represents a compound. That means every vertex and edge is labeled. Simplify speaking, every vertex is an atom, the vertex label is atom type; every edge is a chemical bond, the edge label is chemical bond type.

We tested many graph kernels, and found Tanimoto kernel and Min/Max Tanimoto kernel are the best two kernels. So our introduction of graph kernels will focus on them.

Let $P(d)$ be the set of all possible atom-bond labeled paths containing a maximum of d bonds. Using a depth-first search approach, the binary feature map ϕ for a molecule $\mathbf{u}$ and a given depth d can be written as:

$$\phi_d(\boldsymbol{u}) = (\phi_{path}(\boldsymbol{u}))_{path \in P(d)}$$

Here $\phi_{path}(\boldsymbol{u})$ is equal to 1 if at least one depth-first search of depth less or equal to d starting from one of the atoms of $\mathbf{u}$ produces the path $path$. The counting feature map φ_d is defined similarly by using φ_{path} to count the number of labeled paths of each kind. The difference between then is $\phi_{path}(\boldsymbol{u})$ is 0 or 1, $\varphi_{path}(\boldsymbol{u})$ is the count of paths.

Let $\mathbf{u}$, $\mathbf{v}$ denote two molecules and d be an integer (d is depth), we can define a Tanimoto kernel function $k_d(\mathbf{u},\mathbf{v})$:

$$k_d(\mathbf{u},\mathbf{v}) = \sum_{path \in P(d)} \phi_{path}(\mathbf{u})\phi_{path}(\mathbf{v})$$

This is a dot product.

Tanimoto kernel: Consider the feature map ϕ_d and the corresponding kernel k_d. The Tanimoto kernel k_d^t is defined by:

$$k_d^t(\mathbf{u},\mathbf{v}) = \frac{k_d(\mathbf{u},\mathbf{v})}{k_d(\mathbf{u},\mathbf{u}) + k_d(\mathbf{v},\mathbf{v}) - k_d(\mathbf{u},\mathbf{v})}$$

To simplify, Tanimoto kernel is:

$$\frac{|\phi(\mathbf{u}) \cap \phi(\mathbf{v})|}{|\phi(\mathbf{u}) \cup \phi(\mathbf{v})|}$$

If $\phi(\mathbf{u})$ is regarded as the set of features than can be extracted from u using depth-first search exploration, then k_d^t simply computes the ratio between$|\phi(\mathbf{u}) \cap \phi(\mathbf{v})|$, *i.e.* the number of elements in the intersection of the two sets $\phi(\mathbf{u})$ and $\phi(\mathbf{v})$, and $|\phi(\mathbf{u}) \cup \phi(\mathbf{v})|$, *i.e.* the number of elements in the set corresponding to the union of $\phi(\mathbf{u})$ and $\phi(\mathbf{v})$.

Min/Max Tanimoto kernel: let $\mathbf{u}$, $\mathbf{v}$ denote two molecules and d be an integer. Consider the feature map $\varphi_d(.)$, and the corresponding $\varphi_{path}(.)$. $\varphi_{path}(\boldsymbol{u})$ is the count of paths. The Min/Max Tanimoto kernel k_d^m is defined by:

$$k_d^m(\mathbf{u},\mathbf{v}) = \frac{\sum_{path \in P(d)} \min(\varphi_{path}(\mathbf{u}), \varphi_{path}(\mathbf{v}))}{\sum_{path \in P(d)} \max(\varphi_{path}(\mathbf{u}), \varphi_{path}(\mathbf{v}))}$$

To simplify, Min/Max Tanimoto kernel is:

$$\frac{\sum_i min(U_i, V_i)}{\sum_i max(U_i, V_i)}$$

This kernel function is closely related to the Tanimoto kernel in at least two different ways. First, it is identical to the Tanimoto kernel when applied to binary vectors. Second, in a more subtle way, the Min/Max Tanimoto kernel can be viewed as a Tanimoto kernel on a different set of binary vectors obtained by transforming the vector of counts.

The Min/Max Tanimoto kernel takes into account the frequency of different paths in a molecule and, like the Tanimoto kernel, its value is always between 0 and 1. Using path counts rather than binary indicator variables, the Min/Max Tanimoto kernel produces a more reliable way of assessing similarity between molecules of different sizes.

1.4. Maximum Common Sub-structure

If some chemical compounds can inhibit the growth of one type of cancer cells, they may share one or more molecular sub-structures. If we use graph to represent a compound, these common molecular substractures are common sub-graphs.

From the beginning of this century, many people have used local patterns in mining molecular datasets. Maximum common sub-structure (MCS) has been used in almost every aspects of graph mining. In the field of biological sciences, MCS have been used in applications like finding protein function sites and drug *screening*.

CHAPTER 2. LITERATURE REVIEWS

Different methods have been used to classify compounds. Generally speaking, there are two main categories: methods that use the global information and those use local similarity.

