# Rare-class Classification using Ensembles of Subsets of Variables

by

Jabed H Tomal

M.Sc., University of Windsor, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies

(Statistics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2013

# Abstract

An *ensemble of classifiers* is proposed for predictive ranking of the observations in a dataset so that the rare class observations are found in the top of the ranked list. Four *drug-discovery* bioassay datasets, containing a few active and majority inactive chemical compounds, are used in this thesis. The compounds' activity status serves as the *response variable* while a set of descriptors, describing the structures of chemical compounds, serve as *predictors*. Five separate descriptor sets are used in each assay. The proposed ensemble aggregates over the descriptor sets by averaging probabilities of activity from random forests applied to the five descriptor sets. The resulting ensemble ensures better predictive ranking than the most accurate random forest applied to a single descriptor set.

Motivated from the results of the ensemble of descriptor sets, an algorithm is developed to uncover *data-adaptive subsets of variables* (we call phalanxes) in a variable rich descriptor set. Capitalizing on the richness of variables, the algorithm looks for the sets of predictors that work well together in a classifier. The data-adaptive *phalanxes* are so formed that they help each other while forming an ensemble. The phalanxes are aggregated by averaging probabilities of activity from random forests applied to the phalanxes. The *ensemble of phalanxes* (EPX) outperforms random forests and regularized random forests in terms of predictive ranking. In general, EPX performs very well in a descriptor set with many variables, and in a bioassay containing a few active compounds.

The phalanxes are also aggregated within and across the descriptor sets. In all of the four bioassays, the resulting ensemble outperforms the ensemble of descriptor sets, and random forests applied to the pool of the five descriptor sets.

The ensemble of phalanxes is also adapted to a *logistic regression model* and applied to the *protein homology* dataset downloaded from the KDD Cup 2004 competition. The ensembles are applied to a real test set. The adapted version of the ensemble is found more powerful in terms of predictive ranking and less computationally demanding than the original ensemble of phalanxes with random forests.

# Preface

This thesis is written up under the supervision of my advisors Dr. William J. Welch and Dr. Ruben H. Zamar. The research questions, developed methods and the analyses were thoroughly discussed in our weekly research meeting. For all the Chapters of this thesis, the computations, analyses, first-drafts and follow-up revisions were done by me. Dr. Welch and Dr. Zamar spent a huge amount of time checking the results and improving the writing to make this thesis better.

A *provisional patent application* is submitted based on the work in Chapter 3: Jabed H. Tomal, William J. Welch and Ruben H. Zamar (UBC Ref: 14-019 US Prov, 2013), Ensembling Classification Models Based on Phalanxes of Variables with Applications in Drug Discovery. *US Patent & Trademark, No. 61/832,384.* In this Chapter, I developed the research question and most of the steps of the Algorithm 3.1 to uncover data-adaptive subsets of variables in a variable-rich dataset. The work is currently under revision to submit for possible publication.

# Table of Contents

## Appendices

# List of Tables

# List of Figures

# List of Algorithms

# Glossary

**RF** Random Forests

**RRF** Regularized Random Forests

**EDS** Ensemble of Descriptor Sets

**EPX** Ensemble of Phalanxes

**LR** Logistic Regression Model

**AHR** Average Hit Rate

**IE** Initial Enhancement

**APR** Average Precision

**RKL** Rank Last

**AP** Atom Pairs

**BN** Burden Numbers

**CAP** Carhart Atom Pairs

**FP** Fragment Pairs

**PH** Pharmacophores

**SNP** Single Nucleotide Polymorphism

**KDD** Knowledge Discovery and Data Mining

# Acknowledgements

My sincere gratefulness goes to my supervisors Professor William J. Welch and Professor Ruben H. Zamar for their thoughtful guidance, remarkable insights and continuous encouragement throughout this research. I have acquired a handful of good qualities from them, especially inquisitiveness, abilities to think hard and skills for academic writing. This thesis would not be possible to finish without their scholarly support.

I thank Dr. Alexandre Bouchard-Côté for his time and effort in serving in the supervisory committee, reviewing the thesis and providing thoughtful comments.

My thanks go to Dr. S. Stanley Young for his prompt help to generate the descriptor sets for the bioassay datasets.

I also want to thank Professors Raymond Ng and Gabriela Cohen Freue for serving as the university examiners, and Professor Marianthi Markatou for serving as the external examiner.

I want to convey my sincere gratitude to the faculty and administrative staff of the department of statistics. I have learned a lot taking courses with Professor John Petkau and Professor Paul Gustafson. My cordial thanks are for Peggy Ng, Andrea Sollberger, Elaine Salameh, and Viena Tran for their supportive and prompt administrative help.

Special thanks are for my friends in the department for their warm support during my study. Andy Leung helped me debugging and running a portion of C codes used in my thesis. Camila P. Estevam de Souza was always supportive with her big smile throughout my graduate life. Thank you very much!

The warmest thanks and gratefulness are for my wife, Shafinaz Shafique, for her support, care and affection throughout my study. My son, Tashreeq, was always with me to bring happiness in my life. My sincere gratefulness goes to parents who were and are always in my heart.

This thesis has been written up with the generous financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada, to which I want to express my sincere thankfulness.

# Dedication

***To My Family***

# Chapter 1

# Introduction

The theory of classification has received considerable attention in statistics and in machine learning. A classifier places objects into different classes based on the characteristics of the objects. In this thesis, I will be dealing with supervised classification only where a classifier is learned on a dataset with objects of known classes and the learned classifier is then used to predict the classes of test objects. The dataset with objects of known classes, where a classifier is trained or learned, is known as training or learning dataset and the dataset for which objects' predictions are required is known as test dataset.

Suppose we wish to classify an incoming email as "Spam" or "Non-spam" using features of the email like: email ID, subject, length, percentage of specific words etc. (DeBarr and Wechsler, 2012). We build a classifier on previously received emails, already filtered as "Spam" or "Non-spam", and classify incoming emails based on their features. The collection of received emails forms the training or learning data and the collection of incoming emails forms the test data.

The application of statistical classification includes all disciplines of science and business. Machine learners used classification methods in robotics, internet search engines, image/face/speech recognitions, computer games and in many other systems. However, the goal of my research is to identify relevant rare class objects from a large collection of objects. The reason why an object is considered relevant depends on the context of a specific problem. For example, in terrorist detection the relevant objects are individual communications deemed very suspicious (Fienberg, 2004); in credit card fraud detection the relevant objects are individual transactions that are fraudulent (Srivastava et al., 2008); in spam detection the relevant objects are individual emails that are highly likely to be spam (DeBarr and Wechsler, 2012); in drug discovery the relevant objects are chemical compounds that are active against a specific biological target (Warmuth et al., 2001), for example, the human immunodeficiency virus (HIV).

The goal is to rank the rare-class objects ahead of the majority class objects. The objects in a test set are ranked using the probabilities of belonging to the rare-class from a classifier learned on a training set. The top ranked objects may be shortlisted for further investigation. The shortlist is intended to include most, if not all, of the rare-class objects in the test set. The length of the shortlist will depend on the available resources in a study.

Although this rare class ranking problem is similar to those encountered in the two-class classification problem, the underlying objective is different. In particular, automatic classifi-

cation is of little interest. This is because further investigation of the top-ranked candidates is almost always necessary. For example, in the fraud detection application, one seldom terminates a credit card account without confirming the suspected fraud (Bolton and Hand, 2002). Therefore, we are most interested in producing an effective ranking of all of the candidates so that any further investigation is least likely to be carried out in vain.

With the advancement of computer speed, memory and different softwares, the number of variables in recent statistical datasets has become very large. For example, the datasets in drug discovery and in computational chemistry contain a few hundreds to thousands of variables. The analysis of such large datasets poses two types of problems: (i) large computational time and (ii) poor model building in terms of low prediction power and high complexity of the resulting model. The large computational time incurred by a model is usually handled by developing fast computational tools or algorithms for fitting the model. The poor model building strategy is usually solved by filtering or selecting variables or by regularizing the model. The problems relating to the large number of variables is popularly known as the "curse-of-dimensionality". Instead of repeating the conventional word "curse-of-dimensionality", we term this aspect of data as "rich-in-variables" as a variable-rich dataset may possess many signals for modelling the response.

Ensembling is a fairly novel method of improving prediction accuracy of a classifier. In an ensemble, a collection of classifiers is constructed and the class membership of an object is determined by aggregating individual classifiers. Ensemble methods have been shown to be more accurate than non-ensemble methods and insensitive to irrelevant feature variables (e.g., Breiman, 1996a, 2001; Freund and Schapire, 1996a; Wang, 2005). Capitalizing on the "richness-of-variables" in a dataset, I develop an ensemble method which outperforms existing off-the-shelf ensembles in terms of predictive ranking of the rare class objects.

## 1.1 Objectives

The objectives of this thesis are:

1. Placing rare class objects before the majority class objects in highly unbalanced two-class problems.

2. Developing an ensemble which capitalizes on the "richness-of-variables" in a dataset to outperform existing ensembles.

## 1.2 Some Classifiers

In this section, I will briefly describe some popular non-ensemble classifiers. The first two were the top performing non-ensembles in two-class ranking problems (Hughes-Oliver et al., 2011; Wang, 2005).

### 1.2.1 Classification Tree

A classification tree is a binary decision tree constructed by splitting a node into two child nodes (descendants) repeatedly, beginning with the root node that contains the whole training sample. The tree building procedure partitions the space of the feature variables successively into smaller hyper-rectangles (nodes) so that the class variable in each hyper-rectangle is as homogeneous as possible.

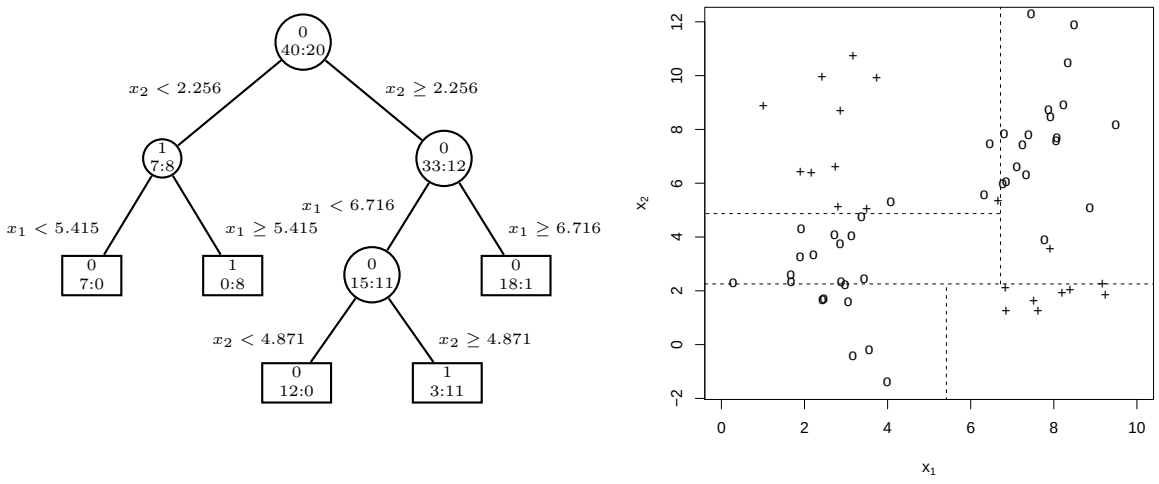
At a node, the tree growing process chooses a split from all possible splits so that the resulting left and right descendants are as pure as possible. If all of the objects of a node belong to a particular class, the node is called pure. The tree growing algorithm first finds each feature's best split and then finds the node's best split choosing one that maximizes a splitting criterion. The process then splits the node using the node's best split if stopping rules are not satisfied.

An example will make our understanding clear. For this, I used a toy dataset of size 60 with two classes (1 and 0) and two feature variables, $x_1$ and $x_2$. The observations are generated as follows. Let $\boldsymbol{\mu}_1 = \{2.5, 2.5\}$, $\boldsymbol{\mu}_2 = \{7.5, 7.5\}$, $\boldsymbol{\mu}_3 = \{2.5, 7.5\}$, $\boldsymbol{\mu}_4 = \{7.5, 2.5\}$ be four mean vectors and $\Sigma = \begin{pmatrix} 1.00 & 0.15 \\ 0.15 & 4.00 \end{pmatrix}$ be the common variance-covariance matrix. A total of 40 class 0 observations are generated from the mixture of two bivariate normal distributions $\alpha\ N(\boldsymbol{\mu}_1, \Sigma) + (1-\alpha)\ N(\boldsymbol{\mu}_2, \Sigma)$, where $\alpha = 0.5$ is the mixture parameter. The other 20 class 1 observations are generated independently from the following distribution $\alpha\ N(\boldsymbol{\mu}_3, \Sigma) + (1-\alpha)\ N(\boldsymbol{\mu}_4, \Sigma)$. Unlike the rare class problem, the classes are kept only roughly unbalanced so that the tree partitioning method can be explained better.

The left panel of Figure 1.1 shows a typical classification tree grown on the simulated dataset. The root node, which contains all 60 cases, is split using the feature variable $x_2$ into two child nodes: $x_2 < 2.256$ (left descendant) and $x_2 \geq 2.256$ (right descendant). The left descendant is an internal node, an internal node is represented by a circle, which contains 15 cases with 7 cases from class 0 and 8 cases from class 1. Thus, the predicted class for this node is 1 with probability 8/15. This left internal node is further split using the feature variable $x_1$ into two child nodes: $x_1 < 5.415$ (left descendant) and $x_1 \geq 5.415$ (right descendant). Both descendants are terminal nodes, represented by a rectangle, with predicted classes 0 and 1, respectively, each with probability 1.

The right descendant of the root node is split using the feature variable $x_1$ into two child nodes: $x_1 < 6.716$ (left descendant) and $x_1 \geq 6.716$ (right descendant). The right descendant is a terminal node with predicted class and probability 0 and 18/19, respectively. The left descendant, an internal node, is further split using the feature variable $x_2$ into two terminal nodes: $x_2 < 4.871$ (left terminal) and $x_2 \geq 4.871$ (right terminal). If a test case is sent down this tree and it reaches the right-most terminal node, the case will be predicted as from class 0 with probability 18/19.

The resulting hyper-rectangles obtained by splitting the two-dimensional space are dis-




(a) (b)

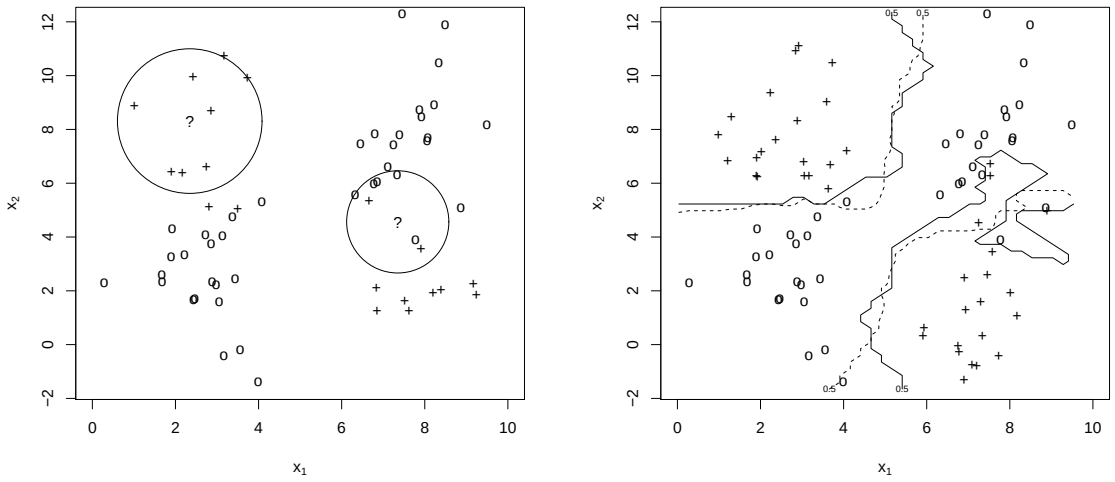
Figure 1.1: (a) An example of classification tree grown from a simulated dataset of size 60 with two classes and two feature variables. (b) The corresponding partitions of the two-dimensional space into hyper-rectangles. Class 0 and class 1 cases are denoted by "o" and "+", respectively.

played in the right panel of Figure 1.1. It is evident that the space partitioning process of a tree allows the effect of one feature variable to be different at various values of other feature variables, and includes non-linear effect of features.

The tree building algorithm uses one of many splitting criteria to determine how homogeneous (i.e., pure) the response variable is in a child node: deviance, entropy, Gini index, and misclassification error. Among them, deviance, entropy, Gini index are mainly used to grow a tree and misclassification error is usually used to prune a tree. The rules that decide when to stop growing a tree are called "stopping rules". When growing a tree, a node will not be split (i) if the node becomes pure, (ii) if all cases have identical values for each feature variable, (iii) if the tree size reaches the user-specified maximum tree size, (iv) if the size of the node is smaller than the user-specified minimum node size, (v) if the child node size is smaller than the user-specified minimum child node size, and (vi) if the improvement of *information gain* from the best split of the node is smaller than the user-specified minimum improvement. For further reference, please see Breiman, Friedman, Olshen, and Stone (1984) and Chapter 9 of Venables and Ripley (2002).

### 1.2.2 *k*-Nearest Neighbours

The $k$-nearest neighbours (KNN) classifier assigns a class to an object based on the majority class of that object's $k$ nearest neighbours. The value of $k$, the number of nearest neighbours, is usually chosen by the user. Hall, Park, and Samworth (2008) study the effect of the value




(a) (b)

Figure 1.2: (a) The 7-nearest neighbours for two test objects (?) using Euclidean distance for the simulated dataset. (b) Two decision boundaries using $k$-nearest neighbours: the solid line is for $k = 1$ and the dashed line is for $k = 7$. Class 0 and class 1 cases are denoted by "o" and "+", respectively.

of $k$ on the misclassification error rate and present a number of methods for selecting the value of this parameter. Often Euclidean distance is used to define the neighbourhood for continuous features. The pioneers of KNN are Cover and Hart (1967), who investigated the properties of 1-NN, the rule that uses one nearest neighbour. If there are ties, the predicted class is determined by random mechanism.

The algorithm for KNN is very simple. For example, we want to predict classes of the two test objects (2.34, 8.31) and (7.35, 4.56), using the simulated dataset (used in Section 1.2.1) of size 60. The left panel of Figure 1.2 shows two 7-nearest neighbourhoods of the test objects. In the neighbourhood of the object (2.34, 8.31), we have seven training observations, all are from class 1. Hence, this test object will be predicted as class 1 with probability 1. The neighbourhood of the second test object (7.35, 4.56) contains 7 training observations: five from class 0 and two from class 1. Thus this object will be classified as class 0 with probability 5/7.

The right panel of Figure 1.2 contains two decision boundaries, one for $k = 1$ (the solid line) and the other for $k = 7$ (the dashed line). It is evident that the decision boundary for $k = 1$ is more irregular than $k = 7$. Seemingly, the classifier with $k = 1$ overfits the data and captures noise variability. For low values of $k$ we have low bias and high variance in prediction, and vice versa. The balance between bias and variance is usually struck by choosing $k$ via cross validation.

### 1.2.3 Discriminant Analysis

In discriminant analysis, the joint distributions of $D$ dimensional feature variables for different classes are usually assumed multivariate normal with class-specific mean vectors and covariance matrices, and the posterior class probabilities conditional on features are determined by `Bayes` rule. The unknown parameters (such as class prior probabilities, mean vectors and covariance matrices) are usually computed from the training sample. The assumption of equal covariance matrices results in linear discriminant analysis (LDA) (Fisher, 1936); whereas the assumption of unequal covariance matrices results in quadratic discriminant analysis (QDA). LDA works well when the feature space is linearly separable for classes, and QDA works well when the decision boundary is quadratic in the feature space. Friedman (1989) proposed a compromise between LDA and QDA, which allows shrinking of separate covariance matrices in QDA toward a common covariance as in LDA by introducing a regularization parameter. This method is known as regularized discriminant analysis (RDA). In some applications mixture of Gaussian distributions is a good choice for the joint distributions of features, which generates a classification method known as mixture discriminant analysis (MDA) (Hastie, Tibshirani, and Friedman, 2009, Section 6.8). On the other hand, nonparametric discriminant analysis (Hastie, Tibshirani, and Friedman, 2009, Section 6.6.2) uses *kernel* density approach to determine the joint distribution of features in each class.

LDA is one of the oldest classification methods, but often found effective especially when the decision boundary is linearly separable as it generates discriminant functions which are linear combinations of features. The assumption of multivariate normal is not a requirement, but it is helpful in determining cut-offs of the discriminant functions to help separating objects in the two class problem. In a multi-class problem, the prediction for a new object can be made by computing posterior probabilities for each class or by determining optimal discriminant functions. Suppose we have $c$ classes and $s = \min(c-1, D)$ discriminant functions $\mathbf{w}_j^T\mathbf{x}$, where $\mathbf{w}_j$ is the vector of weights for the $j$th discriminant function, and $\mathbf{x}$ is the $D$ dimensional vector of feature variables. A new object with vector of features $\mathbf{x}^{new}$ will be assigned to class $c$ if

$$\sum_{j=1}^{s}\left[\mathbf{w}_j^T\left(\mathbf{x}^{new}-\bar{\mathbf{x}}_c\right)\right]^2 \leq \sum_{j=1}^{s}\left[\mathbf{w}_j^T\left(\mathbf{x}^{new}-\bar{\mathbf{x}}_i\right)\right]^2;\ \forall i \neq c, \tag{1.1}$$

where $\bar{\mathbf{x}}_i$ is the mean vector of $\mathbf{x}$'s for class $i = 1, \cdots, c$. In words, the object $\mathbf{x}^{new}$ will be classified as from class $c$ if the transformed space (by common covariance matrix) of $\mathbf{x}^{new}$ is closest to the centroid of class $c$.

### 1.2.4 Neural Networks

Neural networks (NN) are very popular in machine learning. The history of NN dates back to McCulloch and Pitts (1943). The motivation goes back to Widrow and Hoff (1960) and Rosenblatt (1962). The idea of neural networks mimics the structure of the human brain, where many biological neurons are interconnected with each other to perform complicated

tasks. In a neural network, each neuron receives linear combination of inputs as derived features, transforms the derived features using a nonlinear function (usually the `sigmoid` function) and passes the result to another neuron. The output function, generally the `softmax` function (Hastie, Tibshirani, and Friedman, 2009, Section 11.3) in classification, allows a final transformation. In general there are several layers of neurons in a network with many units in each layer. The units in the middle of a network, computing the derived features, are called hidden units because the values of the derived features are not directly observed. The terminal neurons are sometimes allowed to receive linear combinations of the raw inputs to skip hidden layer(s) simplifying the interpretation of a network. The structure of a neural network is usually complex, but the central idea is to extract linear combinations of inputs as derived features, and model the class variable as a nonlinear function of these features.

A neural network has unknown parameters, often called weights, and we seek values for them so that the model fits the training data well. For classification, cross-entropy (deviance) is generally used as a measure of fit which is minimized by gradient descent known as back-propagation (Hastie, Tibshirani, and Friedman, 2009, Section 11.4). Typically we don't want the global minimizer of the cross-entropy, as this is likely to be an overfit solution. Instead regularized cross-entropy is used with a penalty term, often called weight decay, which is usually optimized via cross validation. Scaling of the input variables and multiple starting values of the weights are often utilized to achieve optimal weights at the global minimum of the cross entropy. The choice of the number of hidden layers is guided by background knowledge and experimentation. It is better to have many hidden units than too few (see Hastie, Tibshirani, and Friedman, 2009, Section 11.5.4). With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data; with many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.