2.1. Global Information

One kind of the methods computes the similarity between compounds using global similarity. Gartner *et al.* introduced many kernels used in Support Vector Machines [Gartner 2003][Gartner& Flach 2003]. Leslie *et al.* used Spectrum kernel [Leslie 2002]. Spectrum kernel is a string kernel using 1D representations of molecules. Swamidass *et al.* compared several graph kernels using 1D, 2D, and 3D representations of molecules [Swamidass 2005]. Researchers usually count the common labeled paths between two graphs to compute the similarity. Ralaivola *et al.* use three graph kernels to predict mutagenicity, toxicity, and anti-cancer activity on three publicly available data sets [Ralaivola 2005]. When counting the paths, some paths are repeated short paths. This problem is known as totters. Mahe *et al.* use Marginalized graph kernel, add a Morgan index process to remove totters [Mahe 2005]. Mahe *et al.* used pharmacophore kernel for virtual screening [Mahe 2006]. Pharmacophore kernel is a 3D graph kernel. Mahe *et al.* provided both exact computation and fast approximations. Ramon *et al.* invented Subtree graph kernel [Ramon 2003]. Mahe *et al.* revised Subtree graph kernel [Mahe 2009]. Wang *et al.* used graph kernels in chemical compound searching [Wang 2010].

2.2. Local Similarity

Another kind of methods use MCS occurrences as features and use SVM *etc.* to classify. Finding the MCS of two graphs is a NP-hard problem. Ballester *et al.* used Shape Signature to achieve a high speed in searching compounds [Ballester 2007]. Cao *et al.* used MCS to search

and predict compound activity [Cao2008]. Ferreira *et al.* integrated ChEBI ontology information with structure information [Ferreira 2010]. Cao *et al.* used geometric embedding and locality sensitive hashing to accelerate compound searching and clustering. [Cao2010]. Schietgat *et al.* found a MCS of two outer planar graphs to accelerate the MCS finding [Schietgat 2010]. Hariharan *et al.* invented an algorithm to find the MCS of 3 or more graphs [Hariharan 2011].

2.3. Use More Information

Eckert compared many similarity measures and methods [Eckert 2007]. Holliday used 25 search engine results as similarity measure to improve the chemical compound classification [Holliday 2011].

CHAPTER 3. MATERIALS AND METHODS

3.1. NCI Cancer Screens

The NCI dataset collection has been made publicly available by the National Cancer Institute and provides screening results for the ability of thousands of compounds to suppress or inhibit the growth of a panel of 60 human tumor cell lines [NCI 2012]. The datasets used here correspond to the parameter GI50, the concentration that causes 50% Growth Inhibition. For each cell line, approximately 3,500 compounds are provided together with information on their ability to inhibit cancer, which defines a two-class classification problem [NCI 2003]. We use the datasets of [Ralaivola 2005], which we requested from Dr. Schietgat.

Schietgat *et al.* only gave the AUC of 5 screens in their paper [Schietgat 2010]. So we use the same 5 screens.

3.2. Graph Kernels to Measure the Global Similarity

Dr. Perret *et al.* open sourced their graph kernel software [Perret 2007]. ChemCpp is an open-source software to generate kernel matrixes. It includes several graph kernels, such as: marginalized graph kernel (sd2gram kernel), spectrum kernels (Tanimoto kernel, Min/Max Tanimoto kernel, Lambda^k kernel), pharmacophore kernels for 3D structure (3D Spectrum kernel, 3D binary kernel, 3D Tanimoto kernel), *etc.* We tested all these kernels and found Tanimoto kernel and Min/Max Tanimoto kernel were far better than others.

Tanimoto kernel and Min/Max Tanimoto kernel can use different depth d. We tested d from 5 to 11. When d is 10, the results are the best. Ralaivola *et al.* found 10 is the best d in their paper too [Ralaivola 2005].

We imported these kernel matrixes into Gist SVM (http://www.chibi.ubc.ca/gist/) to classify compounds.

3.3. Local Similarity Based on MCSs

MCSs are widely used in graph mining. There are some open source software can be used in MCS finding. We use a software called PMCSFG [Schietgat 2013]. It can find a number of MCSs between a randomly selected pair in a graph set. After obtaining the MCSs, we used VF2 Sub-graph Isomorphism algorithm to find whether a MCS occurred in a full graph. These occurrences were used as features. Thus, in local common sub-structure models, all feature values are 0 or 1. VF2 algorithm is explained in Figure 1:

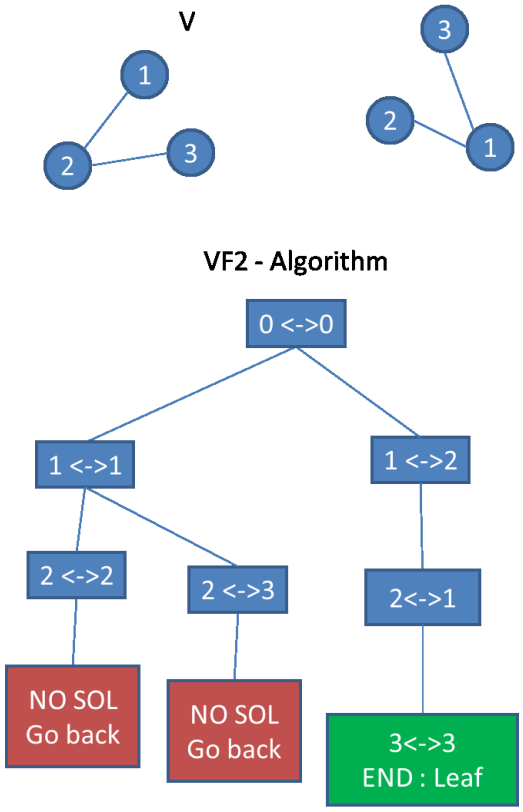

Figure 1. VF2 Algorithm

(From http://stackoverflow.com/questions/8176298/vf2-algorithm-steps-with-example)

The two graphs above are V and the drawing on the right top. The VF2 algorithm is described in Figure 1, step by step.

First, let's use the following notations: XV is the node X in V

We want to know if V is sub-graph isomorphic to the graph on the right; in the VF2 algorithm we will try to match each node in V with a node in right graph.

Step 1: Match empty subset of V with empty subset of right: it always works

Step 2: Now we can match 1V with 1right, 2right or 3right. Match 1V witch 1right: it works.

Step 3: We can match 2V with 2right or 3right. Let's match 2V with 2right: it works because {1V 2V} and {1right 2right} are isomorphic.

Step 4: Try to match 3V with a node in right, but cannot (It would be possible if there was an edge between node 3 and 2 in right, and no edge between 3 and 1). So we go back to Step 2.

Step 5: Match 2V with 3right.

Step 6: Cannot find a solution. Go back to step 2. There is no solution in step 2, so go back to Step 1.

Step 7: Match 1V with 2right.

Step 8: Match 2V with 1right.

Step 9: Match 3V with 3right. It works. We matched {1V 2V 3V} with {2right 1right 3right}. If we test all the solution and it never works, V is not a sub-graph of right.

In this example, V is a sub-graph of right. Actually, these two graphs are isomorphic.

3.4. Classification Methods

We used several classification algorithms.

- Random Forest

Random forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. The algorithm for inducing a random forest was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark. The term came from random decision forests that was first proposed by Tin Kam Ho

of Bell Labs in 1995. The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho and Amit and Geman in order to construct a collection of decision trees with controlled variation.

- Gradient Boosted Regression Trees

Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions. GBRT is an accurate and effective off-the-shelf algorithm that can be used for both regression and classification problems. The advantages of GBRT include natural handling of data of mixed type (heterogeneous features), predictive power, and robustness to outliers in input space (via robust loss functions). The disadvantage of GBRT is scalability. Due to the sequential nature of boosting it can hardly be parallelized.

- Support Vector Machine (SVM)

The principle of SVM is illuteacted in Figure 2.

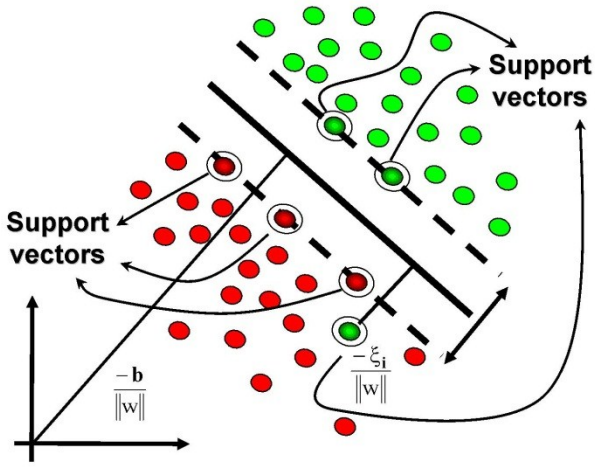

Figure 2. Support Vector Machine and Support Vectors

(From http://kokzard.blogspot.com/2011/10/jfjkdshfkjsldf.html)

SVMs are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. The basic SVM takes a set

of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, a SVM training algorithm builds a model that assigns new examples into one category or the other. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

- Elasticnet

The elastic net is a regularized regression method that combines the L1 and L2 penalties of the lasso and ridge methods.