### 1.2.5 Support Vector Machines

Support vector machine (SVM) is also a popular classifier with application in statistics and in machine learning. SVM was introduced by Boser, Guyon, and Vapnik (1992) and the current standard incarnation was proposed by Cortes and Vapnik (1995). The idea is to separate the space of feature variables so that the classes become as distinct as possible. In a two class problem, SVM separates the feature space by a linear hyperplane, and its parameters are determined using a training sample by a constrained optimization technique. To produce a nonlinear decision boundary, SVM constructs a linear boundary in a large, transformed version of the feature space. Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space. In SVM the points well inside their class boundary do not play a big role in shaping the optimal hyperplane; and this attractive criterion differentiates SVM from LDA. In LDA, the decision boundary is usually determined by the property of the covariance of the class distributions and the positions of class centroids. The support vector machine is usually extended to

multi-class problems by solving many two-class problems. A classifier is built for each pair of classes, and the final classifier is the one that dominates the most. However, SVM is slow in high dimensions and does poorly in the presence of many noise feature variables (Hastie, Tibshirani, and Friedman, 2009, Chapter 12).

### 1.2.6 LAGO

LAGO is a computationally efficient classifier, developed by Zhu, Su, and Chipman (2006), to rank the rare class objects ahead in a two-class problem. LAGO only targets the rare class for detection and saves much computation time for a large database. Whenever the goal is ranking rare objects, a classifier computes posterior class probabilities conditional on feature variables and uses the computed probabilities to rank the target. Zhu, Su, and Chipman (2006) argued that, as far as ranking is concern, posterior class probabilities given the features and the ratio of the distributions of features for the rare class to the background class serve the same purpose, as the former is a monotone transformation of the latter for fixed prior probabilities. In LAGO, the density for the rare class is computed by a *Kernel* approach with mass only on the rare class objects. LAGO considers the distribution of the features for the background class around a rare class object flat (i.e., Uniform) and reciprocal to the bandwidth of the *Kernel* density determined earlier. Finally, the values of the ratio of the distributions of features of the rare class to the background class are used to rank the objects. Zhu, Su, and Chipman (2006) generalize LAGO to multivariate problem using the naive *Bayes* principle, i.e., by multiplying the univariate feature's distributions together as if they are independent to each other. LAGO has a parameter $k$ like KNN and is usually determined adaptively by cross-validation.

## 1.3 Ensembles of Classifiers

An ensemble of classifiers is an aggregated collection of models/classifiers which can be considered as a model/classifier by itself. An ensemble can improve classification accuracy of a non-ensemble classifier. In this section, I briefly present five ensembles: bagging, random forests, regularized random forests, boosting, and ensemble of systematic subsets.

### 1.3.1 Bagging

Breiman (1996a) introduced an ensemble method, called bagging, which aggregates over several tree classifiers constructed on bootstrap samples of the training observations. Bagging is the acronym of bootstrap aggregating. A bootstrap sample is a sample with replacement of equal size of the training observations (Efron and Tibshirani, 1994). The bootstrap samples are used to construct multiple versions of tree predictors and bagging aggregates them into an ensemble. The aggregation is done by majority vote when predicting a class variable and

by averaging over the versions when predicting a numerical outcome. When aggregation is done by majority vote, ties are handled by random mechanism.

The vital element for the success of bagging is the instability of the prediction method used to build constituent classifiers. If perturbation of a training dataset causes significant changes in prediction performances, then bagging can improve accuracy. Breiman (1996b) showed that bagging can push a good but unstable procedure a significant step toward optimality. On the other hand, bagging does not improve performance of a stable classifier. Instability of a classifier is studied by Breiman (1996b), where it is pointed out that neural networks, classification and regression trees, and subset selection in linear regression are unstable, while k-nearest neighbour is stable.

### 1.3.2 Random Forests

In bagging, each tree is built using a bootstrap sample of a training dataset employing all of the feature variables. Breiman (2001) incorporated one extra layer of randomness, random selection of features at each node of a tree, to bagging and named the new ensemble as random forests. In random forests, each tree is grown using a bootstrap sample of training data; and at each node of the tree, the best split is chosen from a random sample of $m_{try}$ variables instead of all feature variables. Algorithm 1.1 shows the steps in learning a random forests classifier using a training set and predicting the test observations. This extra layer of randomness is used to break correlation between classifiers and thus to improve overall accuracy. Each tree is grown to its maximal depth and aggregation is done by majority vote.

**Algorithm 1.1** Random Forests

1. Let the number of training observations and predictor variables be $n$ and $D$, respectively.
2. For $n_{\text{tree}} = 1$ to $M$:
   - Generate a synthetic training set by drawing a bootstrap sample of size $n$ from the original training set.
   - Build a fully grown unpruned classification tree using the synthetic training set.
   - At each node of the tree, randomly choose $m_{\text{try}} = \sqrt{D}$ variables to find the split-point of that node.
3. End For.
4. Predict test observations by aggregating the predictions from the $M$ classification trees using majority vote.

Breiman (2001) showed that the upper bound on prediction error of random forests depends on the ratio of average correlation between classifiers and average strength of all classifiers. Thus, a tighter upper bound of prediction error can be achieved by minimizing average correlation and maximizing average strength of classifiers. The extra layer of randomness helps to lower average correlation; and growing trees to maximal depth helps to increase

average strength. Although growing trees to maximal depth increases prediction variability, aggregation over many trees reduces overall variance and controls overfitting.

A random forests offers built-in estimation of test-set error via the out-of-bag (OOB) samples. Approximately one third of training cases remains out on each bootstrap sample and constitutes the OOB sample. Importance of feature variables can be assessed in random forests. Each feature variable is randomly permuted in the OOB samples and its impact on prediction error is measured; and variables with large changes in prediction error are deemed important. Random forests also computes a proximity matrix by counting how often a pair of points land in the same terminal node. The proximity matrix is useful for outlier detection, clustering, and missing value replacement etc.

The algorithm for random forests is fast, and often faster than growing and pruning a single tree. Random forests has only one tuning parameter $m_{try}$. Breiman suggests to use $m_{try} = \sqrt{D}$, where $D$ is the number of feature variables. To construct a random forests ensemble, I grow 500 trees with tuning parameter $m_{try} = \sqrt{D}$ using the **R** package **randomForest**. Constructing 500 trees in a random forests is a common practice.

### 1.3.3 Regularized Random Forests

Deng and Runger (2012) proposed the regularized version of random forests. The goal is to select high-quality feature subsets without reducing the prediction performance of the ensemble itself.

The only difference between regularized random forests and random forests (Breiman, 2001) is in the tree growing process. A regularized random forests regularizes variables at each node of the trees in the forest. In order to be selected for splitting a node, a variable needs to produce substantially larger 'information gain' than the variables already selected in previous splits. For example, let $F$ be the feature set used in previous splits in a tree. To split a node, the tree growing algorithm avoids selecting a new feature $x_j \notin F$ unless its $gain(x_j)$ is substantially larger than the $\max(gain(x_i))$ for $x_i \in F$. To achieve this goal, the *regularized gain* is calculated as:

$$gain_R(x_j) = \begin{cases} \lambda \times gain(x_j) & x_j \notin F \\ gain(x_i) & x_i \in F \end{cases} \tag{1.2}$$

where $\lambda \in [0, 1]$ is the regularization coefficient. The $gain_R(x_j)$ is used for splitting each non-terminal nodes instead of $gain(x_j)$. In this thesis, I have used the **R** package **RRF** (Deng, 2012) to fit regularized random forests.

### 1.3.4 Boosting

Boosting, pioneered by the work of Schapire (1990), Freund (1995) and Freund and Schapire (1996a,b, 1997), is an ensemble method. The boosting algorithm starts by assigning equal weights to all observations and sequentially builds a series of classifiers by reweighting only

the misclassified observations. It focuses more on the observations misclassified by previous classifiers. At the end, each classifier in the ensemble casts a vote with appropriate weight determined from its misclassification rate. The AdaBoost.M1 (Freund and Schapire, 1996a) algorithm for classification is presented below.

**Algorithm 1.2** AdaBoost.M1

Let we have $n$ observations in the training data $(\mathbf{x}_i, y_i), i = 1, \cdots, n$, where the class variable is $y_i = 1$ for class 1, and $-1$ for class 2 with $\mathbf{x}_i$ be the vector of feature variables.

1. Initialize the observation weights $w_i = 1/n$, $i = 1, \cdots, n$.
2. For $m = 1$ to $M$:
   - Fit a classifier to the training data with weights $w_i$ and get the prediction $C_m(\mathbf{x}) \in \{-1, 1\}$ .
   - Compute weighted misclassification error
   $$\text{err}_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq C_m(\mathbf{x}_i))}{\sum_{i=1}^{n} w_i}.$$
   - Compute weight for the $m$th classifier
   $$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right).$$
   - Increase weights for the misclassified observations
   $$w_i \leftarrow w_i . \exp\left[\alpha_m I\left(y_i \neq C_m(\mathbf{x}_i)\right)\right],$$
   and scaled to sum 1, $\forall i = 1, \cdots, n$.
3. Output aggregated classifier
$$C(\mathbf{x}) = sign\left(\sum_{i=1}^{M} \alpha_m C_m(\mathbf{x})\right).$$

Algorithm 1.2 aims to improve classification accuracy of a weak classifier. However, the weak classifier needs to perform better than a random guess. If the current classifier misclassifies an observation, then this observation will receive more weight in the next iteration. In the final overall ensemble, classifiers that are accurate predictors of the training data receive more weight, whereas, classifiers that are poor predictors receive less weight.

### 1.3.5 Ensemble of Systematic Subsets

The ensemble of systematic subsets (Wang, 2005) is particularly designed to rank objects from a particular class ahead of the other class. Although this idea is intuitive for a two class problem, it can be easily generalized to the multi-class problem, i.e., ranking objects from a particular class ahead of the other classes. For this ensemble, the constituent classifiers are constructed on the subsets of feature variables and aggregation is done by probability

averaging, i.e., averaging probabilities of belonging to the target class from different classifiers built on subsets of variables. The performance of this ensemble would depend on the strength and diversity of the constituent classifiers constructed on the subsets of feature variables.

Consider that there are three feature variables in a dataset: $x_1$, $x_2$ and $x_3$. The set of all possible subsets is $\{x_1, x_2, x_3, (x_1, x_2), (x_1, x_3), (x_2, x_3), (x_1, x_2, x_3)\}$. A total of seven classifiers can be constructed each using a subset of features, and the probability of belonging to the target class is averaged over the seven classifiers. Such average probabilities are used to rank the target objects ahead in the list of all objects. Many ensembles can be constructed based on the orders of the subsets: the systematic subset of order one averages over the first three classifiers, order two averages over the first six classifiers, and order three overages over the seven classifiers.

Wang (2005) employed KNN and classification tree as base learners to construct the ensemble of systematic subsets. When KNN is the base classifier, leave-one-out cross validation was used to choose $k$, the number of nearest neighbours. Although the use of KNN was found encouraging in terms of prediction performance by Wang (2005), it appeared computationally expensive for large data. An increase in the number of feature variables exponentially increases the number of possible subsets making the ensemble even slower.

## 1.4 Variable Selection via Filtering

Some of the classifiers presented in section 1.2 perform variable selection automatically. For example, a tree growing algorithm selects the best split at each node from all possible available splits. Thus, while splitting a node, a classification tree performs variable selection automatically. The other non-ensemble classifiers - LDA, KNN, NN, and SVM – do not possess this attractive feature of automatic variable selection. Among LDA, KNN, NN, and SVM, variable selection is usually performed by penalizing unimportant variables. For example, penalized discriminant analysis uses regularization in disregarding useless and correlated features (Hastie, Tibshirani, and Friedman, 2009, Section 12.6).

Among the ensembles, bagging and random forests possess the automatic variable selection property, as their constituent models are built using tree based methods. Furthermore, both of the ensembles aggregate constituent classifiers using majority vote. If aggregation is done by majority vote, the performance of an ensemble with a few bad classifiers and many good classifiers is not harmed much. A similar statement is also true for the ensemble of systematic subsets. In this ensemble, aggregation is done by probability averaging. A few weak classifiers based on unimportant feature variables cannot do much harm when aggregation is done by probability averaging. But the number of good and weak classifiers plays a big role in determining the performance of the ensemble. If an ensemble contains many weak classifiers then its performance is highly likely to be decreased (Wang, 2005, Section 5.3).

Boosting and regularized random forests perform variable selection automatically as both

of the ensembles use classification tree based algorithm to grow constituent classifiers. The latter ensemble puts an extra effort through regularization to find a compact subset of important features. Thus, in a sense, the regularized random forests is too aggressive in filtering variables.

Often, high dimensional datasets contain many useless or unimportant variables. In the presence of many useless variables, the performances of the predictive models applied to such datasets become weak. This motivates me to perform model selection by filtering unimportant and useless variables. Wang (2005) proposed a permutation test to screen weak or noise classifiers out constructed through irrelevant feature variables. She proposed random permutation of training objects to obtain a distribution of random prediction. The classifiers with cross-validated prediction performances exceeding 95% quantile of the distribution of random prediction are proposed important; else filtered from the ensemble. A similar approach will be adapted in this thesis to filter weak and useless variables.

## 1.5 Drug Discovery Datasets

I have used four PubChem BioAssay datasets (`www.ncbi.nlm.nih.gov/pcassay`). Each dataset provides bioassay screens of chemical substances which are either active or inactive against a biological target. The activity status of a chemical compound is determined through high-throughput screening (HTS). The selected datasets are highly unbalanced: on average the ratio is 1 active compound to 100 inactive compounds. The process of determining activity status of many chemical compounds in a chemical library is costly and time consuming.

Chemists are able to understand the structures of chemical compounds at the atomic level which provides quantitative representation of chemical structures. A set of variables representing the quantitative structure of chemical compounds is also known as a descriptor set. The molecule-based hypothesis is that compounds with similar chemical structures have similar activities (Mezey et al., 2001). This hypothesis leads to the quantitative structure-activity relationship (QSAR) models which relate the biological activity status of chemical compounds with their quantitative structures. The QSAR models can be used to rank the test compounds and the top ranked candidates can be passed through high-throughput screening to determine their activity status. This process of virtual screening is much cheaper and faster than the high-throughput screening of all of the compounds as it is very cheap and fast to generate descriptor sets using computer softwares. There exist many molecular descriptor sets in QSAR studies; (see Leach and Gillet, 2007; Todeschini and Consonni, 2000). I have used five variable-rich descriptor sets for each of the four PubChem Assays. The drug discovery datasets are described in detail in Sections 2.2 and 2.3 of Chapter 2.

## 1.6 Protein Homology Dataset

Protein homology means biological homology between proteins. Two or more proteins are homologous in the sense that they have a common evolutionary origin. Knowing homology status helps scientists to infer the evolutionary sequences of many proteins (Koonin and Galperin, 2003, Chapter 2). The protein homology dataset is downloaded from the 2004 KDD Cup website. The response variable is the homology status between a candidate protein and a native protein. The predictors are the 74 feature variables which are derived from the similarity search between a candidate protein and a native protein. A total of 303 native proteins are considered in this dataset which represent 303 blocks. There are several candidate proteins in each block which are tested for homology against a native protein. The 303 blocks are randomly divided into training and test sets by the KDD Cup organizers. The training set contains a total of 153 blocks for which the homology status is known to us. The test set contains a total of 150 blocks for which the homology status is completely unknown to us. The dataset is highly sparse: on average there are 5 homologous proteins in each block in the training set. The protein homology dataset is described in detail in Section 5.2 of Chapter 5.

I fit an ensemble using the 153 blocks in the training set. The fitted ensemble is used to rank the candidate proteins in each test block so that the rare homologous proteins are found in the top. The estimated probabilities of being homologous are used to rank the candidate proteins in each block. Thus, a performance metric is computed in each test block. The final performance of the ensemble is measured by averaging the performance metric over the 150 test blocks. The method will be described in detail in Chapter 5.

## 1.7 Summary

The goal of this thesis is to rank the rare class objects ahead of the majority class objects in a dataset in such a way that the rare objects are found at the top of the ranked list. We propose ensemble methods to serve this goal. The ensembles are developed focusing on datasets with many variables. Instead of regularization, I use the richness of variables in a dataset to formulate the ensemble. The motivation for developing our ensemble comes from the work described next in Chapter 2.

# Chapter 2

# Ensembling Natural Subsets of Variables

## 2.1 Introduction

In drug discovery, the compounds in a chemical library may be "active" or "inactive" against a biological target. High-throughput screening (HTS) may be used to assay a portion of the compounds to determine their activity status. In quantitative structure-activity relationship (QSAR) studies, it is customary to model the biological activity of compounds by their chemical structures or physiochemical properties known as molecular descriptors (Hughes-Oliver et al., 2011; Merkwirth et al., 2004; Rusinko et al., 1999; Svetnik et al., 2003). In this Chapter, I train a classifier using molecular descriptors of chemical compounds with known activity status. The trained classifier is used to estimate the probability of activity for the test compounds.

The classifiers are applied to four drug discovery bioassays (described in Section 2.2) which contain very small proportions of active compounds compared to the proportion of inactive compounds. Moreover, I am interested in identifying the rare active compounds only. In such problems, a prediction model might perform poorly in classifying the rare active compounds. For example, if the proportion of active compounds is smaller than 1%, a classifier that classifies every compound to be inactive will have high correct classification rate of greater than 99% without being able to correctly classify any active compound. Obviously, this does not serve our purpose at all. Hence, unlike classification, I rank all of the compounds in a dataset using the probabilities of activity so that the rare actives are found at the top of a shortlist of all compounds. Thus, a classifier will be considered good as long as it ranks more actives earlier in the shortlist – no matter whether this model can correctly classify any active or not. As such, if a chemist becomes interested in finding the rare active compounds in a huge chemical library, he/she can go through a shortlist of the top ranked compounds only and can save time and money.

Many non-ensemble classifiers, such as neural networks, recursive partitioning, $k$-nearest neighbours, support vector machines etc., have been successfully applied to QSAR studies (see Aoyama et al., 1990; Doniger and Hofmann, 2002; Kauffman and Jurs, 2001; Rusinko et al., 1999). In this chapter, I will be dealing with ensembles of classifiers. Merkwirth et al. (2004) described the application of ensemble methods to binary classification problems

in QSAR studies. They built ensembles by combining classifiers constructed on random subspace of features using a particular modelling method chosen from $k$-nearest neighbours and support vector machines. Their performances were compared with the non-ensemble $k$ -nearest neighbours, support vector machines, and linear regression model with ridge penalty. The resulting ensembles were found better performing than the non-ensemble classifiers.

Bruce et al. (2007) presented a comparative assessment of several machine learning tools for mining drug data, including support vector machines and decision tree based ensembles: boosting, bagging, and random forests. The authors demonstrated that the ensemble "random forests" can provide consistent improvement in predictive performance over single decision tree in terms of classification accuracy. Svetnik et al. (2003) applied random forests for classification of chemical compounds in QSAR studies. Their analysis demonstrates that random forests is a powerful tool capable of delivering highly accurate performance.

Chen et al. (2004) proposed two techniques to deal with unbalanced classification problem using random forests: one is based on assigning large (small) weight for the misclassification of the minority (majority) class objects, and the other is based on under sampling of the majority class objects while drawing bootstrap samples during training process. Both of the approaches were shown to improve prediction accuracy for the minority class. Zhang et al. (2009) proposed an ensemble of classification trees for QSAR studies by adaptively determining the threshold of probability, instead of using the default 0.5, to classify classes of chemical compounds. In this ensemble, the constituent classifiers were built by sampling 70% of the feature variables and then growing classification trees optimized by pruning method. This ensemble was shown to perform well in highly unbalanced QSAR data. Their works (Chen et al., 2004; Zhang et al., 2009) dealt with the classification of the minority class, whereas, in this chapter, I am interested in early ranking of the minority class objects.

Hughes-Oliver et al. (2011) compared a comprehensive collection of the state-of-the-art QSAR models and five descriptor sets to rank the rare active compounds ahead of the inactive compounds in several highly unbalanced two-class assay datasets. Among the models, there were eleven non-ensemble and one ensemble classifiers. Repeatedly the ensemble random forests with one of the five sets of descriptors appeared as the top performing model. But clearly there was no clear winner among the descriptor sets. For example, in one of the assay datasets, AID362, the performance of random forests using the descriptor sets 'Burden Numbers' and 'Atom Pairs' (the descriptor sets will be described in Section 2.3) was in the top and fourth position from the top, respectively - whereas, in another assay dataset, AID364, the positions of the descriptor sets in terms of performance (using random forests, indeed) swapped.

As noted earlier, the molecular descriptors play an important role in developing a classifier. There exist many molecular descriptor sets in QSAR studies (see Leach and Gillet, 2007; Todeschini and Consonni, 2000). Although the number of available descriptor sets is large, there is no consensus on the types of input descriptors to build uniformly good performing

QSAR model (Zhang et al., 2009). In such a situation, rather than using a single set descriptor, we use several sets to build our model.

Until recently, many researchers have either tried to find a good model/classifier or a good set of descriptors of the chemical compounds in QSAR studies. But the performance of the descriptor sets is target-specific, and we are not there yet to label the best set of descriptors. As such, we have proposed an ensemble method which aggregates over several molecular descriptor sets employing one of the most accurate QSAR classifiers: the random forests. I rank compounds in a dataset by averaging the probabilities of activity from random forests classifiers applied to different sets of molecular descriptors (the method is presented in Section 2.4). The proposed ensemble performs better than the most accurate random forests classifier that uses a single descriptor set. Our method is found more computationally efficient than the random forests classifier applied to the pool of descriptor sets. Moreover, for the most part, the former model outperforms the latter.

## 2.2 BioAssay Data

I focus on the application of our methods to four BioAssay datasets made available by the Molecular Libraries Screening Center Network (MLSCN) (`http://mli.nih.gov/mli/mlscn`). The bioassay datasets were briefly introduced in Section 1.5 of Chapter 1. These assays typically resulted in continuous responses (percent inhibition) from a primary screen and binary responses (active versus inactive) from a secondary screen. In this thesis I focus on the binary responses. Table 2.1 presents the biological targets, number of compounds, number of active compounds and proportion of actives in the four assay datasets.

### 2.2.1 Assay AID364

Assay AID364 is a cytotoxicity assay conducted by The Scripps Research Institute - Molecular Screening Center (`http://mlpcn.florida.scripps.edu/`). The cytotoxicity of a chemical compound is the quality of being toxic to cells which is helpful to develop potential human therapeutics (`http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=364`). In this assay the goal of the Scripps Research Institute was to screen for cytotoxic compounds which were active against cancer cells. A total of 3311 compounds were tested, and 50 (1.51%) of them were found active. This is the smallest dataset, in terms of the number of compounds, among the four assays.

### 2.2.2 Assay AID371

Assay AID371, also a cytotoxic assay, was conducted by the Southern Research Molecular Libraries Screening Center (`http://www.southernresearch.org/life-sciences/`). The ability of a cytotoxic compound to inhibit the growth of the human non-small cell lung tumour line, A549, is a preliminary indication of anti-cancer activity for treating patients with lung

cancer (http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=371). This assay contains a total of 3312 compounds with 278 actives (8.4%).

### 2.2.3 Assay AID362

Assay AID362 is a formylpeptide receptor ligand binding assay conducted by the New Mexico Molecular Libraries Screening Center (http://nmmlsc.health.unm.edu/). The formylpeptide receptor (FPR) family of G-protein coupled receptors (GPCR) contributes to the localization and activation of tissue-damaging leukocytes at sites of chronic inflammation (http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=362). This assay contains 4279 unique compounds out of which 60 are active (1.4%).

### 2.2.4 Assay AID348

According to Wikipedia, "Beta-glucocerebrosidase is an enzyme with glucosylceramidase activity that is needed to cleave, by hydrolysis, the beta-glucosidic linkage of the chemical glucocerebroside, an intermediate in glycolipid metabolism." (http://en.wikipedia.org/wiki/Glucocerebrosidase). The inherited deficiency of this enzyme results in Gaucher's disease, a lysosomal storage disease characterized by an accumulation of glucocerebrosides. The deficiency also creates some forms of disorder affecting the central nervous system. This beta-glucocerebrosidase assay AID348, developed by the National Institute of Health Chemical Genomics Center (http://www.ncats.nih.gov/research/reengineering/ncgc/ncgc.html), has 4946 compounds out of which 48 are active. This is the largest assay in terms of the number of compounds with only 0.97% active compounds.