- Generalized Linear Model

The generalized linear model is a flexible generalization of ordinary linear regression that allows for response variables that have other than a normal distribution. The generalized linear model generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

- Artificial Neural Networks (ANN)

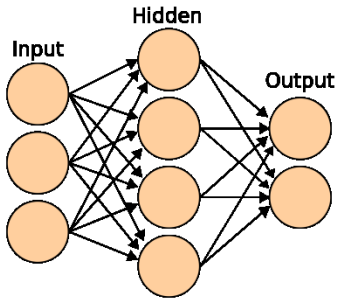

Figure 3. Artificial Neural Networks

(From http://en.wikipedia.org/wiki/Artificial_neural_network)

Artificial neural networks are composed of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Simple artificial nodes, called "neurons", are connected together to form a network which mimics a biological neural network.

- k-Nearest Neighbor algorithm (kNN)

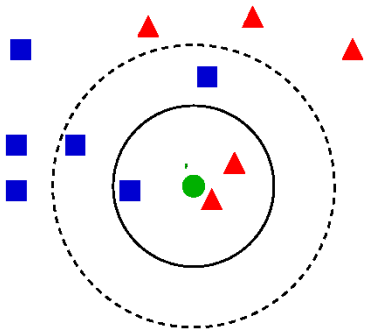

Figure 4. k-Nearest Neighbor Algorithm

(From http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)

The k-nearest neighbor algorithm (kNN) is a method for classifying objects based on closest training examples in the feature space. kNN is a type of instance-based learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer). We find 7 is the best k in our experiment.

3.5. Feature Selection

As we increase the number of MCS, AUC value increases, too. But it takes more and more time. Most of the MCSs we found are useless. They include hydrocarbons or/and benzene ring, *etc.* These MCSs appear in almost every organic compound. We use Random Forest to obtaining a ranking of the relative importance of the MCSs [Kursa 2010]. Eliminating useless variables

reduced model training time and improved performance. After feature selection, only 300-400 features in 6,400 are selected. The training time decease drastically, and AUC, F1 score increase slightly. It allows us to try more algorithms.

3.6. Ensemble

We used an ensemble method to combine the global and local information, and made use of multi classification algorithms to get better result. We tested two simplest ensemble methods: mean and median. The mean ensemble was better than the best single model in some screens. The median ensemble was better than mean ensemble and the best single model in every screen.

We used MATLAB fitensemble() function. All ensemble methods: AdaBoostM1, LogitBoost, GentleBoost, RobustBoost, LPBoost, TotalBoost, RUSBoost, Bag, and Subspace Discriminant, were tested. Subspace Discriminant was far better than others. As previous works showed, using only a portion of models is better than using all of them. We found the ensemble of 5 models was better than that of 9 models. Add more models could not improve the result. Subspace Discriminant result was better than median. Every model output a discriminant value for every instance in training and testing set. The discriminant value was a real number in [0, 1]. We trained fitensemble() function on the discriminant values of all training sets.

Lastly, we used the weighted liner blending method. This method was used in Netflix prize, Heritage Health Prize, and many KDD cup contests. Weighted liner blending gave the best result. We used Particle Swarm Optimization (PSO) to find the best weight set. Assume that we have a swarm of S particles, the algorithm of PSO proceed as follows:

- For each particle $i = 1, ..., S$ do:

- Initialize the particle's position with a uniformly distributed random vector: $x_i \sim U(b_{lo}, b_{up})$, where b_{lo} and b_{up} are the lower and upper boundaries of the search-space.
- Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
- If $f(p_i) > f(g)$ update the swarm's best known position: $g \leftarrow p_i$
- Initialize the particle's velocity: $v_i \sim U(-v_{up}, v_{up})$

- Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:
  - For each particle i = 1, ..., S do:
    - For each dimension d = 1, ..., n do:
      - Update the particle's velocity:

$$v_{i,d} = wv_{i,d} + c_1 rand()(p_{i,d} - x_{i,d}) + c_2 rand()(g_d - x_{i,d})$$

        w is inertia weight, c_1 and c_2 are acceleration constants, $rand()$ returns a random number in [0,1).
    - Update the particle's position: $x_i \leftarrow x_i + v_i$
    - If $f(x_i) > f(p_i)$ do:
      - Update the particle's best known position: $p_i \leftarrow x_i$
      - If $f(p_i) > f(g)$ update the swarm's best known position: $g \leftarrow p_i$
- Now g holds the best found solution.

We use the mean of AUC as the f( ) function. One set of model weights are coordination of a particle. When PSO find the best g, this set of weights gives the highest mean of AUC. To avoid over fitting, we use one set of model weights for all screens. If every screen use its own weight set, the AUC will higher.

PSO is running on the discriminant values of all testing sets. To avoid over fitting, we train PSO on 3 screens: 786-0 M14 SNB-19. Test the result on other 2 screens: HCT-116 NCI-H522.

3.7. Environment

We use 64bits Linux on NDSU CCAST clusters and Computer Science department laboratory, utilize the up-to-date Intel Composer XE 2013 for Linux and Intel Math Kernel Library SMP. Use -O3 -funroll-loops -march=core2 pthread OpenMP. So the programs can use multi cores. We compiled R, R packages, ChemCpp, and our own programs.

CHAPTER 4. RESULTS

4.1. Definition of Measures Used

For a two-class classification problem, the TP FP TN FN is defined as:

	Actual class (observation)	
Predicted class (expectation)	TP (True Positive) Correct result	FP (False Positive) Unexpected result
	FN (False Negative) Missing result	TN (True Negative) Correct absence of result

The precision, recall, accuracy, F1 are defined as:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F_1 = \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

In signal detection theory, a Receiver Operating Characteristic (ROC) curve is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold

is varied. It is created by plotting the fraction of true positives out of the positives (True positive rate) v.s. the fraction of false positives out of the negatives (False positive rate), at various threshold settings.

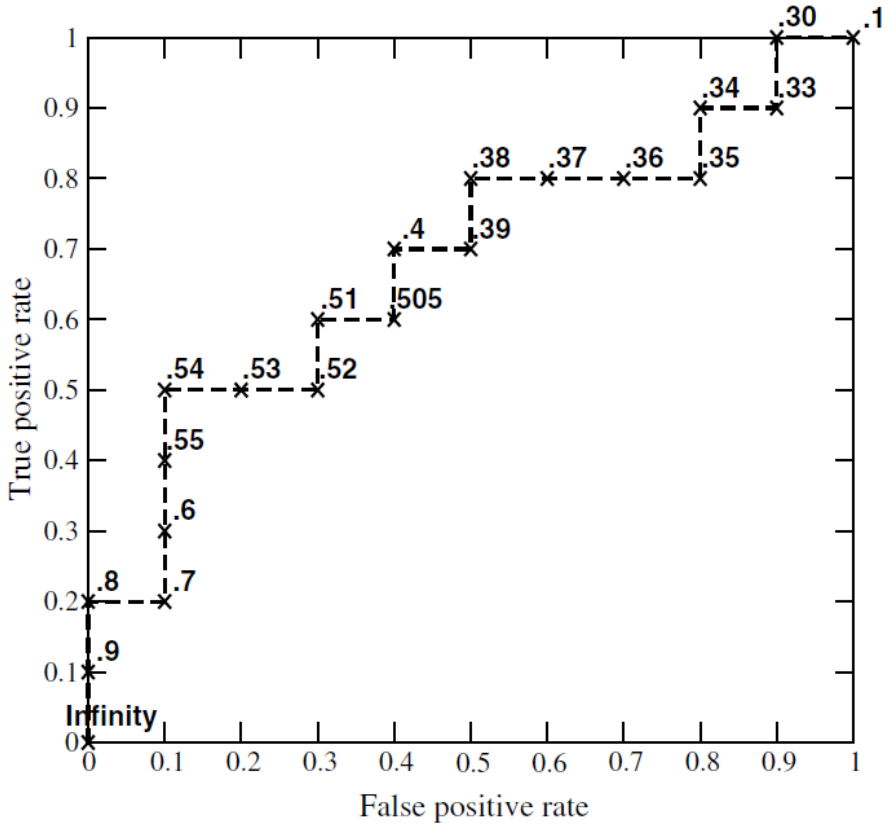

Figure 5. ROC Curve and AUC

(From https://ccrma.stanford.edu/workshops/mir2009/references/ROCintro.pdf)

AUC is Area Under the (ROC) Curve. The AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. In practice the AUC performs very well and is often used when a general measure of predictiveness is desired.

4.2. Data Set and Positive/Negative Instance Number

Ralaivola *et al.*, Schietgat *et al.*, and we use the same NCI data set. We use the same 10-fold cross-validation datasets as Schietgat *et al.*'s. Ralaivola *et al.* used 80% as training set and 20% as testing set. Schietgat *et al.* and we used 90% as training set and 10% as testing set.

10-fold cross-validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on training set, and validating the analysis on the testing set. To reduce variability, 10 rounds of cross-validation are performed using different partitions, and the validation results are averaged over the 10 rounds.

Since Schietgat *et al.* only provided results for 5 data sets; we choose the same 5 data sets to evaluate our methods. Table 2 shows the positive and negative instance numbers in each of the 10-fold cross-validation datasets.

Table 2. Data sets

+/-	786-0	HCT-116	M14	NCI-H522	SNB-19
Test0	184/168	205/168	182/174	214/144	184/189
Test1	184/168	205/168	182/174	214/144	184/189
Test2	183/168	205/168	182/174	214/144	184/189
Test3	183/168	205/167	182/174	214/144	184/189
Test4	183/167	205/167	182/174	214/144	184/189
Test5	183/167	205/167	181/174	214/143	184/188
Test6	183/167	205/167	181/173	214/143	184/188
Test7	183/167	205/167	181/173	214/143	184/188
Test8	183/167	205/167	181/173	213/143	184/188
Test9	183/167	204/167	181/173	213/143	184/188
Total	1832/1674	2049/1674	1815/1736	2138/1435	1840/1885

There is no overlapping between testing sets. In every fold, one testing set is selected; the other 9 sets are used as training set. We can get the AUC, F1, and other measures for every fold. After 10 folds, we can get the mean of 10 AUC, F1 values and their standard deviation.