Table 2.1: The four bioassay datasets considered in this chapter. The numbers in () are the number of active compounds in each of the assays.

| Assay | Biological Target | # of Compounds | Fraction of Actives |
|---|---|---|---|
| AID364 | Potential human therapeutics | 3311 ( 50) | 0.0151 |
| AID371 | Lung tumor cells | 3312 (278) | 0.0839 |
| AID362 | Tissue-damaging leukocytes | 4279 ( 60) | 0.0140 |
| AID348 | Gaucher's disease | 4964 ( 48) | 0.0097 |

## 2.3 The Descriptor Sets / Natural Subsets of Variables

Molecular descriptors are numeric variables that describe the structure or shape of molecules, helping to predict the activity or properties of molecules in drug discovery. The Quantita-

tive Structure-Activity Relationships (QSAR) model relates activity/toxicity/drug potency of chemical compounds with their molecular descriptors. In this chapter, I have used five sets of molecular descriptors for the four PubChem assays. As the molecular descriptor sets are available in the QSAR literature as separate sets of variables each describing different properties of the chemical compounds, I term each of them as a "natural subset of variables."

The descriptor sets are obtained using the descriptor generator engine: computer software called PowerMV (Liu et al., 2005). The generated descriptor sets are: Atom Pairs (AP), Burden Numbers (BN) (Burden, 1989; Pearlman and Smith, 1999), Carhart Atom Pairs (CAP) (Carhart et al., 1985), Fragment Pairs (FP), and Pharmacophores Fingerprints (PH). The Burden Numbers are continuous descriptors, and the other four are bit string descriptors where each bit is set to "1" when a certain feature is present and "0" when it is not. See Liu et al. (2005) and Hughes-Oliver et al. (2011) for further information on these descriptor sets, how they are generated and what properties of the molecules they represent.

Table 2.2: The number of non-constant feature variables for each of the five descriptor sets. The numbers in () show the total number of feature variables generated by PowerMV.

| **Descriptor Sets** | **Assay Name** | | | |
|---|---|---|---|---|
| | **AID364** | **AID371** | **AID362** | **AID348** |
| Atom Pairs / AP (546) | 380 | 382 | 360 | 367 |
| Burden Numbers / BN (24) | 24 | 24 | 24 | 24 |
| Carhart Atom Pairs / CAP (4662) | 1585 | 1498 | 1319 | 1795 |
| Fragment Pairs / FP (735) | 580 | 580 | 563 | 570 |
| Pharmacophores / PH (147) | 120 | 119 | 112 | 122 |
| All Features / AF (6114) | 2689 | 2603 | 2378 | 2878 |

PowerMV generates a total of 546, 24, 4662, 735 and 147 feature variables for AP, BN, CAP, FP and PH, respectively. Table 2.2 shows the number of non-constant feature variables for each descriptor set and assay. The binary descriptor set CAP provides the largest number of feature variables ($1319 - 1795$), and the continuous descriptor set BN provides the smallest number of feature variables (24). Three of the four binary descriptor sets AP, CAP and FP are very rich in terms of number of variables. The continuous descriptor set BN is also rich in the sense that a continuous variable possesses much more resolution than a binary variable. The last row of Table 2.2 shows the number of feature variables after pooling the five descriptor sets together. I call this pool of descriptor sets *all features* and denote by AF.

## 2.4 Ensemble of Descriptor Sets

I build an ensemble by averaging probability of activity across descriptor sets and call the resulting ensemble as the "ensemble of descriptor sets"; in short EDS. Let us consider there are $n_s$ sets of molecular descriptors. Using training data, I build $n_s$ random forests using

$n_s$ sets of descriptors and compute probabilities of activity using each of the classifiers. The computed probabilities are then averaged across classifiers/sets of descriptors to build the final ensemble. The averaged probabilities of activity are used to rank the compounds in the test set. The compound with the largest averaged probability of activity is ranked first, followed by the compound with the second largest averaged probability of activity and so on.

The logic for building such an ensemble is straightforward. Dietterich (2000) pointed out that the performance of an ensemble depends critically on three factors: (i) strong base learners, i.e., good modelling/classification method to build the constituent models/classifiers (ii) low correlation between predictions obtained from the constituent classifiers, i.e., a diverse set of constituent models/classifiers, and (iii) the strategy of combining results from the constituent models/classifiers. To construct EDS, I choose "random forests" as the base learner, which is undoubtedly a strong classifier. Random forests is a highly accurate ensemble as it has often appeared to be the top performer in various QSAR studies (Bruce et al., 2007; Hughes-Oliver et al., 2011; Svetnik et al., 2003; Zhang et al., 2009). I conjecture that the different sets of descriptors are so diverse that they will generate considerably low correlated classifiers, i.e., low correlation between probabilities of activity across the classifiers.

There are two popular schemes to aggregate multiple classifiers in an ensemble: majority voting and probability averaging. Wang (2005) showed that the scheme of probability averaging is better than the majority voting. Zhang et al. (2009) pointed out that the gain in accuracy due to probability averaging relative to majority voting can get more than 6% improvement if the base learner is more accurate than that of a learner that performs random selection.

In this chapter, I average classifiers across five sets of descriptors: Atom Pairs (AP), Burden Numbers (BN), Carhart Atom Pairs (CAP), Fragment Pairs (FP), Pharmacophores (PH). The advantages are two-fold: (i) improvement in every dataset investigated in this chapter over the most accurate random forest applied to any single set of molecular descriptors, and (ii) less computational requirement than the random forest applied to the pool of descriptor sets (AF). The base learner "random forests" is constructed by the **R** packages **randomForest** (Liaw and Wiener, 2011). As there is only one tuning parameter ($m_{\text{try}}$) in random forests, I use the default, $m_{\text{try}} = \sqrt{D}$, following the suggestion of Breiman (2001).

## 2.5 Evaluation of Classifiers

The goal of this thesis is to rank the test compounds. But, I have assay datasets where the classes for all of the compounds are known, i.e., there is no test data. Hence, to evaluate predictive ranking performances of the ensembles, one potential solution is to split the data randomly into two parts of approximately equal size, and to consider one of the parts as training and the other as test. Simultaneously, it is possible to swap the training and test parts to use the entire data set. This splitting process of the data is called 2-fold cross-

validation. In this chapter, I have used balanced 10-fold cross-validation which I am going to describe next.

### 2.5.1 Balanced 10-fold Cross-validation

In regular 10-fold cross-validation the dataset is randomly partitioned into 10 folds each containing approximately equal number of compounds. Treating one of these folds as a "test set", the remaining $(10-1) = 9$ folds are combined together to form a "training set" in order to learn a model. This model is then applied to the "test set" to obtain predictions. The process is repeated, holding out each of the 10 folds for testing in turn. The advantage of the 10-fold cross-validation over a one-time split (2-fold cross-validation) is the increased precision of error estimation.

Since the assay datasets contain very few active compounds, I have to be careful in making the splits. For example, in regular 10-fold cross-validation, a particular fold might completely miss the rare active compounds and thus could make the training and test splits unrepresentative to each other with respect to the rare active compounds. Thus, I randomly divide the data into 10 folds in such a way that the resulting folds are approximately of equal sized with approximately equal number of actives in each. I named this new splitting method as the balanced 10-fold cross-validation.

In a particular turn of the cross-validation, a set of compounds appear exactly once in the test fold. After all of the ten turns of the cross-validation, all of the compounds appear exactly once in one of the ten test folds. Thus, having completed a balanced 10-fold cross-validation, I obtain the probabilities of activity for all of the compounds in an assay and use them to rank the compounds in order to determine the performance of the corresponding model.

### 2.5.2 Hit Curve

A *hit curve* provides graphical inspection of the performance of a classifier when the objective is ranking rare class objects ahead of the majority class objects. I used the estimated probabilities of activity to rank the compounds in a dataset and thus to produce a shortlist of the top ranked compounds. The determination of the number of compounds to be shortlisted for further investigation depends on the available resources in a project.

Let $n$ be the total number of compounds in a test set and let $a$ be the number active compounds. Let $0 \leq a_t \leq a$ be the number of actives (called *hits*) in a shortlist of size $t \in [1, n]$. The *hit curve* is a plot of $a_t$ versus $t$ or equivalently a plot of $a_t/a$ versus $t/n$ where the information is given in term of percentages. A hit curve can be viewed as a variant of the receiver-operating characteristic (ROC) curve where we plot true positive and false positive rates in the vertical and horizontal axes, respectively. The hit curve is an effective method for evaluating a ranking procedure showing its performance at all possible shortlist cutoff-point $t \in [1, n]$. Classifier 1 with hit curve $H_1(t)$ is unambiguously superior to classifier 2 with hit curve $H_2(t)$ if $H_1(t) \geq H_2(t)$ at every $t$.

The left panel of figure 2.1 shows seven hit curves generated by seven classifiers using the AID364 data. A hit curve resembles a step function with slopes in places where there are ties in the probabilities of activity. Having seen the hit curves, we may realize that finding the best hit curve may not be an easy task as they often cross and overlap each other. In such a situation, a single number summary of a hit curve might be helpful to evaluate classifiers and to facilitate comparisons.

### 2.5.3 Initial Enhancement

*Initial enhancement* (IE), proposed by Kearsley et al. (1996), is a popular measure to evaluate performances of classifiers when the objective is ranking rare class compounds. Suppose, having ranked the compounds using the estimated probabilities of activity, I have a shortlist of top $t \leq n$ compounds which are highly likely to be active. Initial enhancement at the shortlist of length $t$ is the hit rate at $t$ divided by the proportion of actives in the entire collection of compounds. The IE is defined as:

$$\text{IE} = \frac{(a_t/t)}{(a/n)}.$$

It is a relative measure of hit rate improvement offered by a classifier beyond what can be obtained under random ranking, and values much larger than 1 are desired. Notice that IE depends on the particular shortlist cutoff point $t$. Moreover, IE doesn't distinguish whether the actives are ranked at the very top or right before the end of the shortlist. Therefore, IE is not our favorite ranking evaluation method.

Here, we will see that sometimes IE does not reward a hit curve well with many actives in the start of a shortlist. The left panel of Figure 2.1 shows seven hit curves obtained from applying random forests to the five descriptor sets of the AID364 assay data. Following the results of Hughes-Oliver et al. (2011), IE is computed at $t = 300$. I compared two ensembles in terms of IE: random forests applied to atom pairs (AP) and random forests applied to the pool of five descriptor sets (AF). It is very clear that AP gives a very good initial enhancement of 6.401 which is bigger than 5.960, the IE obtained from AF. That is, AP ranks more actives in this shortlist of 300 compounds and hence the larger IE. But, careful inspection of the two hit curves (*red* and *yellow* lines are for AP and AF, respectively) reveals that AF identifies more actives earlier in the list than AP. If our goal is to rank the rare actives earlier in the list than at a particular test point, we should choose random forests with AF as our desired QSAR classifier among the two. As IE fails to reward early ranking of active compounds in a shortlist, we should try to find another metric to evaluate QSAR models simultaneously with IE.

### 2.5.4 Average Hit Rate

The *average hit rate* (AHR) gives a single number summary for a hit curve and is a common measure in information retrieval (Zhu, 2004). Suppose we shortlist the top $t \leq n$ compounds and $a_t$ of them are active. Then

$$h(t) = \frac{a_t}{t} \in [0, 1] \tag{2.1}$$

is the *hit rate* for the top $t$ ranked. Naturally, we want $h(t)$ to be as large as possible at every $t \in [1, n]$. Let $1 \leq t_1 < t_2 < \cdots < t_a \leq n$ be the positions of the active compounds in the ranked list. The average hit rate – occasionally referred to as *average precision* – is defined as

$$AHR = \frac{1}{a} \left[ h(t_1) + h(t_2) + \cdots + h(t_a) \right]. \tag{2.2}$$

AHR averages the "hit rates" of the selected active compounds, and larger AHR corresponds to the hit curve with most rapid early rise. If a hit curve ranks more actives ahead of the inactive compounds, then AHR rewards the hit curve by assigning a bigger number. AHR reaches the maximum 1, if all of the active compounds are ranked before the inactive compounds amongst those selected. To calculate AHR, we assume random ordering of the response in a tied group. Further details on AHR can be found in Chapter 3 of Wang (2005).

Sometimes AHR does not fully respect the properties of a hit curve in a single number. This measure gives very large weights to the actives found earlier in the list than those are found later. For example, we can compare two hit curves corresponding to random forests applied to atom pairs (AP) and pharmacophores (PH) in Figure 2.1. AP gives an AHR of 0.279 and PH gives an AHR of 0.285 – but for most of the list the hit curve corresponding to AP (the *red* line) is far above than the hit curve corresponding to PH (the *sky-blue* line). The larger AHR for PH is due to the fact that it identifies more hits at the start of the list: 10 of the first 10 compounds tested are hits.

If we rely on IE alone, we might infer that the AP gives a better model. On the other hand if we rely on AHR alone, we might infer that the PH gives a better model. Thus, sometimes, an evaluation metric alone might not be enough to reward a good hit curve. Hence, ideally I shall try to choose a classifier by maximizing both of the assessment measures: IE and AHR, i.e., I shall look for a classifier which gives more hits earlier in the list as well as more actives at the chosen test point. Otherwise, I will choose a classifier by maximizing AHR alone provided that the IE is close to the maximum.

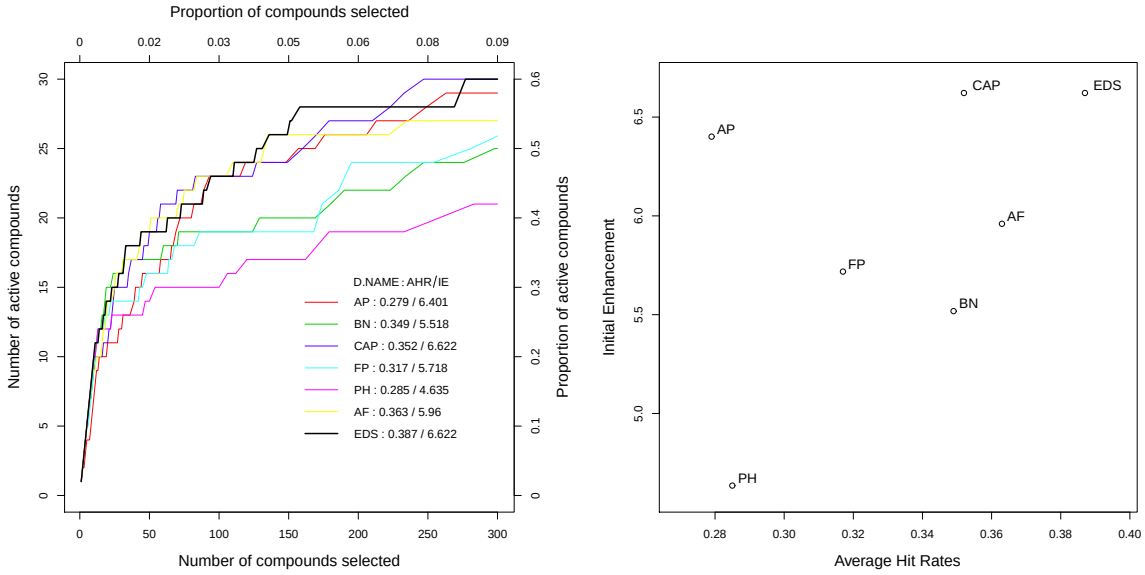
## 2.6 Results

This section contains results after applying the described ensembles to the assay datasets. So far, we have five random forests for the five sets of descriptors: atom pairs (AP), burden

numbers (BN), carhart atom pairs (CAP), fragment pairs (FP), and pharmacophores (PH). There are also two ensembles using all of the descriptor sets: random forests applied to the pool of descriptor sets (AF), and ensemble of descriptor sets (EDS). In total, there are seven ensembles to compare in each of the four assays.

### 2.6.1 Assay AID364

Panel (a) of Figure 2.1 shows seven hit curves corresponding to seven classifiers. The hit curves are produced from the probabilities of activity obtained from a balanced 10-fold cross-validation of the compounds of AID364 assay. As this assay contains only 50 active compounds, I computed IE at $t = 300$. The top three ensembles in terms of AHR are EDS, AF and CAP; and in terms of IE are EDS, CAP and AP. So the top performing ensemble is EDS. The top performing random forests using a single descriptor set is CAP. The performance of random forests with the pool of descriptor sets (AF) is in second and fourth positions in terms of AHR and IE, respectively.




(a) (b)

Figure 2.1: (a) The hit curves obtained from applying random forests to the five descriptor sets (AP, BN, CAP, FP and PH), and to the pool of the five descriptor sets (AF) of the AID364 dataset. The other hit curve is for the ensemble descriptor sets (EDS) applied to the AID364 dataset. (b) The corresponding initial enhancement (IE) versus average hit rate (AHR) plot for the seven ensembles applied to the AID364 assay. Both of the panels are obtained using the probabilities of activity from a particular balanced 10-fold cross validation.

To make comparison of the methods easy and straightforward, I have plotted IE against AHR for the same balanced 10-fold cross-validation of the AID364 dataset (Panel (b) of Figure 2.1). As the target is to maximize both AHR and IE, a good method is positioned in the top right corner of this plot. Fortunately, the proposed ensemble EDS is found in the top right

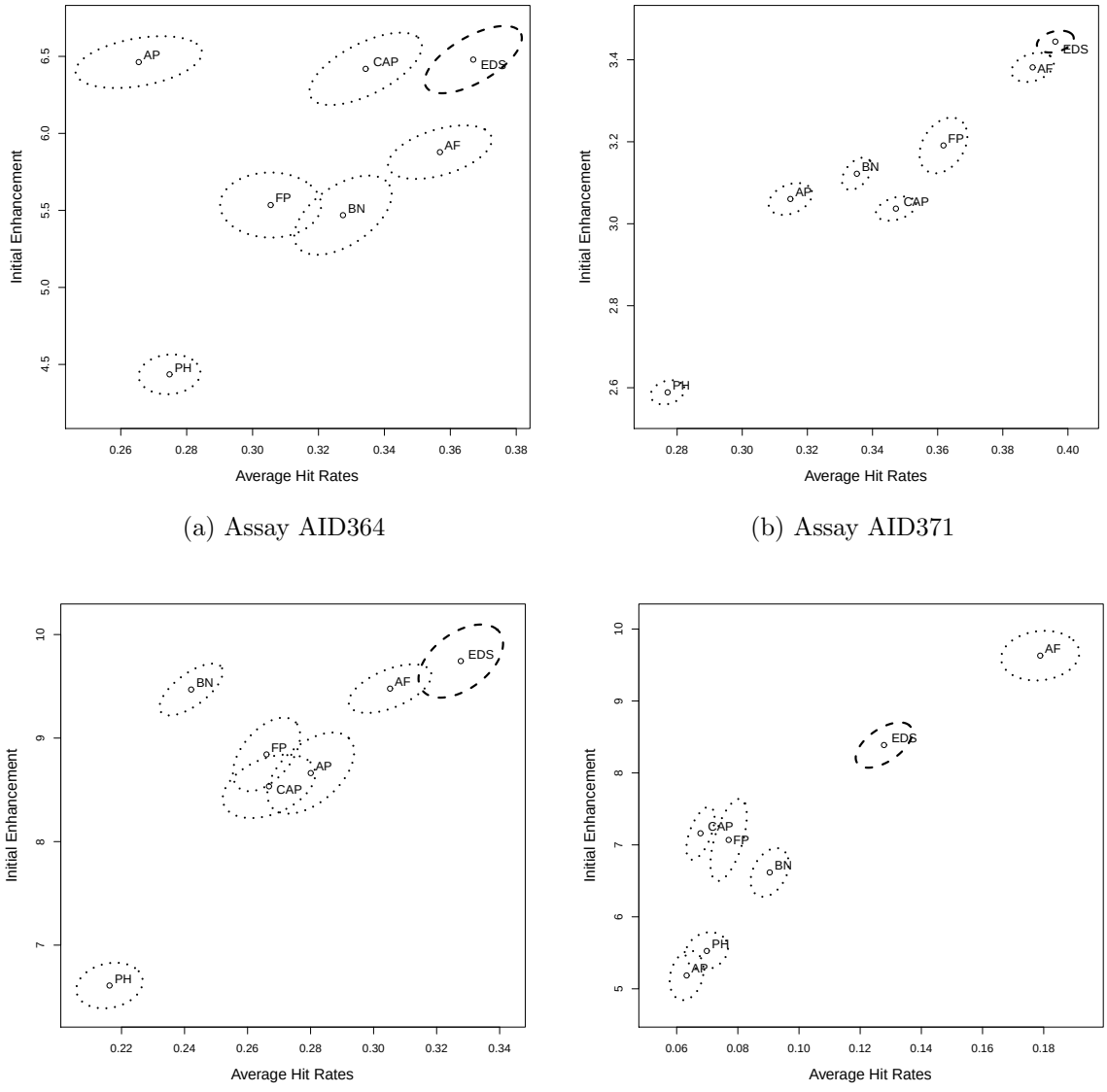
corner indeed. The top performing random forest using a single descriptor set is CAP, and our ensemble EDS outperforms CAP in terms of AHR and performs equally in terms of IE.

In order to compare average performances of the classifiers, I repeated the balanced 10-fold cross-validation for a total of 16 times. There are 16 processors available in my department's computing network for parallel computation and hence the 16 cross-validations. Since I want to compare performances in terms of AHR and IE, I performed a multivariate analysis of variance (MANOVA) with two dependent variables: AHR and IE. The factor variable *methods* has seven labels (AP, BN, CAP, FP, PH, AF, EDS) and the *blocking factor* has 16 blocks corresponding to the 16 cross-validations. After fitting the MANOVA model, employing standard assumptions (bivariate normality for the dependent variables, homogeneity of the covariance matrices across levels of the independent variables, independence of observations etc.), I have found that the effects of *methods* on both of the responses are highly significant using four testing methods: Pillai's Trace, Wilks' Lambda, Hotelling's Trace, Roy's Largest Root (Johnson and Wichern, 2002, Section 6.9). Table 2.3 shows the Hotelling-Lawley test-statistic from the MANOVA fit using two responses, AHR and IE, to the AID364 bioassay data. The *blocking factor* is also found significant which signifies the importance of including blocks in the model.

Table 2.3: Hotelling-Lawley test-statistic from the MANOVA fit using two responses, AHR and IE, to the AID364 bioassay data.

| Factors | *df* | Hotelling-Lawley | App $F$ | Num *df* | Den *df* | $Pr(> F)$ |
|---|---|---|---|---|---|---|
| Methods | 6 | 19.203 | 140.824 | 12 | 176 | $< 0.0001$ |
| Blocks | 15 | 4.877 | 14.307 | 30 | 176 | $< 0.0001$ |
| Residuals | 90 | | | | | |

As the effect of the factor variable *methods* is found significant, I have compared bivariate mean vectors ($\overline{\text{AHR}}$, $\overline{\text{IE}}$) for the seven ensembles. Using the bivariate normality assumption, the 95% confidence region for the bivariate mean vector is constructed. Panel (a) of Figure 2.2 shows the plots of mean AHR versus mean IE with 95% confidence regions for the seven ensembles in the AID364 assay. In terms of mean AHR and IE, the top performing ensemble is EDS. Using mean IE alone, the top performing random forests constructed on a single descriptor set are AP and CAP, and their performances exceeded the performance of AF. All of the three models (EDS, AP and CAP) provide significantly larger mean IE than AF. Using mean AHR, the second performer from the top is AF followed by CAP and BN. The message is: when we use different assessment criteria, the performances of the classifiers fluctuate a lot – but my ensemble EDS is in the top right corner. It is clear that when we aggregate classifiers over the five descriptor sets (AP, BN, CAP, FP, and PH), we get higher AHR and IE than any of the classifier constructed on a single descriptor set.




(a) Assay AID364

(b) Assay AID371

(c) Assay AID362

(d) Assay AID348

Figure 2.2: Plots of mean initial enhancement ($\overline{\text{IE}}$) versus mean average hit rates ($\overline{\text{AHR}}$) with 95% confidence bands obtained from applying the seven classifiers to the four bio assay datasets: AID364, AID371, AID362, and AID348.

### 2.6.2 Assay AID371

Assay AID371 contains 3312 compounds in total, out of which 278 are active. As there are 278 active compounds, I have computed IE at $t = 600$ test point. I have fitted a MANOVA using bivariate responses of AHR and IE, and the results are found similar to the results of the assay AID364. Panel (b) of Figure 2.2 shows the plots of mean AHR versus mean IE for the seven classifiers applied to the AID371 assay. As we have seen earlier, averaging random forests across descriptor sets provides better predictive ranking than any constituent random forests applied to any single descriptor set. In terms of the assessment measures AHR and IE, the top performing ensemble is EDS followed by AF. The top performing random forest among the individual descriptor sets is FP, and its performance is significantly lower than my ensemble EDS.