4.3. Single Model Result

We utilize 5 cancer drug sets: 786-0, HCT-116, M14, NCI-H522, and SNB-19. We use different methods to compare compound similarity, and based on these global and local similarity to predict compound functions. A single model is an algorithm and using a type of features.

Min/Max Tanimoto kernel and Tanimoto kernel use SVM as the classification method and use global similarity information as their features.

Other single models use local similarity. We use a 0/1 matrix of 6400 MCSs as local similarity features. These single models include: Random Forest, Gradient Boosted Regression Trees, SVM (MCSs), Elasticnet, Generalized Linear Model, Artificial Neural Networks, and k-Nearest Neighbor. The AUC of all single models are show in Table 3.

Table 3. AUC (%) comparison of single models

(From high to low. Two models using global similarity have gray background; other seven models use local similarity)

Model	786-0	HCT-116	M14	NCI-H522	SNB-19
Random Forest	80.84±1.94	80.94±2.07	79.56±1.63	79.31±1.60	78.92±1.44
Gradient Boosted Regression Trees	79.13±2.16	80.50±1.86	77.47±1.94	78.04±3.55	76.65±1.26
SVM (MCSs)	78.75±2.26	80.12±2.26	77.80±2.01	77.89±2.85	76.52±1.64
Elasticnet	77.47±1.95	79.42±1.90	76.98±2.11	77.32±4.08	76.00±1.08
Generalized Linear Model	77.28±2.72	78.66±2.11	75.66±1.98	75.96±2.96	75.52±2.09
Min/Max Tanimoto kernel	77.04±2.60	77.63±1.51	76.93±2.74	76.55±2.08	74.50±2.81
Tanimoto kernel	74.57±2.59	72.88±1.45	75.47±2.56	75.08±1.98	73.47±1.48
Artificial Neural Networks	70.29±2.46	72.02±3.66	69.49±2.07	70.71±4.54	67.81±3.57
k-Nearest Neighbor	54.53±2.53	54.16±3.36	51.15±3.33	54.40±3.45	53.42±2.63

4.4. Ensemble Result

We also tried different ensemble methods. The simplest way of ensemble is to use mean and median of individual methods as the ensemble result. We used mean and median as baselines to compare with other complicate ensemble methods. Random Forest is the best single model. The mean of 9 models is as good as Random Forest. The median of 9 models is better than Random Forest in every data set.

The Subspace Discriminant ensemble of 5 models is far better than median. But after adding more models, the Subspace Discriminant ensemble of 5 models is almost the same. Subspace Discriminant ensemble use the discriminant values on training sets in every fold as its full data set, and make a prediction use the discriminant values on testing sets in every fold.

Lastly, we use a weighted linear ensemble. Every model is set a weight w_i. w_i is a real number between -1 and 1, $\sum_{i \in models} w_i = 1$.

We define $d_{i,j}$ as the discriminant value of instance j in model i. d_j is the instance j ensemble discriminant value of several models. We have:

$$d_j = \sum_{i \in models} w_i \, d_{i,j}$$

We use PSO to find the best weight set. To avoid over fitting, we use the same weight set for all 5 data sets. If every data set use different weight set, the AUC result should be higher. PSO use the discriminant values on testing sets in every fold. All ensemble results are show in Table 4.

Table 4. AUC (%) comparison

(Compare different ensemble methods and the best single model. Mean and median are used as baselines)

Type	786-0	HCT-116	M14	NCI-H522	SNB-19
Random Forest	80.84±1.94	80.94±2.07	79.56±1.63	79.31±1.60	78.92±1.44
mean(9 models)	80.93±2.04	81.55±2.33	78.95±1.96	79.71±2.87	78.39±1.40
median(9 models)	81.45±1.96	82.24±2.02	80.10±2.11	80.75±3.15	79.30±1.36
Subspace Discriminant (5 models)	82.81±2.19	82.99±1.74	81.92±1.95	81.53±2.07	80.30±1.76
Subspace Discriminant (9 models)	82.73±2.20	83.31±1.95	81.67±2.03	81.64±2.67	80.20±1.51
Weighted liner blending 1(10 models)	83.23±2.04	83.49±1.78	82.12±2.05	82.11±2.29	80.88±1.44

Subspace Discriminant ensemble using 5 models includes: (1) Min/Max Tanimoto kernel, (2) Tanimoto kernel, (3) sd2gram kernel, (4) Random Forest, and (5) SVM (MCSs). sd2gram is a graph kernel. Ensembles using 9 models include 4 more: (6) Generalized Linear Model, (7) Elasticnet, (8) k-Nearest Neighbor and (9) Artificial Neural Networks. Ensemble using 10 models includes 1 more: (10) median of model 1-9.

Since PSO use the discriminant values on testing sets in every fold and make a prediction using testing sets discriminant values; it may introducing bias and causing over fitting. So we use the discriminant values on testing sets in 3 data sets as training set, and use other 2 data sets as testing set. These 3 training data sets are: 786-0, M14, and SNB-19. We compare the result of

two weighted linear blending with another best ensemble method, Subspace Discriminant, in Table 5.

Table 5. AUC (%) comparison of different ensembles on testing set

Ensemble	HCT-116	NCI-H522
Subspace Discriminant (9 models)	83.31±1.95	81.64±2.67
Weigh2 (786-0 M14 SNB-19 as training, 10 models)	83.37±1.80	81.98±2.19
Weight1(10 models)	83.49±1.78	82.11±2.29

The AUC values of Weigh2 are slightly lower than those of using Weight1. But they are still better than Subspace Discriminant ensemble. It indicated that our weighted linear ensemble is a better ensemble method.

We compare our best results, Weight1, with that of two previous studies. Ralaivola *et al.* reported their results in 2005 [Ralaivola 2005]. They used Min/Max Tanimoto kernel and Tanimoto kernel. Min/Max Tanimoto kernel was better than Tanimoto kernel in every screen. Schietgat *et al.* reported their results on same data sets in 2010[Schietgat 2010].They tried different MCS numbers and their best result was using 6400 MCSs. Schietgat *et al.* used a combination of Tanimoto kernel and MCSs. They only used SVM classifier. We only compared our results with their best result (Table 6). Our ensemble result was better than theirs in every screen.

Table 6. AUC (%) comparison of previous works and ours

Cancer	2005 Ralaivola *et al.* Min/Max Tanimoto kernel d=10	2010 Schietgat *et al.* 6400 MCSs & Tanimoto kernel	Zhu *et al.* ensemble
786-0	79.41±0.89	80.7	83.23±2.04
HCT-116	80.29±1.18	82.0	83.49±1.78
M14	79.73±1.91	80.2	82.12±2.05
NCI-H522	79.50±1.71	80.3	82.11±2.29
SNB-19	79.40±1.04	79.4	80.88±1.44

Our results use Weight1 linear blending.

We listed our 2 weight sets of weighted liner blending 1 and weighted liner blending 2 (Table 7). The same set are used in all data sets. If every data set use its own weight set, the result should be better.

Table 7.Weight sets of linear blendings

Model	Weight1	Weight2
Min/Max Tanimoto kernel	0.3398	0.3205
sd2gram	0.2556	0.2622
Random Forest	0.1156	0.1381
SVM (MCSs)	0.0894	0.0623
Tanimoto kernel	0.0838	0.1583
median(9 models)	0.0555	-0.0334
Generalized Linear Model	0.0286	0.0293
Elasticnet	0.0246	0.0545
k-Nearest Neighbor	0.0121	0.0117
Artificial Neural Networks	-0.0050	-0.0034

4.5. Training Time Comparison

Lastly, we compared the training time, AUC, F1 *etc.* of two Random Forest classifications. The first one used all 6400 MCSs features. The second one used only 337 MCS features. We can see the second result is a little better and the training time is much shorter. It is because many MCSs are useless. They are an atom, a hydrocarbon chain or a benzene ring, *etc.* They occur in almost every compound. Filter out these MCSs improve the result and save a lot of time. So we can try more models. The training time and classification results are show in Table 8.

Table 8. Training time comparison of Random Forest without feature selection and after it

model	Without feature selection	After feature selection
Feature number	6400	337
Training time	67:04:25	00:05:14
AUC (%)	79.65±2.24	80.84±1.94
F1 (%)	74.27±1.98	75.39±1.67
Accuracy (%)	72.39±1.82	73.67±1.46
Precision (%)	76.37±3.38	77.30±3.59
Recall (%)	72.35±1.47	73.69±1.66

Feature selection helps us saving a lot of time and makes us can do more experiments.

CHAPTER 5. DISCUSSION AND CONCLUSION

5.1. Discussion

Combine global information and local similarity, their result outperformed individual graph kernels use global similarity. We not only use both global and local information, but also use a library of classification models. Our result is better than theirs because they only use SVM.

5.2. Future Work

Next thing we need to do is finishing all 60 screen classifications. We'll add a lot of models. We only used the best $d = 10$ in Tanimoto and Min/Max Tanimoto kernel. A smaller d may provide more useful information. We may try not use a fixed depth, but a d from 1 to a small number. Many graph kernels are not as good as Tanimoto and Min/Max Tanimoto kernel. We'll add these kernels as new models. Many algorithms can be tuned. For example, we only use the default parameters of Artificial Neural Network. We may tune these algorithms. There are some other classification algorithm, such as Stochastic Gradient Descent, can be add too. Our object is using about 100 models, and then selecting a subset of them uses Genetic Algorithm [Bottou 2004] *etc.* Lastly, make an ensemble on this subset.