### 2.6.3 Assay AID362

As the number of active compounds is 60 in this assay, the IE has been computed at $t = 300$ test compounds. The bottom left panel of Figure 2.2 shows the plots of mean AHR versus mean IE for the seven classifiers applied to the assay AID362. I see results similar to the earlier assays AID364 and AID371: averaging random forests over the descriptor sets gives better predictive ranking of the rare actives than the random forests applied to any of the single descriptor set. In terms of mean IE and AHR, the top performing ensemble is EDS. The second performer from the top is AF. The top performing random forests constructed on single descriptor set is BN using IE and AP using AHR.

### 2.6.4 Assay AID348

Assay AID348 contains the largest number of compounds (4946) among the four assays with only 48 active compounds. The assessment measure IE is calculated at $t = 300$ shortlisted compounds. The MANOVA with bivariate response of AHR and IE gives similar results as other three assays, and hence we directly present the mean AHR versus mean IE plots (Panel (d) of the Figure 2.2). I see that averaging random forests over the descriptor sets gives better predictive ranking of the rare active compounds than the random forests applied to any single set of descriptors. The two ensembles, EDS and AF, which use the five sets of descriptors, have significantly outperformed all of the five random forests constructed using any single set of descriptors. The classifier EDS appeared in the second position in terms of both AHR and IE.

Surprisingly, random forests with the pool of descriptor sets (AF) appeared as the top performing ensemble in terms of mean AHR and IE for this assay. This result is not consistent with the results from other assays. To be honest, it is hard to answer exactly why random forests with the pool of descriptor sets (AF) outperformed EDS. But I conjecture that the constituent classifiers of EDS are in general weak and so is their ensemble. I provide an

example in the following paragraphs.

The basic principle of constructing an ensemble states that, in order to perform well, an ensemble needs to aggregate strong and diverse constituent classifiers (Breiman, 2001). The diversity among the constituent random forests in EDS will be examined in Section 2.7 . Here, I will be checking the strengths of the constituent random forests of EDS by comparing their performances with *classification trees* applied to the five descriptor sets of the AID348 Assay. The classification trees are fitted using the default settings of the **R** package **Tree** (Ripley, 2011). For a fair comparison, I used the same 16 balanced 10-fold cross-validations as in random forests to evaluate classification trees.

Table 2.4 shows the mean AHRs and IEs over the 16 cross-validations for classification tree (Tree) and random forests (RF) applied to the five descriptor sets (AP, BN, CAP, FP and PH). The table also shows the performances of their ensembles, EDSs. The results are fairly interesting. In terms of mean AHR, Tree with CAP outperformed random forests with CAP. In terms of mean IE, Tree with FP and PH outperformed random forests with FP and PH, respectively. The ensembles of descriptor sets with random forests and classification trees perform fairly similar.

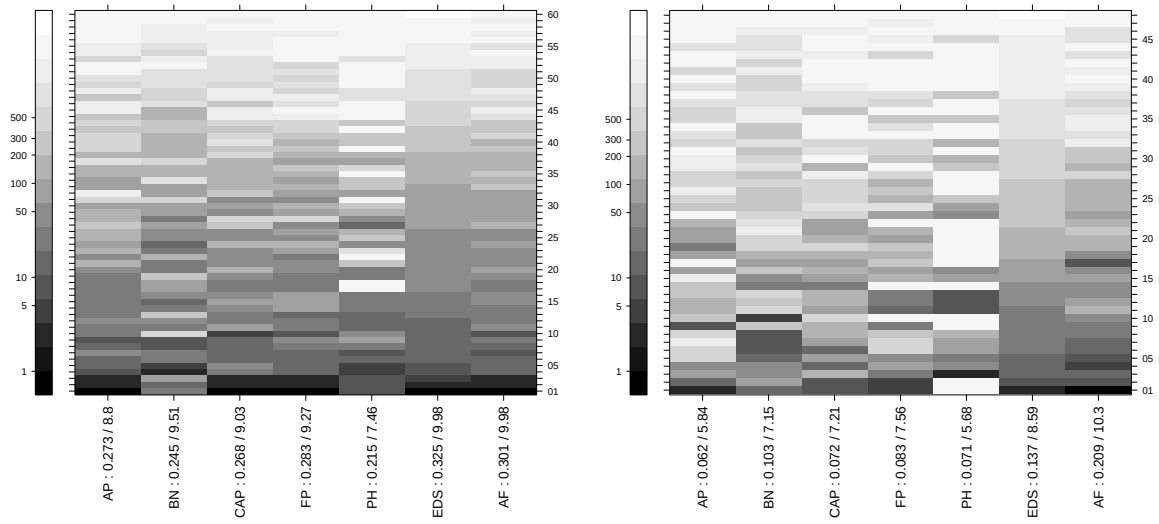
Table 2.4: The mean AHRs and IEs over the 16 cross-validations for classification tree (Tree) and random forests (RF) applied to the five descriptor sets (AP, BN, CAP, FP and PH), and for their ensembles EDSs.

| Metrics | Classifiers | Constituents | | | | | Ensemble |
|---|---|---|---|---|---|---|---|
| | | AP | BN | CAP | FP | PH | EDS |
| AHR | Tree | 0.033 | 0.039 | 0.071 | 0.066 | 0.069 | 0.122 |
| | RF | 0.063 | 0.090 | 0.068 | 0.077 | 0.070 | 0.128 |
| IE | Tree | 3.547 | 4.585 | 6.470 | 7.290 | 5.774 | 8.007 |
| | RF | 5.186 | 6.618 | 7.161 | 7.069 | 5.526 | 8.388 |

In general, random forests provides much better predictive performance than a classification tree. So the question arises, "why random forests could not improve performances of classification trees for some descriptor sets?" Perhaps those descriptor sets contain many unimportant feature variables for which the performances of random forests are decreased. Thus, equipped with weak constituent random forests the ensemble EDS appears weak. However, as you will see, I will improve the performance of our ensemble in Chapters 3 and 4.

## 2.7 Diversity Map

In this section, I shall try to understand why the proposed ensemble of averaging probabilities of activity across descriptor sets provides good predictive ranking of the rare active compounds. I plot the ranks of the active compounds corresponding to the seven classifiers




(a) Assay AID362 (b) Assay AID348

Figure 2.3: Diversity maps of ranks of the active compounds for AID362 and AID348 assays. The probabilities of activity associated with the cross-validation number 1 are used to rank the compounds.

applied to AID362 and AID348 Assays (Panels $a$ and $b$ of Figure 2.3, repectively). The compounds are ranked using the probabilities of activity obtained from the cross-validation number 1. If an active compound is ranked at the top position by a classifier, the compound receives the darkest gray colour. If a compound is ranked down in the sequence, it receives absolutely no colour. The colour key in each plot shows where in the sequence an active compound is ranked by a classifier. I also sequence the active compounds in the right side of the vertical axis. The compounds are sequenced using the probabilities of activity obtained from EDS. As a result, we can see the diversity in ranks across different classifiers. Such a plot is popularly known as the diversity map (Hughes-Oliver et al., 2011). I also report the AHRs and IEs for the seven classifiers to understand how strong the classifiers are.

It is the diversity in AP, BN, CAP, FP and PH which makes our ensemble EDS perform better than the top performing random forests applied to any of the five descriptor sets. You can see there is variation in ranks across the random forests applied to AP, BN, CAP, FP and PH, i.e., different classifiers rank different sets of active compounds well. For example, in AID362 Assay the active compound 1 is ranked very well by AP, CAP and FP, and moderately well by BN and PH. As a result of this variation or diversity, the compound is ranked very well by EDS. We also see diversity in ranks among the classifiers AP, BN, CAP, FP and PH in Assay AID348 (Panel (b) of Figure 2.3). We see EDS ranks a compound very well if all of the constituent classifiers rank it well. If some of the classifiers rank a compound well and the others fail to rank it well then EDS ranks the compound well too. If none of the constituent classifiers rank a compound well then EDS fails to rank it well.

However, it has been observed that the model random forests with AF is computationally very intensive. The next section (Section 2.8) presents computational complexity of the classifiers we presented so far.

## 2.8 Computational Complexity

Table 2.5 shows the computational time in minutes for the seven ensembles to complete a balanced 10-fold cross-validation. The times are obtained in an `Intel Core I5 CPU` (2.67 GHz) with 6.00 GB RAM and 64-bit Windows Operating System. It is easy to see that the computational time does not vary much for increasing the number of compounds, but depends heavily on the number of feature variables. As a result, the computational time for AF is very high comparing to the other models. It is worthy to mention that EDS and AF use the same number of feature variables, but the former requires $33 - 48\%$ less computational time than the latter.

Table 2.5: Observed computational time (in minutes) to complete a balanced 10-fold cross-validation for the seven ensembles applied to the five descriptor sets of the four assay datasets. The number in () shows the number of feature variables used to construct the corresponding classifier.

| Assay Name (Number of Compounds) | AID364 (3311) | AID371 (3312) | AID362 (4279) | AID348 (4946) |
|---|---|---|---|---|
| Atom Pairs (360-382) | 9.09 | 12.30 | 11.28 | 12.76 |
| Burden Numbers (24) | 0.49 | 0.74 | 0.67 | 0.70 |
| Carhart Atom Pairs (1319-1795) | 103.78 | 114.91 | 101.85 | 170.79 |
| Fragment Pairs (563-580) | 21.60 | 27.59 | 25.98 | 31.05 |
| Pharmacophores (112-122) | 3.11 | 3.86 | 3.38 | 5.39 |
| Ensemble of DS (2378-2878) | 138.62 | 159.36 | 144.01 | 220.55 |
| All Features (2378-2878) | 252.70 | 278.99 | 230.14 | 329.62 |

The computational time of a random forest can be divided up into two parts: (1) time for reading the data, and (2) time for computation. Having read the data, the **R** package **randomForest** stores the entire data file into its memory. Obviously, a large data file will occupy large computer memory. So this program is not scalable to a dataset for which the memory requirement exceeds the memory capacity of a computer. So far we have been successful in running random forests using the pool of five descriptor sets (AF). But there are many descriptor sets available in cheminformatics literature. Thus, if the size of the memory requirement for the pool of many descriptor sets ($> 5$) exceeds the memory of a computer then AF would not be scalable to that computer. In the presence of many descriptor sets, EDS is easily scalable, whereas scalability of AF is in doubt.

## 2.9 Summary

I have introduced a novel ensemble method through averaging probabilities of activity over the molecular descriptor sets to rank rare active compounds ahead of the majority inactive compounds in QSAR studies. Convincing results have been observed in favor of the proposed ensemble which ensures significant improvement over the most accurate classifier that can be built using a single set of descriptors. In fact there is no any particular set of descriptors that performs uniformly best over the others: the results of this chapter also favour this statement. Moreover, the proposed method often outperforms the random forest classifier applied to the pool of the five descriptor sets. On the other hand, the ensemble EDS is scalable to many descriptor sets where it is quite difficult or, perhaps, impossible to apply random forests to AF. We have used five sets of descriptors in this study - but it is possible to include many as there are many sets of molecular descriptors available in cheminformatics literature.

One of the reasons why averaging over descriptor sets works better than the top performing random forests applied to any of the five descriptor sets is the diversity of the classifiers built using different sets of molecular descriptors. The classifiers are diverse in the sense that each of them ranks diverse sets of active compounds well. As a result, the ensemble obtained by averaging the probabilities of activity ranks more active compounds – perhaps, a subset of the union of active compounds found by all of the constituent classifiers – than the number of active compounds ranked by any single constituent classifier.

A naive method of pooling the descriptor sets together (AF) gave top performance in conjunction with random forests for the AID348 assay. Perhaps there is manageable number of good feature variables in that assay and a regular random forests can effectively handle them to produce top result. However, this top result alone does not justify the use of AF instead of EDS as the performance of AF fluctuates a lot for other assays. For example, AF sometimes provides weaker performance than the most accurate classifier based on an individual descriptor set. EDS is computationally efficient, scalable to high dimension, and is one of the top-two models. Moreover, this model gives better performance than the top performing classifier based on any single molecular descriptor set.

I used random forests because it is found as one of the top performing ranking models. Some other modelling methods (classification tree, $k$-NN, boosted tree, for example) might also help to give improved performance as they are also found highly accurate in other applications. However, the good results from the ensemble of natural descriptor sets motivates me to uncover *data-adaptive subsets of variables* in a variable-rich descriptor set which I am going to describe next in Chapter 3.

# Chapter 3

# Ensembling Data-Adaptive Subsets of Variables

## 3.1 Introduction

A new ensemble using data-adaptive subsets of variables is proposed and applied to the four highly unbalanced (i.e. sparse) drug discovery datasets. The response variable in our datasets is the *compound activity status*. The predictors are the five sets of variables called "descriptor sets". The variables in each descriptor set describe chemical/molecular structures of the compounds. Each of the five descriptor sets contains a large number of predictors. Without uttering the conventional term "curse of dimensionality" we say that the descriptor sets are *rich* in terms of number of variables. In fact, we judiciously use the richness of the descriptor sets to develop our ranking procedure.

We introduce the concept of *phalanx*, a group of variables that work well together to form one of several models in an ensemble. Capitalizing on the richness of variables, we form phalanxes by grouping variables together. Phalanxes are characterized by their ability to yield a complete predictive model. The variables in a phalanx are in the same model because they complement each other while the variables in different phalanxes work better in ensemble.

*Natural phalanxes* are present in our datasets. For a particular bioassay, each of the five descriptor sets can be considered as a natural phalanx. By ensembling over the natural phalanxes, our ensemble EDS performed better than the top performing random forest built on a single set of descriptor (see Chapter 2). Natural phalanxes are sometimes suggested by subject matter knowledge. For example in Tebbutt et al. (2005) SNP genotyping platform, grouping of the variables is naturally suggested by the different chemical procedures employed in the genotyping platform. Finding a good ensemble of natural phalanxes is loosely connected but different from group LASSO (Yuan and Lin, 2007) and its variants (Meier et al., 2008). The main difference is that in group LASSO the *given* groups are only evaluated by their ability to work together with other groups in a *single* model. Moreover, the groups may or may not conform natural phalanxes – depending on whether the variables in each group are able to yield by themselves a good predictive model.

On the other hand, unknown *statistical phalanxes* may be present in variable rich datasets. Uncovering these phalanxes is an interesting and challenging statistical problem. Motivated from the results of Chapter 2, we propose an algorithm to uncover statistical phalanxes and

demonstrate its performance in several applications. We can think of our phalanx forming algorithm as a special type of clustering of variables where "similarity" means working well together in a particular model and "dissimilarity" means working well apart in different models which are ultimately ensembled.

In principle, phalanxes can be formed using any given classifier. We use random forests (Breiman, 2001) because of its well-known superior performance for ranking compounds. We compare our ensemble of phalanxes with random forests and regularized random forests (Deng and Runger, 2012). As we will see in Section 3.7, the descriptor sets contain a fraction of noise variables, that is, variables which are not useful to rank the compounds. Regularized random forests is useful for screening noise variables in a dataset. By comparing our ensemble of phalanxes against regularized random forests we convey the message that, in this context, phalanxing is better than regularization.

We have found better performance of the ensemble of phalanxes than its competitor random forests and regularized random forests. Our ensemble performs very well when there are many variables in a descriptor set and when the proportion of active compounds is very small. The harder the problem the better the ensemble of phalanxes performs. When the proportion of active is very small, regularized random forests outperforms random forests; and when the number of variables is very large, random forests outperforms regularized random forests. In both of the situations, the difference between the top performer and its nearest performing ensemble is very large – where the top performer is our ensemble. Having convinced by the good results of our ensemble, we urge people to use the term "rich-in-variables" instead of "curse-of-dimensionality".

## 3.2 Datasets and Variables

We analyze 20 datasets from four different assays in the Molecular Libraries Screening Center Network (MLSCN). The response variable in all the cases is the compound activity status. Each of the assays contains a number of compounds that are either active or inactive against a biological target. Section 2.2 of Chapter 2 describes the four bioassay datasets.

All the assays are highly unbalanced (i.e. sparse) in terms of proportions of actives. For three of the assays the fractions of active compounds are around 0.01. Hence, the problem of sparseness poses difficult challenges. For example, a classification tree would soon run short of rare active compounds and tend to build a shallow tree, using a few important variables. If there are many important variables, their participation would become difficult. This problem is partially addressed but not completely resolved by random forests-like ensembles. Our approach allows more participation of the variables in the phalanxes.

The covariates in all the cases are molecular descriptors – numeric variables that describe the structure or shape of molecules – which may help to predict the activity of molecules in drug discovery. Quantitative Structure-Activity Relationship (QSAR) models relate activ-

ity/toxicity/drug potency of chemical compounds with its molecular descriptors. We consider five sets of descriptors for each of the four assays. This gives a total $4 \times 5 = 20$ datasets. Section 2.3 of Chapter 2 describes the five descriptor sets in four bioassays.

## 3.3 Performance Measures

In Chapter 2, we have described a few methods to evaluate ranking procedures when the goal is to detect a few rare occurrences hidden in the midst of a large number of uneventful objects. Ranking is usually performed using a classifier to estimate the probability of activity for the candidate compounds. We rank the compounds using their probabilities of activity. The goal is to rank the actives at the top of the list.

The standard method for evaluating the performance of a classifier is by computing its misclassification error (ME): the proportion of compounds assigned to the wrong class. But ME is not useful when the classes are highly unbalanced (Zhu et al., 2006). Instead of minimizing ME, we evaluate a classifier using a hit curve. The definition of a hit curve is presented in Subsection 2.5.2 of Chapter 2.

Figure 3.1 shows hit curves for three ensembles RF, RRF, and EPX applied to Atom Pairs of AID348 Assay. We consider a shortlist of 300 compounds because we wish to rank the actives earlier in the list. Note that EPX dominates the other two ensembles as it finds more active compounds earlier in the list. RRF and RF do not dominate each other because their hit curves criss-cross at several points.

Comparison of crossing hit curves, as in RF and RRF of Figure 3.1, could be difficult if we have many of them. This comparison would become impossible to handle if we repeat computations many times through cross-validation. Thus, a single number summary of a hit curve is desirable in order to facilitate comparison and to automate the selection process.

The average hit rate (AHR) gives a single number summary for a hit curve and is a common measure in information retrieval (Zhu, 2004). The definition of average hit rate is provided in Subsection 2.5.4 of Chapter 2. If a classifier identifies more hits earlier in the ranked list, AHR reward that classifier by assigning a large number. The AHRs for RF, RRF and EPX (see Figure 3.1) are 0.06, 0.08 and 0.20, respectively. We use AHR not only to evaluate the ranking procedures but also to form the phalanxes (introduced in Section 3.4).

Initial enhancement (IE), introduced by Kearsley et al. (1996), is a popular ranking performance measure in QSAR studies. The definition of IE is presented in Subsection 2.5.3 of Chapter 2. Usually, IE evaluates the performance of a classifier at a particular cutoff point of a hit curve. Naturally, larger values of IE indicate better ranking by a classifier. Following Hughes-Oliver et al. (2011) we use a shortlist (cutoff) of 300 compounds. The IE for RF, RRF and EPX in Figure 3.1 are 5.84, 5.53 and 6.26, respectively. For the reasons mention in Subsection 2.5.3 of Chapter 2, IE is not our favorite ranking evaluation method, however.

We have used balanced 10-fold cross-validation to assess the performance of the ranking

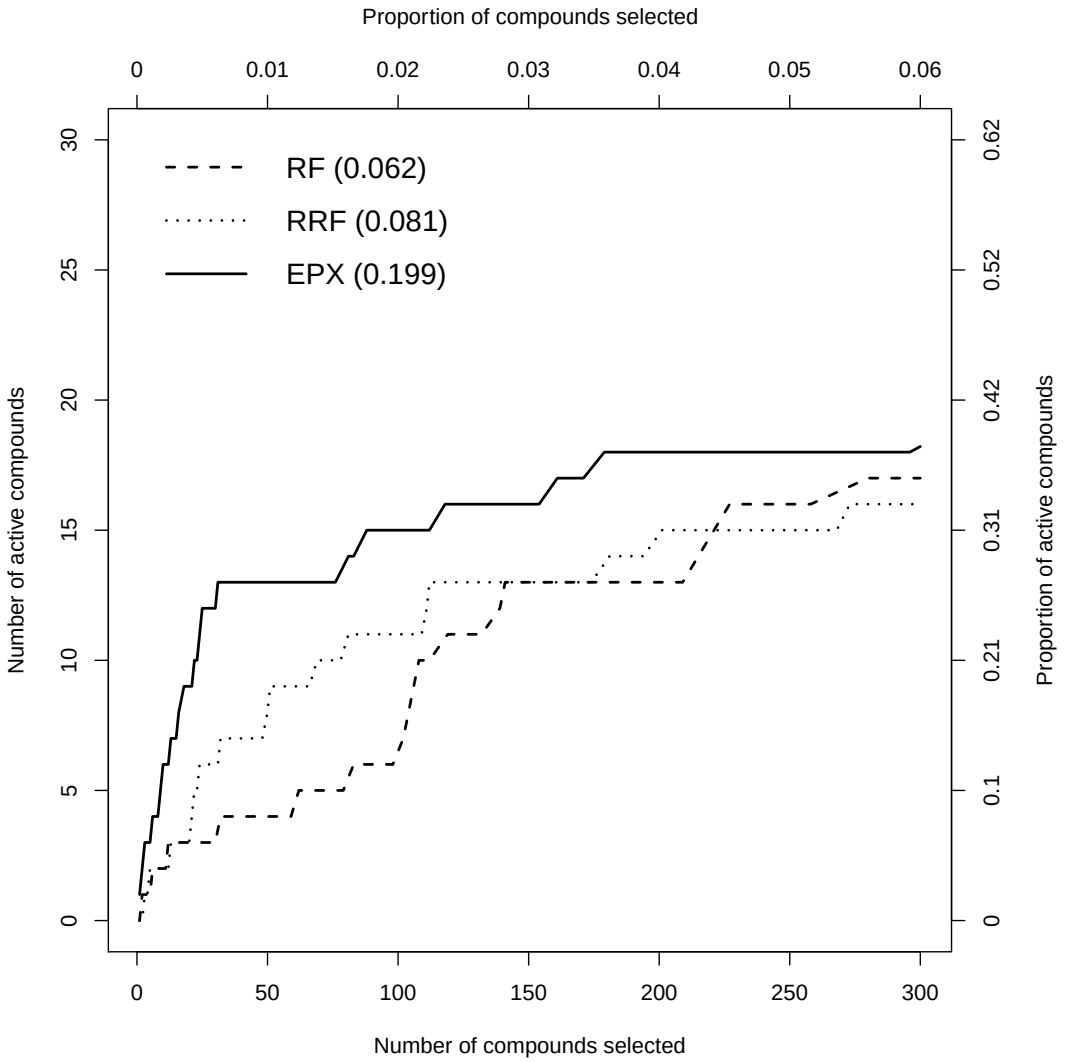



Figure 3.1: Illustration of hit curves for three methods random forests (RF), regularized random forests (RRF) and ensemble of phalanxes (EPX) corresponding to Atom Pairs of AID348 Assay. The number in () is the corresponding Average Hit Rate.

procedures. Since the assay datasets contain very few actives, we make sure that there are enough actives in each of the ten groups. One of the groups is separated to serve as "test set" and the remaining nine groups are combined to fit the ranking model. The ranking model is then applied to the left out "test set" to obtain probability of activity for the compounds. The process is repeated for the ten groups to obtain probabilities of activity for all the compounds (please see, Subsection 2.5.1 of Chapter 2).

## 3.4 Searching for Data-Adaptive Subsets / Phalanxes of Variables

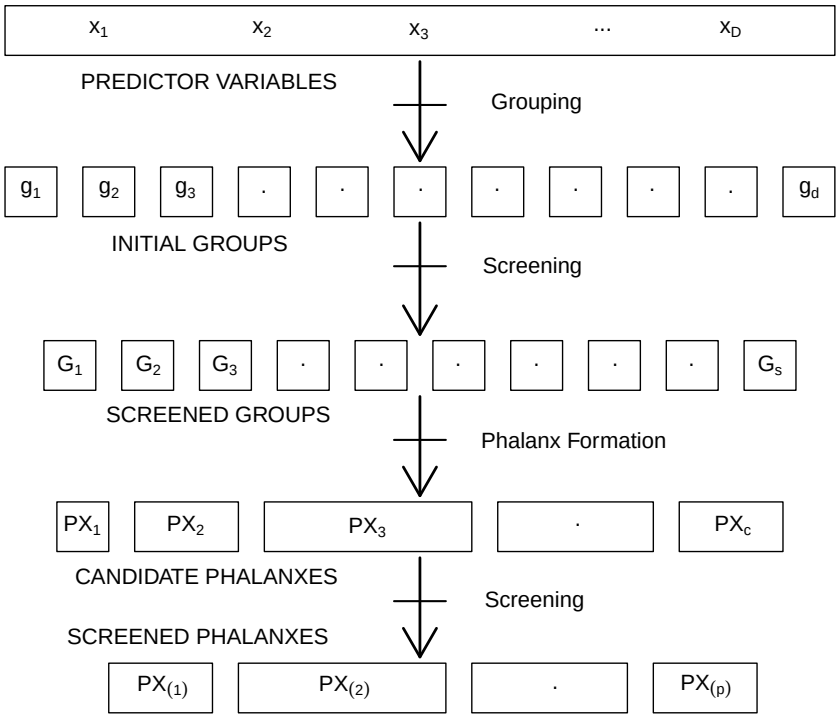
The term *phalanx* is borrowed from the ancient military formation used by `Alexander The Great` and his father `Philip II of Macedon` to deploy infantry soldiers in the battlefield. The soldiers in each phalanx had to trust their neighbours to protect them, and be willing to protect their neighbours too. To provide psychological incentive the phalanxes were organized into group of friends and family members closely together. As a result, the strength of the phalanx would depend upon the individual strength of the soldiers and the psychological/emotional bond between them. A phalanx was an autonomous fighting unit and could be ensembled with other phalanxes to form a formidable military machine. In our case, the groups of variables in a *statistical phalanx* work better together than separated in different groups. Moreover, as the variables in different phalanxes work better separated than together, the phalanxes help each other yielding a stronger ranking procedure.

Given $m$ test items and $D$ covariates there exists in principle an optimal partition of these covariates into $p+1$ groups consisting of $p$ phalanxes plus a subset of screened noise variables. Optimality here is in the sense of best ranking of the given test items by some given criterion (e.g. AHR, IE, etc.). Finding this optimal partition, however, is very difficult (unfeasible) because the number of phalanxes $p$ and the phalanx membership are unknown making the total number of possible solutions exponentially large. Moreover, in most applications we do not have a test set. So, we aim at the more realistic goal of finding a good phalanx partition using cross-validation and a greedy aggregation algorithm that resembles hierarchical clustering. As shown in Figure 3.2, there are five main steps:

- The input of predictor variables $(x_1, x_2, \cdots, x_D)$.
- The arrangement of the $D$ predictors into $d$ initial groups (this step is optional, necessary only when dealing with binary predictors).
- The $d$ initial groups are screened down to $s$ groups.
- The $s$ groups are arranged into $c$ candidate phalanxes.
- The $c$ candidate phalanxes are screened down to $p$ final phalanxes.

Our algorithm and its main parameters – *groups*, *iquant*, *nsim*, and *ntree* – are described below.

Figure 3.2: Schematic presentation of the algorithm of phalanx formation: $D$ predictor variables are grouped into $d$ initial groups, then reduced down to $s$ screened groups, then combined into $c$ candidate phalanxes, which are then reduced to $p$ screened phalanxes in the final army ($D \geq d \geq s \geq c \geq p$).




### 3.4.1 Predictor Variables

The main input of our algorithm is the $D$ predictor variables: $x_1, x_2, \cdots, x_D$. The other input is the response variable $y$.

### 3.4.2 Initial Groups

As a preliminary (optional) step of the phalanx formation algorithm, the $D$ original covariates may be grouped into $d \leq D$ initial groups $g_1, g_2, \cdots, g_d$. This step is implemented by the group indicator *groups* and optional (the default is taking the groups equal to the input covariates). However, initial grouping is convenient and recommended in the case of binary covariates because of two reasons: (i) It reduces computational burden. Since we fit classifiers for all pairs of covariates our computational complexity is quadratic in the number of covariates. (ii) It increases the ranking models resolution. The resolution of a binary predictor is very low

as it only has two possible values 0 and 1. The resolution of a group of $k$ binary covariates is higher as they can jointly distinguish $2^k$ possible situations. So forming groups in the case of binary variables not only saves time but may also increase the overall ranking accuracy of the models.

In our application we aim at forming initial groups of covariates that are as diverse as possible. In particular, we have experimentally verified that grouping based on similar covariate names is better than random grouping. For example, there are seven features which represent the atomic distances (up to 7 bonds) between a pair of atoms or groups of atoms. For Fragment Pairs, an example of these seven features is: AR_01_AR, AR_02_AR, AR_03_AR, AR_04_AR, AR_05_AR, AR_06_AR, and AR_07_AR. Here, AR_02_AR represents two phenyl rings separated by two bonds. In this case each group of covariates corresponds to a particular pair of atoms.

### 3.4.3 Screening Initial Groups

We screen out weak initial groups to reduce computational burden and noise. Strong groups are those which help to rank high the active compounds yielding larger average hit rates. To be considered strong a group must either be strong by itself or in a model with another group or in an ensemble with another group.

The key tool for screening groups (and for other procedures in our algorithm) is the *distribution of AHR under random ranking*. To obtain this distribution we randomly permute the response variable $y$ *nsim*=1000 times and for each permutation we obtain the corresponding AHR. We then compute *iquant*-th quantile of this distribution which we denote $\text{AHR}_{iquant}$. The parameter *iquant* gives the desired level for the probability threshold (0.95 in our applications).

We now describe how we decide if a given group, $g_i$, is strong enough to be kept in the phalanx formation algorithm. We perform three tests and the group survives if it passes at least one of these tests.

**Test 1:** *Individual strength of* $g_i$. We first fit a random forest classifier using $y$ and $g_i$ with *ntree*=150 trees and extract the out-of-bag (OOB) vector of probability of activity, $\widehat{P}(g_i)$. Using these probabilities we compute $\text{AHR}_i = \text{AHR}\left(\widehat{P}(g_i)\right)$. We say $g_i$ is *strong by itself* if

$$\text{AHR}_i \geq \text{AHR}_{iquant}. \tag{3.1}$$

The OOB class probabilities are an attractive feature of random forests. Breiman (1996c, 2001) showed that the OOB class probabilities are as good as the cross-validated class probability. See also Tibshirani (1996) and Wolpert and Macready (1999).

**Test 2:** *Joint strength of* $g_i$ *together with another group.* Now we fit two random forest classifiers with *ntree* = 150 trees. The first one with covariates $\{g_i, g_j\}$ and the second one with covariate $g_j$, $j \neq i$. Again, we obtain the OOB probability of activity vectors $\widehat{P}(g_j)$

and $\widehat{P}(g_i, g_j)$, and compute the corresponding average hit rates $\text{AHR}_j = \text{AHR}\left(\widehat{P}(g_j)\right)$ and $\text{AHR}_{ij} = \text{AHR}\left(\widehat{P}(g_i, g_j)\right)$. We say $g_i$ has *strong joint predictive power together with* $g_j$ if

$$E(\text{AHR}_R) + \text{AHR}_{ij} - \text{AHR}_j \geq \text{AHR}_{iquant}, \tag{3.2}$$

where $E(\text{AHR}_R)$ is the expected AHR under random ranking (roughly equal to $a/n$).

**Test 3:** *Joint strength of $g_i$ in ensemble with another group.* Now we compute $\text{AHR}_{\overline{ij}} = \text{AHR}\left(\left\{\widehat{P}(g_i) + \widehat{P}(g_j)\right\}/2\right)$ and say that $g_i$ has *strong ensemble predictive power with* $g_j$ if

$$E(\text{AHR}_R) + \text{AHR}_{\overline{ij}} - \text{AHR}_j \geq \text{AHR}_{iquant}. \tag{3.3}$$

The group $g_i$ **will not be screened out** if it is strong by itself (satisfies (3.1)) or has strong joint predictive power together with another group (satisfies (3.2) for some $j \neq i$) or has strong joint predictive power in ensemble with any another group ((satisfies (3.3) for some $j \neq i$ ). After removing weak initial groups, we update the list of preliminary groups to $\{G_1, G_2, \cdots, G_s\}$ and the updated group indicator parameter *groups*.

### 3.4.4 Phalanx Formation

The input for this step are the strong groups $G_1$, $G_2$,...,$G_s$ that survived screening. This step resembles hierarchical clustering: at each iteration the two groups that optimize a certain *merging criterion* are merged and the parameters *groups* and $s$ are updated to reflex the merge. The merging criterion consists of two conditions: (i) minimizing the ratio $\text{AHR}_{\overline{ij}}/\text{AHR}_{ij}$ and (ii) beating a baseline value $b$ with default value 1. We keep merging groups of variables until the candidate phalanxes are found good to ensemble, i.e., $b \geq 1$. The following example illustrates the hierarchical procedure.

Let $s = 3$ and consider the three variables $G_1$ = WBN_GC_L_1.00, $G_2$ = WBN_EN_H_0.50, and $G_3$ = WBN_LP_H_1.00 from the descriptor set BN in AID348. The AHRs when we paired them together with each other are: $\text{AHR}_{12} = 0.052$, $\text{AHR}_{13} = 0.037$ and $\text{AHR}_{23} = 0.054$. The AHRs when we paired them to ensemble with each other are: $\text{AHR}_{\overline{12}} = 0.069$, $\text{AHR}_{\overline{13}} = 0.050$ and $\text{AHR}_{\overline{23}} = 0.031$. The corresponding $\text{AHR}_{\overline{ij}}/\text{AHR}_{ij}$ ratios are 1.312, 1.357 and 0.570, respectively. As the variables $\text{G}_2$ and $\text{G}_3$ give the smallest ratio which is less than 1, we merge $\text{G}_2$ and $\text{G}_3$ together. We recomputed AHR by putting $\text{G}_1$, $\{\text{G}_2, \text{G}_3\}$ together as 0.058. We also recomputed AHR by ensembling $\text{G}_1$ and $\{\text{G}_2, \text{G}_3\}$ as 0.069. Their ratio is 1.177 which is greater than 1. Hence, we stop by producing two candidate phalanxes $\text{PX}_1 = \{\text{G}_1\}$ and $\text{PX}_2 = \{\text{G}_1, \text{G}_2\}$.

### 3.4.5 Screening Out Weak Phalanxes

A candidate phalanx is kept in the ensemble if it is individually strong (see 3.1) or it is strong in ensemble with other phalanx (see 3.3). After this second stage of screening, the surviving

**Algorithm 3.1** Phalanx Formation

1. **Setting the arguments:**
   (a) Predictors and the response: $\{x_1, x_2, \cdots, x_D\}$ and $y$.
   (b) Set *initial groups*, *nsim*=1000, *iquant*=0.95, *ntree*=150.
2. **Forming initial groups of variables:**
   (a) $g_1, g_2, \cdots, g_d \leftarrow x_1, x_2, \cdots, x_D$ (*initial groups*).
3. **Screening initial groups of variables/predictors:**
   (a) Record ordering of the compounds. Permute the response variable $y$ and compute AHR. Repeat the process *nsim* times.
   (b) Store $\text{AHR}_{iquant}$, the *iquant*-th quantile of the distribution of AHR under random ranking.
   (c) For each group of variables $g_i$; $i = 1, 2, \cdots, d$, fit a random forest growing *ntree* trees and get the out-of-bag (OOB) probability vector $\widehat{P}(g_i)$. Store the probability vectors in the columns of a matrix, say PROB.
   (d) Using $\widehat{P}(g_i)$, compute average hit rate $\text{AHR}_i = \text{AHR}\left(\widehat{P}(g_i)\right)$ and store in a vector, say SAHR.
   (e) For each pair of initial groups $\{g_i, g_j\}; i = 1, \cdots, (d-1); j = (i+1), \cdots, d$, get $\widehat{P}(g_i, g_j)$, $\left\{\widehat{P}(g_i) + \widehat{P}(g_j)\right\}/2$ and thus $\text{AHR}_{ij} = \text{AHR}\left(\widehat{P}(g_i, g_j)\right)$, $\text{AHR}_{\overline{ij}} = \text{AHR}\left(\left\{\widehat{P}(g_i) + \widehat{P}(g_j)\right\}/2\right)$, respectively. Store $\text{AHR}_{ij}$ and $\text{AHR}_{\overline{ij}}$ in matrices AHRSYN and AHRCOM, respectively.
   (f) Screen the $i$th group of variables out if $\max[\text{AHR}_i, E(\text{AHR}_R) + \text{AHR}_{ij} - \text{AHR}_j, E(\text{AHR}_R) + \text{AHR}_{\overline{ij}} - \text{AHR}_j] < \text{AHR}_{iquant}\ \forall\ j \neq i = 1, \cdots, d$.
   (g) Update 3($c$), 3($d$), and 3($e$) by deleting rows and columns of PROB, SAHR, AHRSYN, and AHRCOM corresponding to the screened out groups of variables.
   (h) Supply the screened groups of predictors: $G_1, G_2, \cdots, G_s$.
4. **Forming candidate phalanxes :**
   (a) Find $r_{\min} = \min_{ij}\{\text{AHR}_{\overline{ij}}/\text{AHR}_{ij}\}; i = 1, 2, \cdots, (s-1), j = (i+1), \cdots, s$.
   (b) If $r_{\min} < 1$, block the groups of variables $\{G_i, G_j\}|r_{\min}$ and $s \leftarrow (s-1)$.
   (c) Redo the steps 3($c$), 3($d$), and 3($e$) specific to the index of $r_{\min}$.
   (d) Repeat 4($a$) to 4($c$) until $r_{\min} \geq 1$ OR $s = 1$.
   (e) Set $c \leftarrow s$ and supply candidate phalanxes: $\text{PX}_1, \text{PX}_2, \cdots, \text{PX}_c$.
5. **Screening candidate phalanxes :**
   (a) Screen the $i$th candidate phalanx out if $\max[\text{AHR}_i, E(\text{AHR}_R) + \text{AHR}_{\overline{ij}} - \text{AHR}_j] < \text{AHR}_{iquant}\ \forall\ j \neq i = 1, \cdots, c$.
   (b) Return $\text{PX}_{(1)}, \text{PX}_{(2)}, \cdots, \text{PX}_{(p)}$: the army of phalanxes.

phalanxes form our army of phalanxes. Notice that we no longer check whether two candidate phalanxes should be merged since we did an exhaustive search for merging predictor groups in the previous step of this algorithm. The output of this step is the army of phalanxes: $\mathrm{PX}_{(1)}, \mathrm{PX}_{(2)}, \cdots, \mathrm{PX}_{(p)}$.

## 3.5 Computational Complexity

Now I show the *computational complexity* of our algorithm in terms of the maximum number of random forests that needs to be grown. In the worst case scenario there is *no initial grouping* of the $D$ feature variables.

**Screening phase:** For the screening phase, we fit a total of $D(D+1)/2$ random forests. In fact, first we fit $D$ random forests, i.e., one random forest for each feature variable. Second, for all possible pairs of feature variables we fit $D(D-1)/2$ random forests, i.e., one random forests for each pair. Hence, in total we have $D(D+1)/2$ random forests. Again, in the worst case scenario there is no screened out feature variables.

**Phalanx formation phase:** For the phalanx formation phase, we fit a maximum of $D(D-1)/2$ random forests. In fact, first we fit *one* random forest to the merged pair of feature variables; then we fit $D-2$ random forests by pairing each of the $D-2$ feature variables with the merged pair. In this iteration, we fit a total of $D-1$ random forests. Note that we do not need to grow random forests for all possible pairs. In the next iteration, we fit $D-2$ random forests, and so on. In the worst case scenario we would have one phalanx at the end and that gives us a total of $D(D-1)/2$ fitted random forests.

Adding the numbers in the two phases, we obtain a maximum of $D^2$ random forests. Thus, the computational complexity of our algorithm is of order $\mathcal{O}\left(D^2\right)$. In this chapter, most of the datasets are high-dimensional in terms of the number of feature variables. Hence, the computational burden is greatly reduced by forming initial groups of the binary feature variables. Moreover, the computational burden is reduced by using many processors in *parallel computation*.

## 3.6 Ensemble of Phalanxes

We fit $p$ random forest classifiers using $p$ phalanxes and obtain probabilities of activity from them. Each of those random forests is grown using the default settings of the **R** package **randomForest** (Liaw and Wiener, 2011). The $p$ random forest classifiers are aggregated together to form the ensemble of phalanxes, denoted by EPX, by averaging probabilities of activity across the $p$ random forests/phalanxes.

## 3.7 Results

We apply the algorithm of phalanx formation to the five descriptor sets of the four assays. For each descriptor set we get an army of phalanxes and ensemble them to form EPX. The results of EPX are compared with RF and RRF. The RF and RRF are constructed using the default setup of the R packages **randomForest** (Liaw and Wiener, 2011) and **RRF** (Deng, 2012).

### 3.7.1 Assay AID348

The algorithm of phalanx formation was applied to the five descriptor sets separately. For a particular descriptor set, the algorithm is run for 3 times with 3 different random seeds. The first column of Table 3.1 identifies the descriptor set, the second column identifies the random runs and the last 5 columns correspond to the 5 main steps of our algorithm (see Figure 3.2). For example, looking at the first row in the body of the table, the descriptor set AP has a total of 367 variables arranged into 75 initial groups of which 22 survive screening. The 22 groups are aligned into 4 candidate phalanxes of which 2 survive screening. The screened phalanxes form the army of phalanxes and are ensembled to get EPX. Rows 2 and 3 show the results for the other two runs. We observe some random variation from run to run but we will see below that performance remains pretty stable.

Table 3.1: The number of variables, initial groups, screened groups, candidate phalanxes, and screened/army of phalanxes for the 3 runs of the algorithm to AID348 assay.

| DS | Run | Number of | | | | |
|---|---|---|---|---|---|---|
| | | Variables | Groups | | Phalanxes | |
| | | | Initial | Screened | Candidate | Screened |
| AP | 1 | 367 | 75 | 22 | 4 | 2 |
| | 2 | | | 19 | 8 | 5 |
| | 3 | | | 22 | 8 | 4 |
| BN | 1 | 24 | 24 | 24 | 8 | 8 |
| | 2 | | | 24 | 9 | 9 |
| | 3 | | | 24 | 4 | 4 |
| CAP | 1 | 1795 | 455 | 398 | 13 | 10 |
| | 2 | | | 128 | 8 | 8 |
| | 3 | | | 352 | 17 | 12 |
| FP | 1 | 570 | 101 | 24 | 6 | 4 |
| | 2 | | | 22 | 5 | 4 |
| | 3 | | | 22 | 5 | 5 |
| PH | 1 | 120 | 21 | 5 | 1 | 1 |
| | 2 | | | 5 | 3 | 2 |
| | 3 | | | 5 | 1 | 1 |

A large number of binary predictors are screened out for the four descriptor sets based

on binary variables: AP, CAP, FP, and PH. For example, in the case of AP, between 71% and 75% of the initial groups are dropped. The ranges for the other binary descriptor sets are 12% to 72%, 76% to 79%, and 77%, respectively. All of the continuous predictors of BN are useful (none is dropped) in our 3 runs. The initial screening is more unstable for CAP, perhaps due to the large number of variables.

We repeat balanced 10-fold cross-validation experiments 16 times to obtain 16 AHR's for EPX, RF and RRF. The average hit rates are reported in Table 3.2. The average hit rates for EPX are much larger than those of RF and RRF, three times larger in the case of CAP, the predictor with the largest number of variables. For the binary descriptor sets AP, CAP, FP and PH – where screening variables is dominant – RRF outperforms RF. But for BN, where there is no filtering, RF outperforms RRF. The last two columns in Table 3.2 shows that EPX consistently beats RF and RRF in all our cross-validation experiments.

Table 3.2: Average hit rate (AHR) for an ensemble of phalanxes (EPX), a random forests (RF), and a regularized random forests (RRF) averaged over 16 repeats of balanced 10-fold cross-validation for the AID348 assay. Larger AHR values are better. The last two columns show the number of times EPX has larger AHR among the 16 repeats of cross-validation relative to RF and RRF.

| DS | Run | Mean AHR | | | EPX beats | |
|---|---|---|---|---|---|---|
| | | EPX | RF | RRF | RF | RRF |
| AP | 1 | 0.182 | 0.063 | 0.081 | 16/16 | 16/16 |
| | 2 | 0.194 | | | 16/16 | 16/16 |
| | 3 | 0.146 | | | 16/16 | 16/16 |
| BN | 1 | 0.143 | 0.090 | 0.078 | 16/16 | 16/16 |
| | 2 | 0.153 | | | 16/16 | 16/16 |
| | 3 | 0.132 | | | 16/16 | 16/16 |
| CAP | 1 | 0.201 | 0.068 | 0.090 | 16/16 | 16/16 |
| | 2 | 0.184 | | | 16/16 | 16/16 |
| | 3 | 0.155 | | | 16/16 | 16/16 |
| FP | 1 | 0.157 | 0.077 | 0.098 | 16/16 | 16/16 |
| | 2 | 0.130 | | | 16/16 | 16/16 |
| | 3 | 0.157 | | | 16/16 | 16/16 |
| PH | 1 | 0.108 | 0.070 | 0.080 | 16/16 | 16/16 |
| | 2 | 0.108 | | | 16/16 | 16/16 |
| | 3 | 0.108 | | | 16/16 | 16/16 |

Figure 3.3 shows the box plots for the 16 AHR's. The three boxplots for EPX correspond to the three runs – with different random seeds – in our cross-validation experiment. Notice that, despite exhibiting some run-to-run variability EPX consistently outperforms RF and RRF. We could stabilize the EPX's performance by using random forests with a larger number of trees at the phalanx formation stage. But this change would not necessarily improve the performance and increase the computational burden of the algorithm.

To further illustrate the performance of EPX we used the balanced 10-fold cross-validation number 1 to draw the hit curve (see Figure 3.4). We choose $t = 300$ which comprises 6% of the compounds of Assay AID348.

Figures 3.1 and 3.4 display the hit curves, descriptor set AP in Figure 3.1 and the other four descriptors sets in Figure 3.4. In all the cases the hit curve for EPX start rising very quickly and dominates the other two curves at every $t \in [1, 300]$.

Although we form the phalanxes by optimizing average hit rates, the army of phalanxes also shows very good performance regarding Initial Enhancement (IE), as shown in Table 3.3, which reports the results from the 16 replications in our balanced cross-validation results for assay AID348.

Table 3.3: The initial enhancement (IE) – averaged over the 16 balanced 10-fold cross-validations – for the three ensembles RF, RRF, and EPX of the five descriptor sets of AID348 Assay.

| Ensembles | AP | BN | CAP | FP | PH |
|---|---|---|---|---|---|
| RF | 5.19 | 6.62 | 7.16 | 7.07 | 5.53 |
| RRF | 5.80 | 6.25 | 6.83 | 7.28 | 5.78 |
| EPX | 6.27 | 8.80 | 8.20 | 8.40 | 6.44 |

### 3.7.2 Assay AID362

In this assay, we observed smaller percentage of screening weak variables than the AID348 Assay. The results for screening and phalanx formation are shown in Table A.1. For the binary descriptor sets AP, CAP, FP and PH, the percentages of screened out initial groups are $25-36\%$, $11-14\%$, $5-10\%$ and $14-24\%$, respectively. As before, none of the continuous variable of BN is filtered. The largest descriptor set CAP tends to supply many phalanxes: $10 - 14$ screened phalanxes form the final army. A good number of phalanxes $(4 - 6)$ are found for BN and FP. The descriptor sets AP and PH show moderate $(1 - 4)$ to small $(1 - 2)$ number of phalanxes, respectively, to ensemble.