5.3. Conclusion

In this paper we described an ensemble of many algorithms. The result is better than previous works. These classification algorithms and ensemble can be used in many areas.

REFERENCES

[Ballester 2007]: Pedro J. Ballester and W. Graham Richards, *Ultrafast Shape Recognition to Search Compound Databases for Similar Molecular Shapes*, Journal of Computational Chemistry, 2007

[Bottou 2004]: Leon Bottou, *Stochastic Learning*, Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence, 2004

[Cao 2008]: Yiqun Cao, Tao Jiang and Thomas Girke, *A maximum common substructure-based algorithm for searching and predicting drug-like compounds*, Bioinformatics, 2008

[Cao 2010]: Yiqun Cao, Tao Jiang and Thomas Girke, *Accelerated similarity searching and clustering of large compound sets by geometric embedding and locality sensitive hashing*, Bioinformatics, 2010

[Eckert 2007]: Hanna Eckert and Jurgen Bajorath, *Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches*, Drug Discovery Today, 2007

[Ferreira 2010]: Joao D. Ferreira and Francisco M. Couto, *Semantic Similarity for Automatic Classification of Chemical Compounds*, PLoS computational biology, 2010

[Gartner 2003]: Thomas Gartner, *A Survey of Kernels for Structured Data*, ACM SIGKDD Explorations Newsletter, 2003

[Gartner& Flach 2003]: Thomas Gartner, Peter Flach, and Stefan Wrobel, *On Graph Kernels: Hardness Results and Efficient Alternatives*, 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003,

[Hariharan 2011]: Ramesh Hariharan, Anand Janakiraman, Ramaswamy Nilakantan, Bhupender Singh, Sajith Varghese, Gregory Landrum, and Ansgar Schuffenhauer, *MultiMCS: A Fast Algorithm for the Maximum Common Substructure Problem on Multiple Molecules*, Journal of Chemical Information and Modeling, 2011, 51

[Holliday 2011]: John D Holliday, Evangelos Kanoulas, Nurul Malim and Peter Willett, *Multiple search methods for similarity-based virtual screening: analysis of search overlap and precision*, Journal of Cheminformatics, 2011

[Kursa 2010]: Miron B. Kursa and Witold R. Rudnicki, *Feature Selection with the Boruta Package*, Journal of Statistical Software, 2010

[Leslie 2002]: Christina Leslie, Eleazar Eskin, William Stafford Noble, *The spectrum kernel: a string kernel for SVM protein classification*, Pacific Symposium on Biocomputing, 2002

[Mahe 2005]: Pierre Mahe, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert, *Graph Kernels for Molecular Structure-Activity Relationship Analysis with Support Vector Machines*, Journal of Chemical Information and Modeling, 2005, 45

[Mahe 2006]: Pierre Mahe, Liva Ralaivola, Veronique Stoven, and Jean-Philippe Vert, *The pharmacophore kernel for virtual screening with support vector machines*, Journal of Chemical Information and Modeling, 2006, 46

[Mahe 2009]: Pierre Mahe and Jean-Philippe Vert, *Graph kernels based on tree patterns for molecules*, Machine Learning, 2009

[NCI 2003]: http://cactus.nci.nih.gov/download/nci/

[NCI 2012]: http://www.dtp.nci.nih.gov/docs/cancer/cancer_data.html

[Perret 2007]: Jean-Luc Perret, Pierre Mahe, and Jean-Philippe Vert, ChemCpp , http://chemcpp.sourceforge.net

[Ralaivola 2005]: Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi, *Graph Kernels for Chemical Informatics*, Neural Networks, 2005

[Ramon 2003]: Jan Ramon and Thomas Gartner, *Expressivity versus Efficiency of Graph Kernels*, Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences, 2003

[Schietgat 2010]: Leander Schietgat, Fabrizio Costa, Jan Ramon, and Luc De Raedt, *Effective feature construction by maximum common subgraph sampling*, Machine Learning, 2011, 83

[Schietgat 2013]: PMCSFG, http://dtai.cs.kuleuven.be/ml/systems/pmcsfg

[Swamidass 2005]: S. Joshua Swamidass, Jonathan Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola and Pierre Baldi, *Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity*, Bioinformatics, 2005

[Wang 2010]: Xiaohong Wang, Jun Huan, Aaron Smalter, and Gerald H Lushington, *Application of kernel functions for accurate similarity search in large chemical databases*, BMC Bioinformatics, 2010

[Wong 2011]: William W.L. Wong and Forbes J. Burkowski, *Using Kernel Alignment to Select Features of Molecular Descriptors in a QSAR Study*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2011, 8