Table A.2 shows the mean AHRs, averaged over 16 repeats of the balanced 10-fold cross-validation, for the three ensembles EPX, RF, and RRF. For the largest descriptor set CAP, the ensemble EPX shows the largest improvement over RF. The second largest improvement over RF is observed in the descriptor set FP, followed by BN and AP. For PH, our ensemble could not improve much, but didn't perform worse, than RF. The ensemble RRF underperforms RF in all of the five descriptor sets. RRF underperforms RF with the largest margin in CAP, where our ensemble shows the largest improvement.

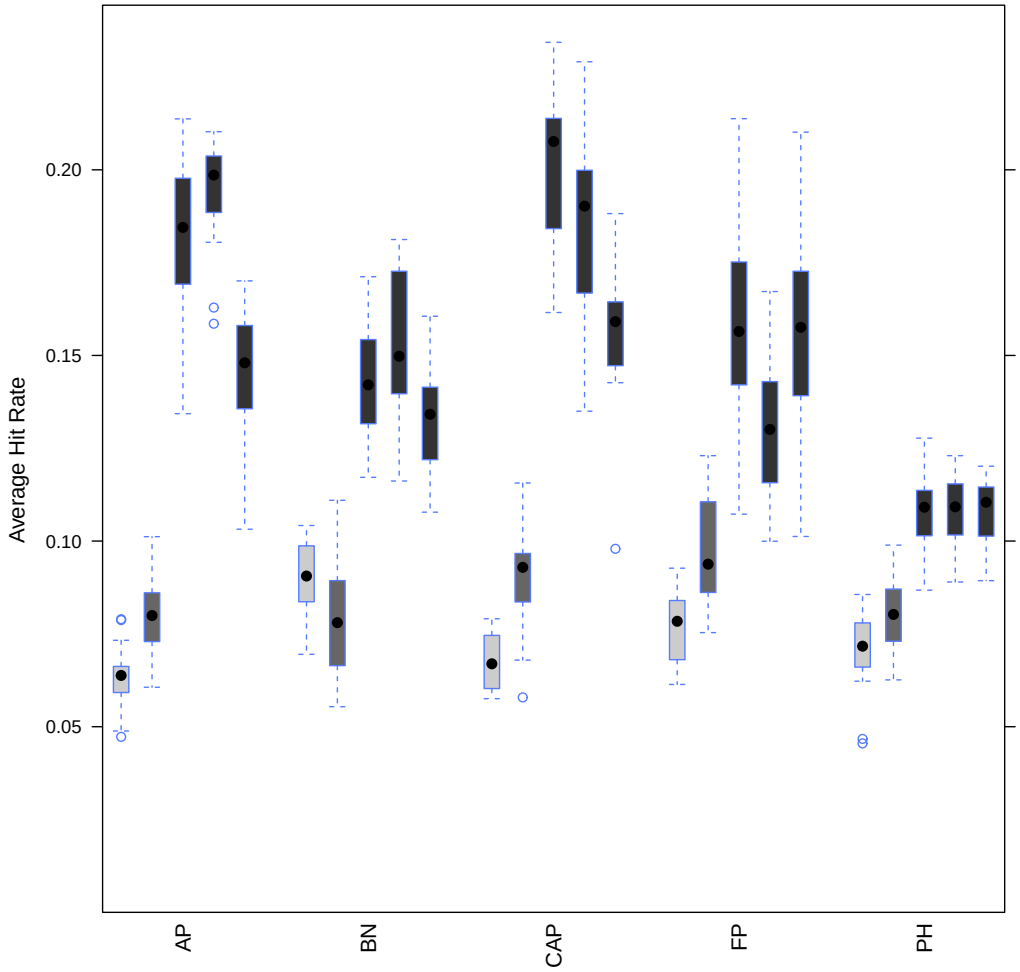
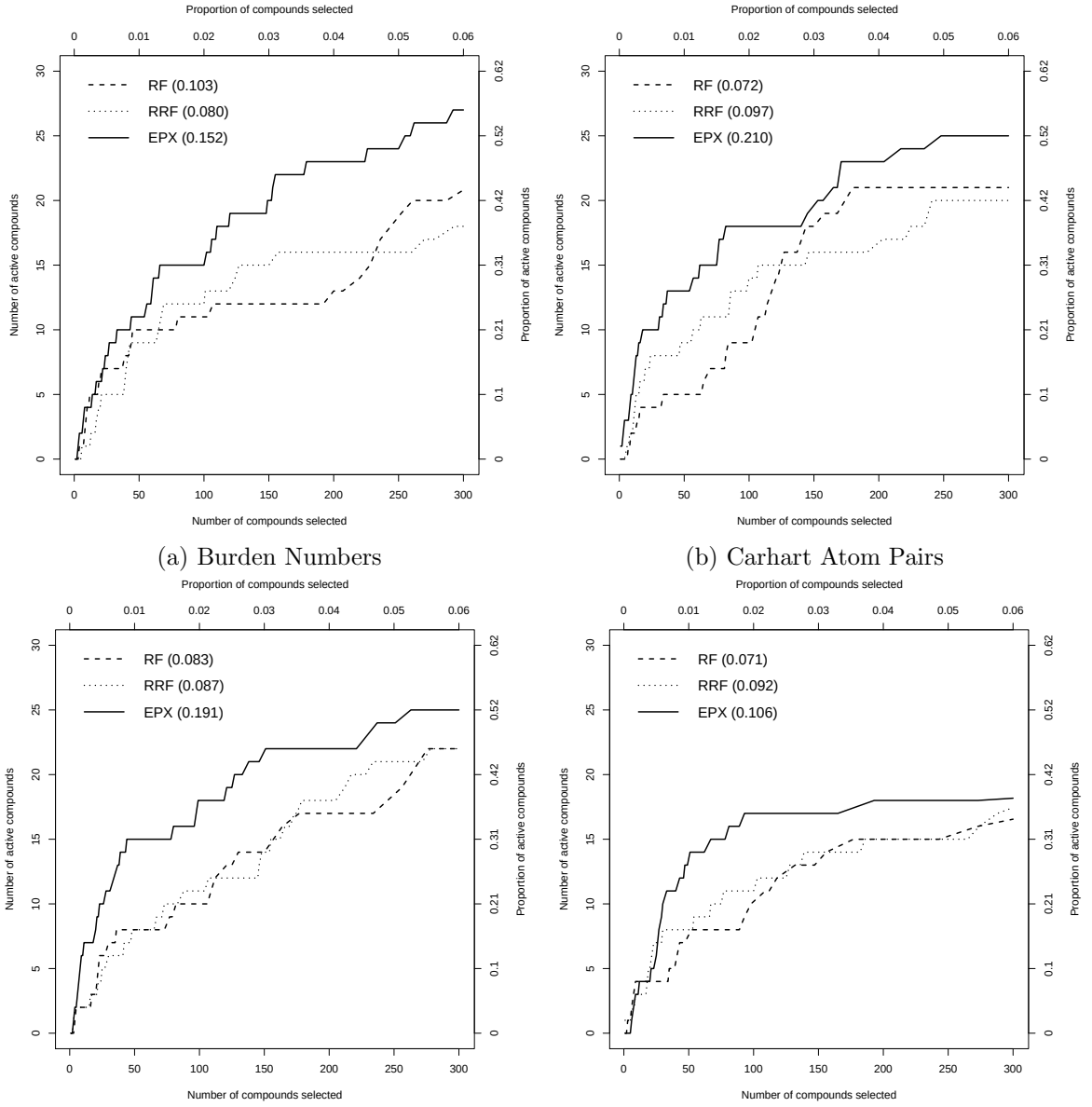



Figure 3.3: Box-plots of average hit rates (AHR) from 16 repeats of cross-validation for a random forests (RF), a regularized random forests (RRF), and for 3 ensembles of phalanxes (EPX) for the five descriptor sets (AP, BN, CAP, FP, and PH) of AID348 assay. The boxes for RF, RRF, and EPX are marked by light-grey, grey and dark-grey colour schemes.




(a) Burden Numbers

(b) Carhart Atom Pairs

(c) Fragment Pairs

(d) Pharmacophores

Figure 3.4: Hit curves for random forests (dashed line), regularized random forests (dotted line), and army of phalanxes (solid line) for the four descriptor sets, BN in panel (a), CAP in panel (b), FP in panel (c) and PH in panel (d). The number in () is the corresponding Average Hit Rate. Data from the AID348 assay.

### 3.7.3 Assay AID364

Like AID362 Assay, we observed roughly the similar proportions of filtering. We also observed roughly the similar room, represented through the number of screened phalanxes, for ensembling. The largest and the smallest descriptor sets tend to supply the most and the least numbers of screened phalanxes, respectively, in the final army. For details, please see the Table A.3.

Table A.4 shows the mean AHRs for EPX, RF and RRF. The largest improvement of EPX over RF is observed in CAP, followed by BN, AP, FP and PH. The highest and the lowest improvements correspond to the largest descriptor set CAP and the smallest binary descriptor set PH, respectively. The ensemble RRF under-performs RF in all of the five descriptor sets. The margin of underperformance of RRF over RF is the highest for the largest descriptor set CAP.

### 3.7.4 Assay AID371

For the binary descriptor sets, the algorithm filters larger and smaller proportions of initial groups than {AID362, AID364} and AID348 Assays, respectively. None of the continuous Burden Numbers is filtered. The largest descriptor set CAP tends to supply the largest number $(6-9)$ of screened phalanxes. The other descriptor sets, including PH, supply a good number $(3-5)$ of screened phalanxes. For details, please see Table A.5.

Table A.6 shows the results of EPX, RF and RRF. The largest improvement of EPX over RF is observed in CAP, followed by AP, BN and PH, respectively. For the 3 runs in FP, the ensemble EPX slightly outperforms RF in 1 run, and underperforms in 2 runs. For this descriptor set, the ensemble EPX outperforms RRF in all 3 runs. The RRF underperforms the RF in AP, BN, CAP and FP, and outperforms in PH.

## 3.8 Diversity map

We focus our attention to answer why EPX ranks the active compounds so well. To begin with, let us recall the work of Breiman (2001), where he pointed out that a strong ensemble needs to have strong and low correlated constituent classifiers. The stronger the classifiers are the stronger the ensemble. The smaller the correlation between predictions of the constituent classifiers is the better the performance of the ensemble.

To proceed, we choose an army of 8 phalanxes from a run of the algorithm 'phalanx formation' to BN of AID348 Assay. We examine how the 8 phalanxes and their ensemble EPX rank the 48 active compounds. We obtain probabilities of activity from a cross-validation, and plot the ranks of the 48 active compounds. Figure 3.5 shows the ranks of the active compounds by the 8 phalanxes each and by their ensemble EPX as well. The darker the colour is the earlier the active is found in the ranked list. We also compute the average hit rates for the 8 phalanxes and for the ensemble as well. In the vertical axis we sequence the

48 active compounds using the probabilities of activity from EPX. On the left hand side, we plot the scale of the ranks of the active compounds.

We see different mass of colour to active compounds across the 8 phalanxes. In words, the phalanxes rank different sets of actives well. If one phalanx misses an active compound, then other phalanxes rank it well and eventually their ensemble ranks the active well. Such behaviour of ranking different sets of actives is called diversity across phalanxes. Having seen the results, we say our algorithm is successful in producing diverse set of phalanxes. Diversity is a similar measure to correlation: the higher the diversity the smaller the correlation.

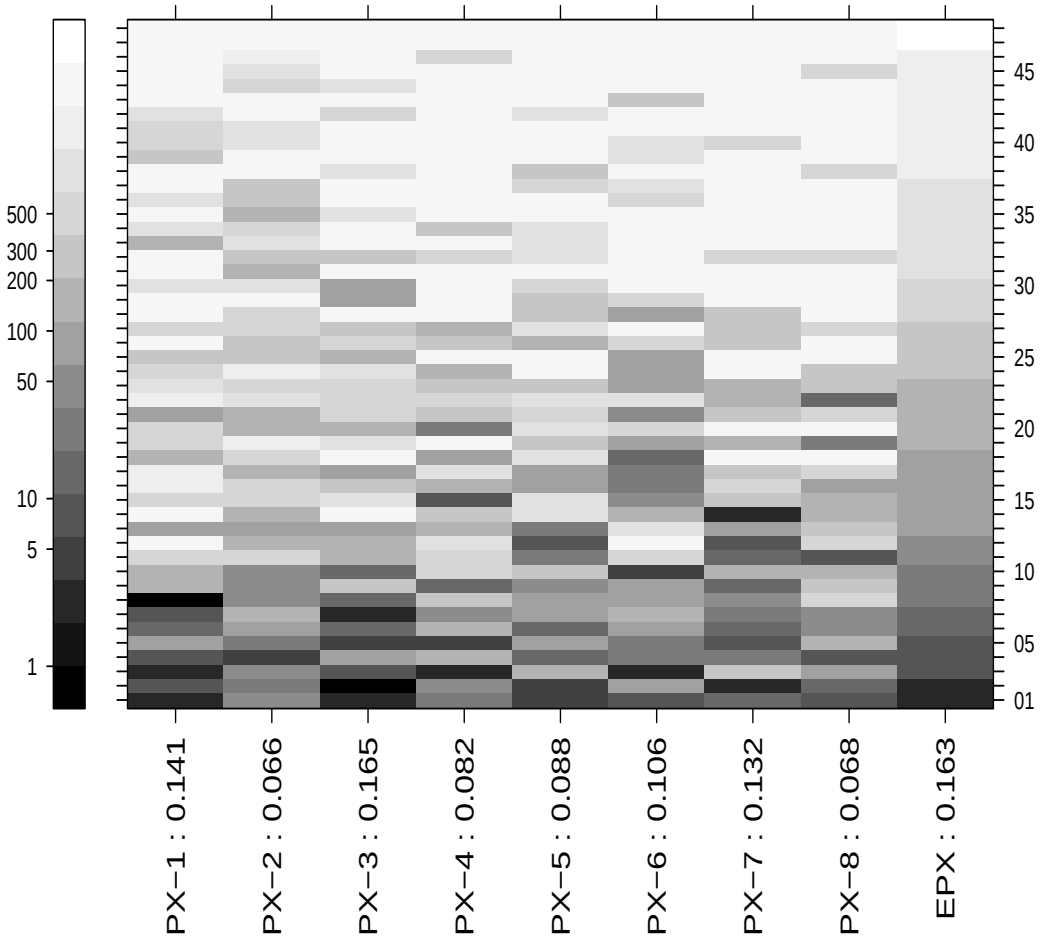



Figure 3.5: Diversity map of ranks of the active compounds by 8 phalanxes from BN of AID348 assay. The average hit rates of a random forest for this cross-validation is 0.103.

In addition to diversity, all of the phalanxes are strong as they have passed the second screening test. Moreover, by checking AHRs of the 8 phalanxes carefully, we find 4 of them are very strong – even stronger than a regular random forest using all of the 24 predictors. For this cross-validation, the random forests with all 24 predictors gives an AHR of 0.103.

The phalanxes 1, 3, 6 and 7 provide larger AHRs than 0.103. This is also true for the other 15 cross-validations. Now we are convinced that our algorithm produces phalanxes that constitute strong classifiers. Equipped with strong and diverse set of classifiers, the ensemble EPX outperforms the benchmark random forests.

For this cross-validation, phalanx 3 provides slightly larger AHR than the overall ensemble EPX. This is not the case for the other cross-validations. In general, the ensemble EPX gives larger AHR than any of the constituent phalanxes.

## 3.9 Summary

In this Chapter, we used four bioassays each with five sets of molecular descriptors. As we applied our ensemble to each descriptor set, there are 20 datasets in a sense. Our ensemble, in all of its 3 runs, clearly outperformed the random forests and regularized random forests in almost every dataset. However, there were a few datasets where our ensemble could not outperform its competitors. We now present those situations case by case. For three runs of EPX in PH of AID362 Assay, we observed larger AHR than RF for a total of 14, 7 and 9 times, respectively, out of 16 cross-validations. In FP of AID371 Assay, EPX produced better AHR than RF for 3, 4, and 12 times, respectively. In the first and second examples, EPX won over RF in 2 and 1 runs, respectively, out of 3 runs. In those two examples, EPX did not lose completely, however. Our ensemble, EPX clearly won against RRF in the 19 datasets out of 20. In PH of AID371 Assay, EPX outperformed RRF in 1 run out of 3.

The highest improvement of EPX over RF was observed for the largest descriptor set CAP. It was CAP for which the largest margin of underperformance of RRF over RF was observed. The story was similar for the continuous descriptor set BN. Note that, the continuous predictor of BN possesses more resolution than any binary predictor. The least improvement of EPX over RF and RRF was observed for the smallest binary descriptor set PH. Given a descriptor set, the most and the least improvements of EPX over RF and RRF were observed for AID348 and AID371 Assays, respectively. The smaller the proportions of actives the larger the improvements of EPX over RF and RRF were. In a nutshell, the harder the problem the better the ensemble of phalanxes performed relative to its competitors RF and RRF.

Some instability in the results of EPX was observed from run to run. Despite this instability, the resulting ensemble consistently outperformed RF and RRF. However, some stability in the results can be achieved by growing a large number of trees ($> 150$, for example) when forming the phalanxes. This change would increase computational complexity, not necessarily the performance, of the ensemble of phalanxes. We plan to tackle such increased computational complexity of our method by coding the algorithm in C/C++ or in Fortran. However, we managed to run our method on fairly large descriptor sets and observed impressive results.

The ensemble EPX is developed by optimizing *average hit rate*. For good performance of EPX in terms of IE, we propose to form the phalanxes by optimizing IE. Moreover, the

extension of this ensemble of phalanxes to classification and regression is straightforward. We need to replace AHR by misclassification error and mean squared error, respectively, to form the phalanxes and to evaluate the ensemble. Our next target is to implement this algorithm for regression and classification.

We have used subject matter knowledge to form initial groups of the binary feature variables. Specifically, a set of feature variables belonging to a particular pair of atoms are grouped together. The goal is to form diverse initial groups: naturally, two sets of feature variables specific to two pairs of atom are diverse to each other. However, such diversity between the initial groups might also be obtained by examining the correlation structures of the feature variables.

On the other hand, the resulting screened phalanxes represent different characteristics of the predictor sets towards the response variable. Such screened phalanxes may also represent multiple mechanism of activity in a dataset. If multiple mechanisms of activity produce different activity classes, the phalanxes can also be formed in such a way that one phalanx represents one activity class which would be ultimately ensembled to provide predictive ranking of the activity classes.

There exist learning algorithms to handle class-imbalances in two-class classification problems through over-sampling of the minority class and under-sampling of the majority class (see Chawla, Lazarevic, Hall, and Bowyer (2003); Chen, Liaw, and Breiman (2004)). The goal of over-sampling of the minority class, for example, is to reduce the increased bias of a classifier towards the majority class and thus to improve prediction accuracy of the minority class. Our method is different from the over or under sampling procedures as it does not correct any bias towards the minority/majority class. Rather, it aims to rank the minority class items at the start of the list.

# Chapter 4

# Ensembling Phalanxes Across and Within Descriptor Sets

## 4.1 Introduction

In Chapter 2, an ensemble of descriptor sets (EDS) is developed aggregating random forests (RF) over several sets of molecular descriptors. Specifically, the chemical compounds in a bioassay dataset are ranked by averaging probabilities of activity from random forests applied to the five sets of molecular descriptors. The top ranked compounds are shortlisted in a way that the most, if not all, of the rare active compounds are found in the shortlist. The developed ensemble performs better than the top performing random forests that uses a single descriptor set. Moreover, for the most part, our ensemble EDS outperforms random forests applied to the pool of descriptor sets (AF). However, for Assay AID348, random forests with the pool of descriptor sets outperforms our ensemble of descriptor sets.

There are five molecular descriptor sets for each of the four bioassays (see Sections 2.2 and 2.3 of Chapter 2). All of the five sets of descriptors are considered rich in variables as they contain a large number of predictors. In the process of developing a good ranking model, we use the richness of variables in the descriptor sets. The ranking model, called the ensemble of phalanxes (EPX), is developed in Chapter 3. First, the data-adaptive phalanxes are formed in a descriptor set by grouping predictor variables together. The variables in a phalanx are good in terms of predictive ranking when used together in a model, and the variables in different phalanxes are good in separate models. The resulting phalanxes are aggregated growing a random forest in each and averaging probabilities of activity across the phalanxes.

The performance of EPX is better than the random forests (Breiman, 2001) and regularized random forests (Deng and Runger, 2012). EPX performs very well when there are many variables in a descriptor set and when the proportions of active compounds are small. The harder the problem the better the ensemble of phalanxes performs compared to the alternative procedures such as random forests.

In this chapter, the ensembles of phalanxes (EPX) are applied to the five descriptor sets replacing random forests to form an improved version of EDS. Specifically, the probabilities of activity obtained from applying EPX to each of the five descriptor sets are averaged across the five sets of molecular descriptors. The new ensemble is denoted by EDS-PX.

The ensemble EDS-PX provides better predictive ranking of the rare active compounds

than EDS in all of the four bioassay datasets. Moreover, EDS-PX outperforms random forests applied to the pool of five descriptor sets (AF). Careful investigation reveals that EDS-PX aggregates strong and diverse sets of constituent classifiers to form a highly powerful ranking model.

## 4.2 Datasets and Variables

In this chapter, four bioassay datasets are used which contain a small proportion of active compounds compared to the proportion of inactive compounds. The response variable in each assay is the compounds' activity status where each compound is recorded as either active or inactive against a biological target. The interest is in detecting the active compounds only. The assay datasets are presented in Section 2.2 of Chapter 2.

There are five sets of predictor variables for each assay. Each set of predictors is also known as a descriptor set. The predictors in a set are numeric variables which describe the structure or shape of the chemical compounds in a bioassay dataset. Following alphabetical order the descriptor sets are: Atom Pairs (AP), Burden Numbers (BN) (Burden, 1989; Pearlman and Smith, 1999), Carhart Atom Pairs (CAP) Carhart et al. (1985), Fragment Pairs (FP), and Pharmacophores Fingerprints (PH). The aim is to develop a predictive ranking model which relates the activity status/toxicity/drug potency of the chemical compounds in a bioassay with all the descriptor sets. The descriptor sets are presented in Section 2.3 of Chapter 2.

## 4.3 Ensemble of Descriptor Sets using Data-Adaptive Phalanxes

First, let me explain the process of aggregating the descriptor sets. Let $n_s$ be the number of molecular descriptor sets in an assay. We build $n_s$ classifiers using $n_s$ sets of descriptors and estimate probabilities of activity using each of the classifiers. Finally, the aggregation is performed by averaging probabilities across the classifiers/sets of descriptors. The averaged probabilities of activity are used to rank the compounds. Specifically, the compound with the largest averaged probability of activity is ranked first, followed by the compound with the second largest averaged probability of activity and so on.

In Chapter 2, RF is applied to the five sets of descriptors: Atom Pairs (AP), Burden Numbers (BN), Carhart Atom Pairs (CAP), Fragment Pairs (FP), Pharmacophores (PH). When the probabilities of activity from random forests are averaged across AP, BN, CAP, FP and PH, the resulting ensemble is called the ensemble of descriptor sets or simply EDS.

In this chapter, the ensemble of phalanxes is applied to the five descriptor sets. The probabilities of activity across the five ensembles of phalanxes are averaged to form another ensemble which we call the "ensemble of descriptor sets from data-adaptive phalanxes" or simply EDS-PX. We will show that EDS-PX is an improved version of EDS.

## 4.4 Evaluation of Classifiers

A classifier is evaluated based on how well it positions the rare active compounds at the beginning of a ranked list of compounds. The compounds in a bioassay are ranked using the estimated probabilities of activity from a classifier. Since the activity statuses of all of the compounds in the four bioassays are known, we used cross-validation to mimic the training and test parts. Specifically, the probabilities of activity for the compounds in a bioassay are obtained using a balanced 10-fold cross-validation. Ten random groups (folds) of the compounds, each with approximately equal number of active and inactive compounds, are formed first. Nine such groups are used for training and the other group is left out for testing. Repeating the training and testing procedures ten times, each time leaving one group out for testing, provides the probabilities of activity for all of the compounds in a dataset. The details are presented in Section 2.5.1 of Chapter 2.

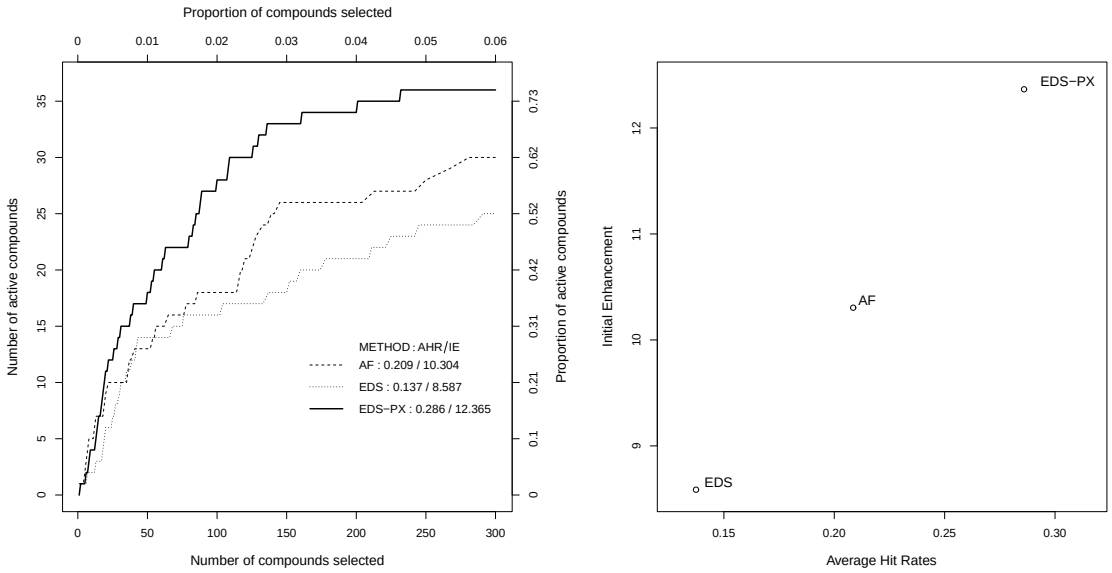
Having estimated the probabilities of activity, the ranking performance of a classifier is evaluated using a hit curve. A hit curve enables visual inspection of the performance of a classifier. A classifier with a high hit curve is preferred. The definition of a hit curve is given in the Subsection 2.5.2 of Chapter 2.

The comparison of the performances of many classifiers using hit curves is not well defined if the curves cross each other. Such comparison might become very hard if the classifiers are to be evaluated repeatedly. To automate such comparison, we summarize a hit curve by a single number. The metric average hit rate (AHR) summarizes a hit curve by a single number and facilitates comparison. AHR varies from 0 to 1, where larger values imply better ranking of the rare active compounds. The definition of AHR is presented in Subsection 2.5.4 of Chapter 2.

Initial enhancement (IE) is also a single number summary of a hit curve evaluated at a particular cut-off point. An IE of 1 implies performance similar to random ranking. Larger values of IE imply better predictive ranking. The definition of IE is presented in Subsection 2.5.3 of Chapter 2.

## 4.5 Results

This section contains results after applying three ensembles to the four bioassay datasets. The three ensembles are: (1) random forests applied to the pool of five descriptor sets (AF), (2) ensemble of descriptor sets using random forests (EDS) as in Chapter 2, and (3) ensemble of descriptor sets using data-adaptive phalanxes (EDS-PX). The results for EDS-PX are obtained from run 1 of EPX presented in Chapter 3.




(a) (b)

Figure 4.1: Hit curves comparing three ensembles – AF, EDS and EDS-PX – for the AID348 assay dataset (panel $a$). Initial enhancement (IE) versus average hit rate (AHR) plot for AF, EDS and EDS-PX (panel $b$). The results are obtained from the estimated probabilities of activity using a balanced 10-fold cross-validation.

### 4.5.1 Assay AID348

I start showing the results of AID348 Assay for two reasons: in this assay (i) EDS appeared weak compared to AF (see Section 2.6.4 of Chapter 2), and (ii) EDS-PX improves the performance of EDS the most. Panel ($a$) of Figure 4.1 shows the three hit curves corresponding to the three ensembles: AF, EDS and EDS-PX. To plot the hit curves I used the probabilities of activity from a balanced 10-fold cross-validation of the compounds of the AID348 assay. As this assay contains only 48 active compounds, I computed IE at 300 shortlisted compounds following the results of Hughes-Oliver et al. (2011). We see that the top performing ensemble in terms of high hit curve is EDS-PX which clearly dominates the other two ensembles, AF and EDS.

For a straightforward comparison of the methods, I have plotted in panel ($b$) of Figure 4.1 IE against AHR for the same balanced 10-fold cross-validation of the AID348 dataset. As the goal is to maximize both AHR and IE, I want our method in the top right corner of this plot. Indeed, EDS-PX is found in the top right corner, followed by AF and EDS, respectively.

Now we will show that the performance of our ensemble is consistent over many cross-validations. Thus, to compare average performances of the ensembles, we repeated the balanced 10-fold cross-validation for a total of 16 times. There are 16 processors available in our department's computing networks for parallel computation, one processor is used for one cross-validation, and hence the 16 cross-validations. The bivariate mean vectors ($\overline{\text{AHR}}$, $\overline{\text{IE}}$) corresponding to the three ensembles are compared. The 95% confidence band for the bi-

variate mean vector is constructed employing the bivariate normality and homogeneity of covariance assumptions. Panel ($a$) of Figure 4.2 shows the plots of mean AHR versus mean IE with 95% confidence bands for AF, EDS and EDS-PX applied to the AID348 assay. In terms of both the mean AHR and mean IE, the top performing ensemble is EDS-PX followed by AF and EDS. The performances are significantly different from one to another.

Next we will compare the performances of the ensembles at the cross-validation level. In a particular cross-validation, we fitted AF, EDS and EDS-PX and computed their AHRs and IEs. The process is repeated 16 times, enabling us to perform pairwise comparison between two methods. Column 3 of Table 4.1 shows the mean AHRs and mean IEs and the winning rates of one ensemble over another for Assay AID348. In terms of AHR and IE, AF beats EDS for 16 out of 16 cross-validations. In terms of both AHR and IE, EDS-PX beats both AF and EDS a total of 16 out of 16 times.

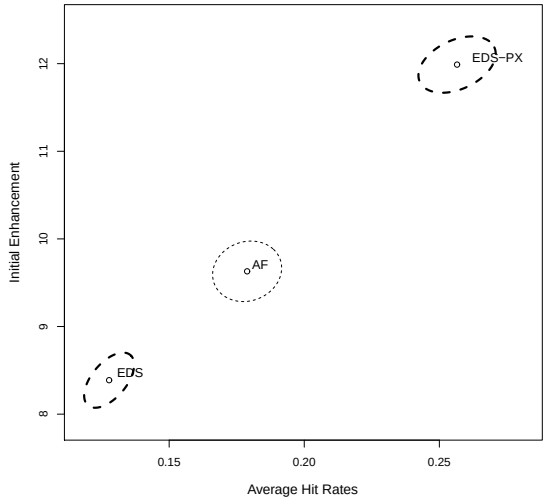
### 4.5.2 Assay AID362

EDS-PX again improves the performance of EDS for both AHR and IE. The improvement is particularly large for AHR. Panel ($b$) of Figure 4.2 shows the plots of mean IE versus mean AHR for the three ensembles applied to AID362. The ensemble EDS-PX is found in the top right corner followed by EDS and AF. Hence, in this assay the top performing ensemble is EDS-PX.

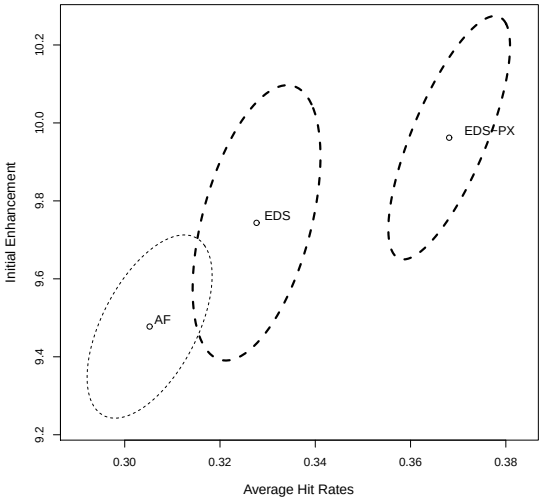
Let us compare the performances of the ensembles at the cross-validation level. Column 4 of Table 4.1 shows the mean AHRs and mean IEs and the winning rates of one ensemble over another. In terms of AHR, EDS wins over AF a total of 15 out of 16 cross-validations, and EDS-PX beats both EDS and AF a total of 16 out of 16 cross-validations. In terms of IE, which is calculated at 300 shortlisted compounds, EDS beats AF a total of 14 times, and EDS-PX wins over AF and EDS a total of 16 and 14 times, respectively, out of 16 cross-validations.

### 4.5.3 Assay AID364

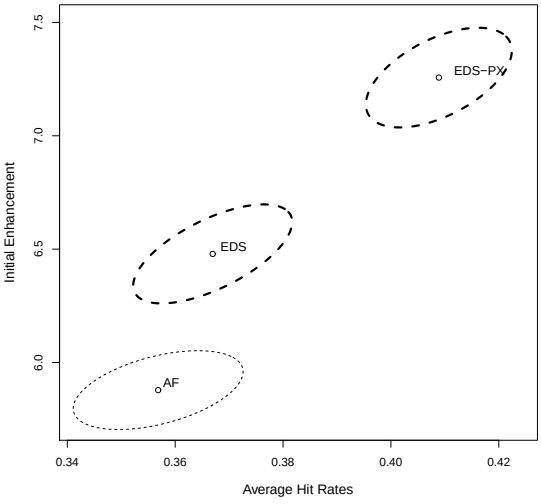
The ensemble EDS-PX shows large improvement over EDS in terms of both AHR and IE. In this assay the IE is computed at 300 shortlisted compounds. Panel ($c$) of Figure 4.2 shows the plots of mean IE versus mean AHR for the three ensembles. Following the diagonal line from the top right corner the ensembles are sequenced as EDS-PX, EDS and AF. Thus the top performer is EDS-PX, and its performance is much larger than the performances of EDS and AF. In column 5 of Table 4.1, we see that EDS-PX beats EDS and AF a total of 16 times out of 16 cross-validations using both evaluation metrics AHR and IE.



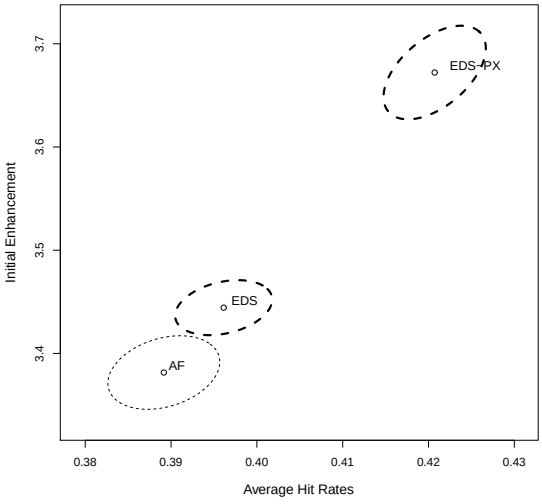

(a) Assay AID348




(b) Assay AID362




(c) Assay AID364




(d) Assay AID371

Figure 4.2: Plots of mean initial enhancement ($\overline{\text{IE}}$) versus mean average hit rate ($\overline{\text{AHR}}$) with 95% confidence regions for four assay datasets: AID348, AID362, AID364, and AID371.

### 4.5.4 Assay AID371

There are 278 active compounds in this assay and the IE is computed at 600 shortlisted compounds. The performance of EDS-PX is far better than the performances of EDS and AF, see panel (*d*) of Figure 4.2. Column 6 of Table 4.1 shows that the ensemble EDS-PX beats EDS and AF a total of 16 times out of 16 cross-validations.

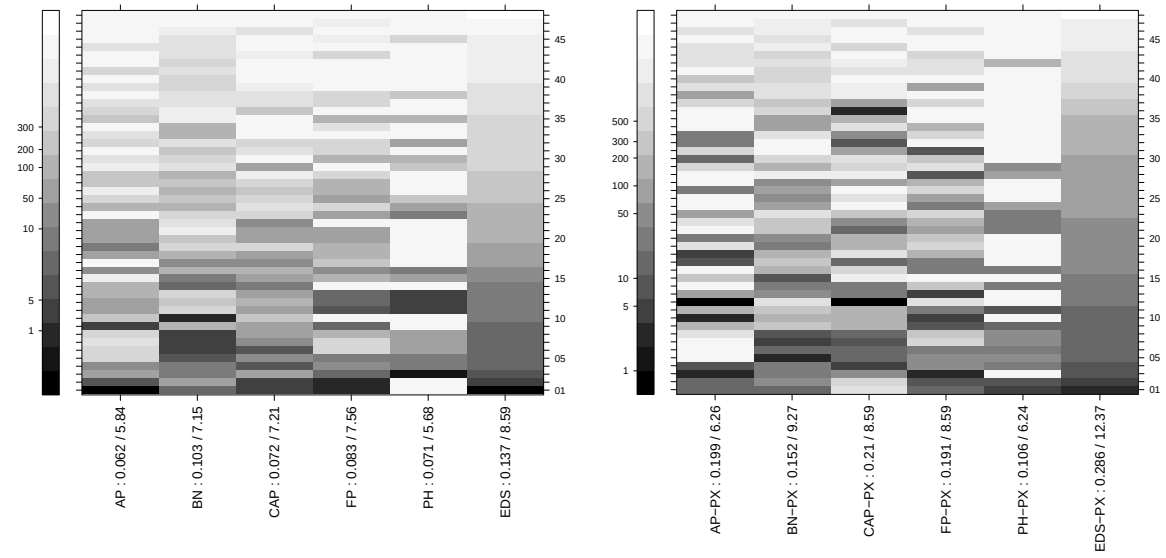
Table 4.1: Mean AHR and IE for AF, EDS, and EDS-PX for the four assay datasets. Winning rates of EDS over AF, and EDS-PX over AF and EDS are also given for AHR and IE.

| Assay Datasets | | AID348 | AID362 | AID364 | AID371 |
|---|---|---|---|---|---|
| Mean AHR | AF | 0.179 | 0.305 | 0.357 | 0.389 |
| | EDS | 0.128 | 0.328 | 0.367 | 0.396 |
| | EDS-PX | 0.257 | 0.368 | 0.409 | 0.421 |
| Larger ($\geq$) AHR | EDS versus AF | 0/16 | 15/16 | 14/16 | 15/16 |
| | EDS-PX versus AF | 16/16 | 16/16 | 16/16 | 16/16 |
| | EDS-PX versus EDS | 16/16 | 16/16 | 16/16 | 16/16 |
| Mean IE | AF | 9.630 | 9.478 | 5.878 | 3.381 |
| | EDS | 8.388 | 9.744 | 6.479 | 3.444 |
| | EDS-PX | 11.989 | 9.962 | 7.257 | 3.672 |
| Larger ($\geq$) IE | EDS versus AF | 0/16 | 14/16 | 16/16 | 12/16 |
| | EDS-PX versus AF | 16/16 | 16/16 | 16/16 | 16/16 |
| | EDS-PX versus EDS | 16/16 | 14/16 | 16/16 | 16/16 |

It was shown in Section 2.6 of Chapter 2 that the aggregate of random forests over the five descriptor sets (EDS) outperformed the top performing random forest applied to any of the single set of descriptors. Moreover, in three of the four bioassays, EDS outperformed the random forest applied to the pool of descriptor sets (AF). In this chapter, we have shown that the aggregated ensemble of phalanxes over the five descriptor sets (EDS-PX) not only outperforms AF but also improves over EDS. However, to check the strengths and diversity of the constituent classifiers of EDS and EDS-PX, we plot a diversity map next in Section 4.6.

## 4.6 Diversity Map

In this section, we will try to better understand why EDS-PX outperforms EDS. As such, the ranks of the 48 active compounds in the AID348 Assay are plotted using EDS and EDS-PX (left and right panels of Figure 4.3, respectively). The ranks are obtained from the probabilities of activity corresponding to the cross-validation number 1. The ranks of the active compounds are also plotted for the constituents random forests of EDS (AP, BN, CAP, FP and PH), and the constituents ensembles of phalanxes of EDS-PX (AP-PX, BN-PX, CAP-PX, FP-PX and PH-PX). If an active compound is ranked first by a classifier, the compound




(a) EDS (b) EDS-PX

Figure 4.3: Diversity maps of EDS (left panel) and EDS-PX (right panel) for the ranks of the active compounds in AID348 assay. The compounds are ranked using the probability of activity corresponding to the cross-validation number 1.

receives the darkest grey colour. If a compound is ranked down in the sequence, it receives absolutely no colour. The colour key on the left of each panel shows where in the sequence an active compound is ranked by a classifier. In the left and right panels, the active compounds are sequenced using the probabilities of activity from EDS and EDS-PX, respectively. We report the AHRs and IEs for all of the constituent classifiers and their ensembles to understand how strong the classifiers are.

We see variation in ranks across the constituent classifiers of EDS: AP, BN, CAP, FP and PH, i.e., the five random forests rank different sets of active compounds well. The story is similar among the constituent classifiers of EDS-PX. However, the constituents of EDS-PX show more diversity, reflected by the contrast in colours among the five ensembles of phalanxes, than the constituents of EDS. Moreover, the constituents of EDS-PX are much stronger than the constituents of EDS. Hence, equipped with stronger and more diverse sets of constituent classifiers, EDS-PX outperforms EDS.

## 4.7 Summary

In Chapter 2, we developed an ensemble by averaging probabilities of activity from random forests applied to the five molecular descriptor sets to rank rare active compounds ahead of the majority inactive compounds in four bioassay datasets. The ensemble of descriptor sets (EDS) provides better predictive ranking of the rare active compounds than the most accurate random forests applied to any of the single set of molecular descriptors. For the most part, in

3 out of 4 bioassays, the ensemble of descriptor sets outperforms the random forests applied to the pool of the five descriptor sets. It is the diversity and strengths of the random forests across the descriptor sets which make EDS a good ranking model.

We define phalanxes as the groups of predictors, where the variables in a phalanx are good, in predicting the rare actives, when used together in a model and the variables in separate phalanxes are good in separate models. Having defined the term phalanx, we label the sets of molecular descriptors as the natural phalanxes. We then pose the question whether there are data-adaptive phalanxes in variable-rich descriptor sets. Such thinking motivates me to develop an algorithm (Algorithm 3.1 in Chapter 3) to unmask data-adaptive phalanxes in a descriptor set. Finally, the ensemble of phalanxes (EPX) is developed by fitting a random forest in each phalanx and averaging probabilities of activity across the phalanxes. Equipped with strong and diverse set of phalanxes, EPX outperforms regular random forests applied to a descriptor set.

In this chapter, we have aggregated the ensembles of phalanxes over the five descriptor sets and compared the resulting EDS-PX with the ensemble of descriptor sets (EDS) and random forests applied to the pool of descriptor sets (AF). In all of the four bioassays, the ensemble EDS-PX outperforms EDS and AF with a big margin. Our method EDS-PX aggregates strong constituent classifiers to produce a powerful committee capitalizing on the data-adaptive and natural diversities within and across descriptor sets, respectively.

The algorithm of phalanx formation could also be applied to the pool of the five descriptor sets in each bioassay. The five sets of initial groups obtained from the five descriptor sets could also be combined into a single set of initial groups to run the algorithm. I did not attempt to do so because of the large computational burden of our algorithm to such a unified set of predictors. However, it is possible to run a group lasso (Meier et al., 2008; Yuan and Lin, 2007) like algorithm to the unified initial groups to obtain a set of screened initial groups through regularization. If the number of screened initial groups drops down to a manageable figure then the algorithm of phalanx formation might be applied to the set of screened initial groups.

# Chapter 5

# Protein Homology: Ensembling via Logistic Regression Models

## 5.1 Introduction

Ensembles are applied to the *protein homology* dataset. This dataset was briefly introduced in Section 1.6 and will be described in more detail in Section 5.2. The ensembles are constructed with the training set by predicting the response variable *homology status* using a set of predictor variables. In addition to the training set, this application comes with a *test set* for which we don't know the status of the response variable. But the performance of an ensemble can be evaluated by submitting intermediate results to the KDD Cup website which gives us a chance for honest comparison of the competing ensembles. As before, the goal is the detection of the rare class, more specifically, *ranking rare homologs* of a native protein ahead of the non-homologous candidate proteins.

The original ensemble of phalanxes is developed in Chapter 3 which requires repeated fits of random forests. But random forests itself is fairly computationally demanding and the repeated fits of random forests, as in the ensemble of phalanxes, increases the computational time substantially. In this chapter, we incorporate the popular and computationally less demanding *logistic regression model* to form the phalanxes and to build the ensemble. The aims are: (i) improving the performance of the ensemble of phalanxes in terms of predictive ranking, and (ii) reducing computational burden of the algorithm. Moreover, we want to show that the ensemble of phalanxes is easily adaptable to a *base learner other than random forests*.

As noted in Section 1.3.2, random forests (Breiman, 2001) possesses inherent variable selection properties and is insensitive to the inclusion of a few noise variables. Unlike random forests, the performance of a logistic regression model containing some good variables is hampered by the inclusion of noise variables. Thus, in order to avoid mixing up noise variables with good variables, we introduce an *intermediate step of filtering* noise variables during phalanx formation using logistic regression. Moreover, we use *cross-validation* to form the phalanxes instead of out-of-bag samples. The modified algorithm of phalanx formation will be discussed in Section 5.4.

The training set of the protein homology data contains a total of $145,751$ rows, each representing a candidate protein, and 74 continuous predictors. The formation of phalanxes

with repeated fits of random forests to such a huge training set appeared computationally very expensive. For example, the parallelization of the processors in our department's computing networks is found ineffective to handle this massive computation. Those computational issues are tackled by *parallel execution* of the algorithm of phalanx formation using many computing nodes in a cluster of processors of the *WestGrid* (Western Canada Research Grid) computing network. Moreover, we reduce the *overhead data-transfer* from the remote to local nodes by sending an appropriate chunk of data used at each node through careful coding of our algorithm.

Ensembles of phalanxes using random forests and logistic regression – denoted by EPX(RF) and EPX(LR), respectively – and regular random forests (RF) are applied to this protein homology dataset. This dataset comes with many *blocks* where each block belongs to a native protein to which the homology status is tested for many candidate proteins. This block structure makes this dataset special. The ensembles are learned using the training blocks and are evaluated on the test blocks. The performance of an ensemble is tested in each test block, and the overall performance is obtained by averaging over all of the test blocks. Thus, in order to provide good predictive ranking an ensemble needs to perform well in as many blocks as possible. The proposed ensembles of phalanxes, EPX(RF) and EPX(LR), are found *better* than RF in terms of predictive ranking. Most importantly, EPX(LR) is found *more powerful* and *less computationally demanding* than EPX(RF).

## 5.2 Protein Homology Data

The protein homology dataset is downloaded from the 2004 KDD Cup competition's website (`http://osmot.cs.cornell.edu/kddcup/datasets.html`). The structure of this dataset is presented in Table 5.1. Each block relates to one native protein. Each row (or line) within a block describes a candidate protein which is tested for homology ($y = 0/1$ for $no/yes$) to the block's native protein. The first element of each line is a BLOCK ID that denotes to which native protein this line belongs. There is a unique BLOCK ID for each native protein sequence. For example, the blocks 279 and 48 in Table 5.1 represent the first and last native proteins in the training set, respectively.

Many candidate proteins were tested for homology to a native protein. The second element of each line uniquely describes the corresponding candidate protein. The variable which identifies the candidate protein sequences is called EXAMPLE ID. For example, the candidate proteins with EXAMPLE IDs from 261532 to 262336 belong to the first training block 279.

The third element of a line provides a realization of the response variable, $y$. The response is a class variable known as *homology status*. The candidate proteins that are homologous to the native protein are denoted by 1, and the non-homologous proteins are denoted (i.e., decoys) by 0. The test set (lower part of Table 5.1) provides "?" in this position.

The following elements in a line are realizations of the feature variables: $x_1, x_2, \cdots, x_D$.

Table 5.1: The structure of the protein homology dataset. The top and bottom portions are extracted from the training and test sets, respectively.

| ID | | Response | Predictors | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| BLOCK | EXAMPLE | $y$ | $x_1$ | $x_2$ | $x_3$ | $\cdots$ | $x_{72}$ | $x_{73}$ | $x_{74}$ |
| 279 | 261532 | 0 | 52.00 | 32.69 | 0.30 | $\cdots$ | -0.35 | 0.26 | 0.76 |
| 279 | 261533 | 0 | 58.00 | 33.33 | 0.00 | $\cdots$ | 1.16 | 0.39 | 0.73 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 279 | 262336 | 0 | 46.00 | 27.08 | 1.72 | $\cdots$ | -0.75 | 0.55 | 0.69 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 48 | 43884 | 0 | 87.96 | 25.26 | -0.94 | $\cdots$ | 1.61 | 0.29 | 0.08 |
| 48 | 43885 | 0 | 48.61 | 25.47 | -0.50 | $\cdots$ | 0.67 | 0.08 | 0.09 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 48 | 44825 | 1 | 87.50 | 29.33 | 5.84 | $\cdots$ | -0.58 | 0.16 | 0.23 |
| 153 | 141691 | ? | 71.84 | 23.17 | -0.57 | $\cdots$ | -0.58 | 0.24 | 0.29 |
| 153 | 141692 | ? | 83.50 | 26.09 | 1.50 | $\cdots$ | 0.50 | 0.11 | 0.05 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 153 | 142501 | ? | 50.49 | 24.56 | 0.84 | $\cdots$ | -0.40 | 0.34 | 0.08 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 140 | 129681 | ? | 82.80 | 31.58 | 2.62 | $\cdots$ | 2.64 | 0.14 | 0.34 |
| 140 | 129682 | ? | 67.52 | 25.00 | 1.76 | $\cdots$ | -0.06 | 0.20 | 0.52 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 140 | 130656 | ? | 23.57 | 27.03 | -1.49 | $\cdots$ | 1.05 | 0.00 | -0.30 |

The feature variables represent the similarity or match between the native protein and the candidate protein which is tested for homology. This dataset contains 74 feature variables which will be used to predict the response $y$.

Many native proteins are considered in this dataset. A total of 303 native proteins provide BLOCK IDs running from 1 to 303. After assigning the BLOCK IDs, the organizers of the KDD Cup assigned the blocks to the training and test sets. The training set contains 153 native protein sequences (153 blocks) and the test set contains 150 native protein sequences (150 blocks). For example, the pairs of blocks (279, 48) and (153, 140) in Table 5.1 are the first and last blocks taken from the training and test set, respectively. The homology status for the training set is known to us, and is unknown for the test set, i.e., undisclosed by the KDD cup organizers.

The training and test sets contain $145,751$ and $139,658$ candidate proteins, respectively. The large number of candidate proteins in each set brings computational challenges for our method.

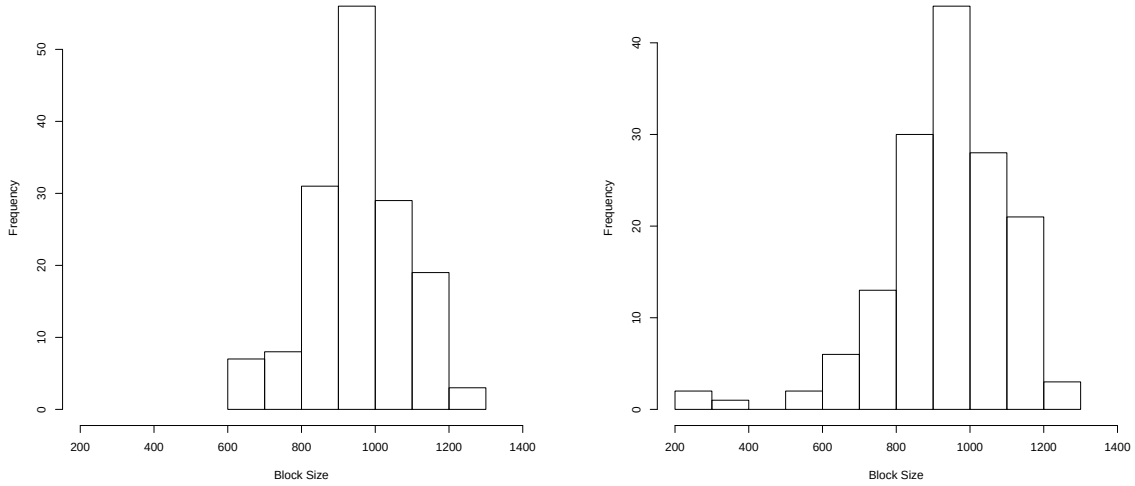
Table 5.2 shows the block sizes for the training and test sets. The 1st quartile, median, 3rd quartile, and the maximum of the block sizes in both of the training and test sets are fairly similar. The minimum block sizes for the training and test sets are 612 and 251 respectively. The test set possesses 3 blocks with sizes 251, 256 and 372, and has a more left skewed distribution; please see the histograms of block sizes in Figure 5.1.

Table 5.2: Block sizes in the training and test sets of the protein homology datasets.

| Datasets | BLOCK SIZE | | | | | |
|---|---|---|---|---|---|---|
| | Min | 1st Qrt | Median | Mean | 3rd Qrt | Max |
| Training | 612 | 859 | 962 | 952.6 | 1048 | 1244 |
| Test | 251 | 847 | 954 | 931.1 | 1034 | 1232 |

Most of the blocks in the training set contain very few homologous proteins. The minimum, median and maximum of the proportions of homologous proteins are: 0.0008, 0.0047 and 0.0581. Figure 5.2 shows the histogram of the proportions in the training set. In this Figure, the 3rd quartile is marked by the vertical bold line. We see that more than 75% of the blocks contain at most 2 homologs per 100 candidate proteins. Thus, in order to do well, the proposed ensemble of phalanxes needs to ensure good ranking of the homologous proteins particularly when the block-wise proportion of homologous protein is small.

There are 74 predictor variables and some of them look very useful as a single variable and some do not. Figure 5.3 (a) shows the kernel density plots of the feature variable $x_{63}$ for the homologous proteins (the solid line) and for the non-homologous proteins (the dashed line) in the training set. This variable seems to differentiate the homologous and non-homologous proteins fairly well. However, there is also evidence of less-informative feature variables in the training set; please see the density plots of the feature variable $x_{47}$ in Figure 5.3 (b). Such




(a) Training (b) Test

Figure 5.1: Histogram of block sizes in the training and test sets of the protein homology datasets.

a less-informative variable is not thrown out instantly, however. We further check whether the variable possesses good marginal prediction power when used with any other predictors in the dataset. If the variable appears useful, it is used in forming the phalanxes.

## 5.3 Evaluation Metrics

We have used three metrics, specific to ranking rare homologous proteins, to evaluate predictive performances of the classifiers. Since the protein homology data comes in blocks, each of the three metrics is computed on each block independently. For each metric, the average performance across the blocks is used as the final metric. Thus, in order to do well a classifier has to rank the rare homologous proteins well across many blocks. By ranking we mean sequencing (or sorting) the candidate proteins in each block using the probability of being homologous to the native protein. The candidate protein with the largest probability of being homologous in a block is ranked first, followed by the second largest probability and so on. The three metrics are as specified by the 2004 KDD Cup competition. For further reading please see the following link: `http://osmot.cs.cornell.edu/kddcup/metrics.html`.

### 5.3.1 Rank Last

By *rank last* (RKL) we mean the rank of the last homologous protein. This metric measures how far the last true homolog falls among the sorted cases. Ties are treated conservatively: if multiple sequences tie, the last element of the tie determines the rank, so ties are not beneficial. An RKL of 1 means that the last true homolog is sorted in the top position.

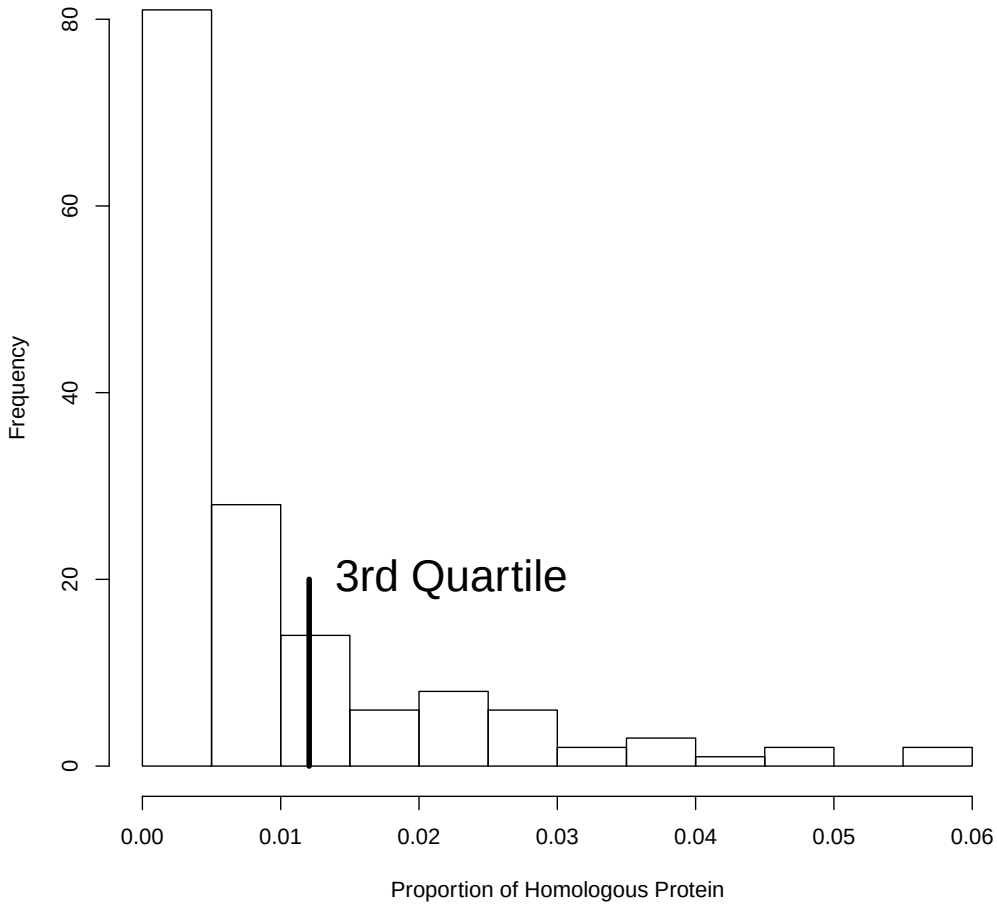
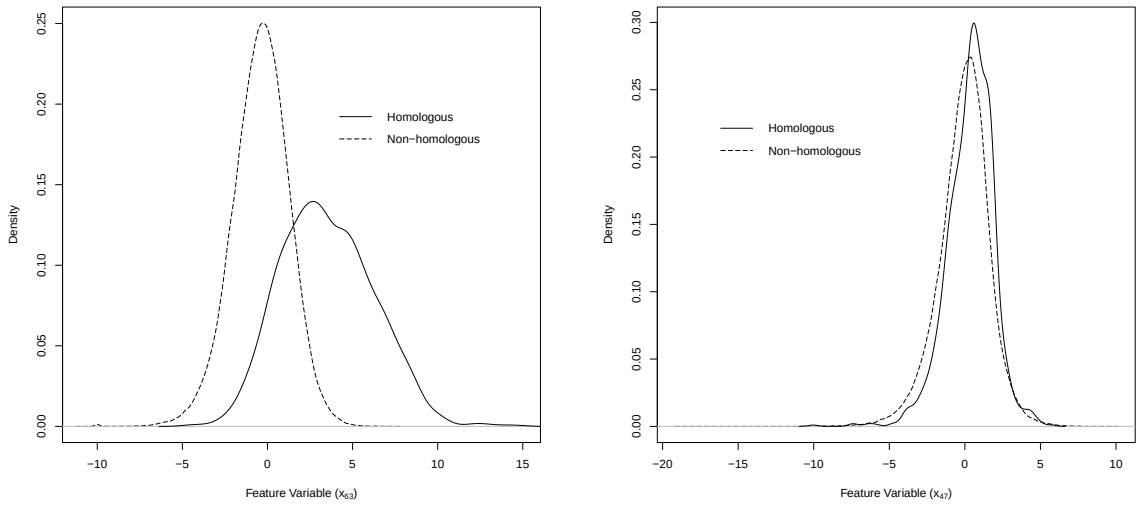



Figure 5.2: Histogram of the proportions of homologous protein in the training set.

This is excellent but can only happen in a block containing only one homologous protein. The maximum possible value of RKL is the size of the block, i.e., the number of candidate proteins. The goal is to keep the RKL as small as possible.

### 5.3.2 Average Precision

The metric *average precision* (APR) is a variant of the metric average hit rate (AHR) we defined in section 2.5.4. They are no different when there are no ties in the score (i.e., probability of being homologous) used for ranking. In case of ties, the former used a non-standard definition, sometimes called "expected precision." This definition uses a method for handling ties that calculates the average precision of all possible orderings of the cases that are tied. An average precision of 1 indicates perfect ranking. The lowest possible average precision depends on the data, and our goal is to maximize this metric.




(a) (b)

Figure 5.3: Kernel density plots of the feature variables $x_{63}$ (panel a) and $x_{47}$ (panel b) for the homologous and non-homologous proteins in the training set.

### 5.3.3 TOP1

The candidate proteins in each block are sorted by the estimated probabilities of being homologous to the native protein. If the top ranked candidate protein is homologous to the native protein, TOP1 scores 1, otherwise 0. TOP1 is calculated conservatively when there are ties. If multiple sequences are tied for rank 1, all of them must be homologous to score a 1 for TOP1. If any of the sequences tied for rank 1 is non-homologous, TOP1 scores 0. This means it is never beneficial to have ties. However, it would be easy (difficult) to maximize TOP1 if a block contains many (a few) homologous proteins. We want to maximize the TOP1.

## 5.4 The Modified Algorithm of Phalanx Formation

Algorithm 5.1 shows the steps of phalanx formation using *logistic regression*. Figure 3.2 in Chapter 3 shows the schematic of the original algorithm (Algorithm 3.1). The differences between the original and the modified algorithms are presented below:

- The modified algorithm incorporates logistic regression as the base classifier. The original algorithm used random forests.
- The modified algorithm uses 10-fold cross validation of the training blocks in phalanx formation to evaluate a predictor (or a set of predictors). Thus, grouping is at the level of blocks, with all the proteins of a block belonging to just one fold. On the other hand, the original algorithm uses out-of-bag (OOB) samples in random forests to evaluate a predictor (or group of predictors) during phalanx formation.

- The original algorithm 3.1 used *average hit rate* (AHR) to optimize the phalanxes. The modified algorithm 5.1 uses *mean block average precision* (MAPR) to optimize the phalanxes.

- In step 4, both the original and adapted algorithms start by searching for the most promising pair of screened predictors to merge together. Unlike the original, the modified algorithm introduces a new step $4(b)i$. The reasons for adding this step are explained in the third paragraph of Section 5.1. In this added step, the new algorithm checks whether the weak performer of the merging pair of predictors degrades the performance of the strong performer. If true, the weak performing predictor (or group of predictors) is filtered out. Otherwise, the pair of predictors (or group of predictors) is merged together.

In this chapter, we have also formed the phalanxes using random forests. For a fair comparison between the resulting ensembles of phalanxes, all of the steps in this adapted algorithm, except for $4(b)i$, are incorporated while forming phalanxes using random forests. The reasons for not incorporating $4(b)i$ are simple: (i) the performance of random forests is insensitive to the inclusion of a few weak variables, and (ii) we want to use as many useful variables as possible in the phalanxes.

## 5.5 Ensemble of Phalanxes

After forming the phalanxes, $p$ ($\geq 1$) classifiers are fitted to obtain $p$ vectors of probabilities of homogeneity. The classifiers are aggregated together by averaging $p$ vectors of probabilities of homogeneity across the phalanxes. If the algorithm ends up with only one phalanx, there is no scope to ensemble over many phalanxes. In this case, fit a classifier to that phalanx of screened predictors. When a logistic regression model is used in phalanx formation, the resulting ensemble of phalanxes is denoted by EPX(LR). For random forests, the resulting ensemble is denoted by EPX(RF).

## 5.6 Results

This section shows the results after applying the ensembles of phalanxes – EPX(RF) and EPX(LR) – and random forests (RF) to the protein homology data. First, we grouped the training blocks into four quarters using the per-block proportions of homologous protein and evaluated the performances of the ensembles in each of the four groups. The goal is to find the situations where the ensembles of phalanxes perform better than random forests. The results are presented next in Subsection 5.6.1. Second, the ensembles are evaluated on the test set. The results are presented in Subsection 5.6.2.

**Algorithm 5.1** Modified Algorithm of Phalanx Formation

1. **Setting the arguments:**
   (a) Predictors and the response: $\{x_1, x_2, \cdots, x_D\}$ and $y$.
   (b) Set *initial groups*, *nsim*=1000, *iquant*=0.95.
   (c) Define a leave 10% blocks out cross-validation.
2. **Forming initial groups of variables:**
   (a) $g_1, g_2, \cdots, g_d \leftarrow x_1, x_2, \cdots, x_D$ (*initial groups*).
3. **Screening initial groups of variables/predictors:**
   (a) Record ordering of the candidate proteins. Permute the response variable $y$ and compute Mean Block APR, i.e., MAPR. Repeat the process *nsim* times.
   (b) Store $\text{MAPR}_{iquant}$, the *iquant*-th quantile of the distribution of MAPR under random ranking.
   (c) For each group of variables $g_i$; $i = 1, 2, \cdots, d$, fit a logistic regression model and get the cross-validated probability vector $\widehat{P}(g_i)$. Store the probability vectors in the columns of a matrix, say PROB.
   (d) Using $\widehat{P}(g_i)$, compute $\text{MAPR}_i = \text{MAPR}\left(\widehat{P}(g_i)\right)$ and store in a vector, say SAPR.
   (e) For each pair of initial groups $\{g_i, g_j\}; i = 1, \cdots, (d-1), j = (i+1), \cdots, d$, get $\widehat{P}(g_i, g_j)$, $\left\{\widehat{P}(g_i) + \widehat{P}(g_j)\right\}/2$ and thus $\text{MAPR}_{ij} = \text{MAPR}\left(\widehat{P}(g_i, g_j)\right)$, $\text{MAPR}_{\overline{ij}} = \text{MAPR}\left(\left\{\widehat{P}(g_i) + \widehat{P}(g_j)\right\}/2\right)$, respectively. Store $\text{MAPR}_{ij}$ and $\text{MAPR}_{\overline{ij}}$ in matrices APRSYN and APRCOM, respectively.
   (f) Screen the $i$th group of variables out if $\max[\text{MAPR}_i, E(\text{MAPR}_R) + \text{MAPR}_{ij} - \text{MAPR}_j, E(\text{MAPR}_R) + \text{MAPR}_{\overline{ij}} - \text{MAPR}_j] < \text{MAPR}_{iquant}\ \forall\ j \neq i = 1, \cdots, d$.
   (g) Update 3($c$), 3($d$), and 3($e$) by deleting rows and columns of PROB, SAPR, APRSYN, and APRCOM corresponding to the screened out groups of variables.
   (h) Supply the screened groups of predictors: $G_1, G_2, \cdots, G_s$.
4. **Forming candidate phalanxes:**
   (a) Find $r_{\min} = \min_{ij}\{\text{MAPR}_{\overline{ij}}/\text{MAPR}_{ij}\}; i = 1, 2, \cdots, (s-1), j = (i+1), \cdots, s$.
   (b) If $r_{\min} < 1$, then:
      i. If $\text{MAPR}_{ij} \leq \max(\text{MAPR}_i, \text{MAPR}_j)$, screen out the $i$th group of variables $\{G_i\}$ if $\text{MAPR}_i \leq \text{MAPR}_j$; else screen out the $j$th group. Update 3($c$), 3($d$), and 3($e$) by deleting rows and columns of PROB, SAPR, APRSYN, and APRCOM corresponding to the screened out groups of variables. Set $s \leftarrow (s-1)$ and go to 4($a$).
   (c) Block the groups of variables $\{G_i, G_j\}|r_{\min}$ together and $s \leftarrow (s-1)$.
   (d) Redo the steps 3($c$), 3($d$), and 3($e$) specific to the index of $r_{\min}$.
   (e) Repeat 4($a$) to 4($d$) until $r_{\min} \geq 1$ OR $s = 1$.
   (f) Set $c \leftarrow s$ and supply candidate phalanxes: $\text{PX}_1, \text{PX}_2, \cdots, \text{PX}_c$.
5. **Screening candidate phalanxes:**
   (a) Screen the $i$th candidate phalanx out if $\max[\text{MAPR}_i, E(\text{MAPR}_R) + \text{MAPR}_{\overline{ij}} - \text{MAPR}_j] < \text{MAPR}_{iquant}\ \forall\ j \neq i = 1, \cdots, c$.
   (b) Return $\text{PX}_{(1)}, \text{PX}_{(2)}, \cdots, \text{PX}_{(p)}$: the army of phalanxes.

### 5.6.1 Grouping the Training Blocks

Groups are formed stratified by proportions of homologous proteins. The 153 training blocks are partitioned into four groups using the per-block proportions of homologous protein. Specifically, the four groups are the four quarters of the 153 training blocks. For example, the first group contains 39 native protein sequences (i.e., blocks) having per-block proportions of homologous proteins less than or equal to the first quartile (0.00143). The second group contains 38 native protein sequences with per-block proportions of homologous proteins greater than the first quartile and less than or equal to the second quartile (0.0047). The third and fourth groups are defined in the same way each containing 38 native protein sequences.

Panel ($a$) of Figure 5.4 shows four histograms of the proportions of homologs in the four groups. Group 1 contains the blocks with the smallest proportions of homologous proteins - there is only 1 homologous protein in each block. In contrast, Group 4 contains the blocks with the largest proportions of homologous proteins, i.e., the number of per-block homologous proteins ranges from 11 to 50. I conjecture that the ranking of the homologous proteins is very challenging in group 1 followed by groups 2, 3 and 4. As the blocks in group 4 contain plenty of homologous proteins, it might be easy to score a good number for some of the evaluation metrics. For example, if there are 50 homologous proteins in a block, it might not be very hard to rank a true homolog in the top position yielding the highest score 1 for TOP1.

Let us now explain the evaluation process of EPX(LR) in a particular group. The evaluation processes for the other groups are analogous. Given the blocks in a particular group, a set of screened phalanxes is obtained by running Algorithm 5.1. As explained in Section 5.4, we used a leave 10% blocks out cross-validation to guide the formation of the phalanxes. The screened phalanxes are ensembled to get EPX(LR). To evaluate the resulting EPX(LR), a completely independent leave 10% blocks out cross-validation is performed to get a vector of the probabilities of being homologous to the native protein for all of the candidate proteins in each block of that group. This vector of probabilities is used to compute the three evaluation metrics. The leave 10% blocks out cross-validation is repeated for a total of 16 times providing 16 numbers for each evaluation metric.

The panels ($b$), ($c$) and ($d$) in Figure 5.4 show the side-by-side box plots comparing the performances of the three ensembles - RF, EPX(RF) and EPX(LR) - in each of the four groups in terms of mean block RKL, mean block APR and mean block TOP1, respectively. Each box represents 16 realizations of a metric for an ensemble in a group. The colors of the boxes differentiating RF, EPX(RF) and EPX(LR) are *white*, *light grey* and *dark grey*, respectively. The numbers at the top of the boxplots are the numbers of phalanxes obtained after running the *modified algorithm of phalanx formation* (Algorithm 5.1).

In terms of mean block RKL (panel ($b$)), our ensembles EPX(RF) and EPX(LR) gave better predictive ranking than RF in all of the four groups. The adapted ensemble EPX(LR) outperformed EPX(RF) in groups 1, 2 and 3, but not in group 4. The ensembles of phalanxes, EPX(RF) and EPX(LR), outperformed RF in all of the four groups in terms of mean block

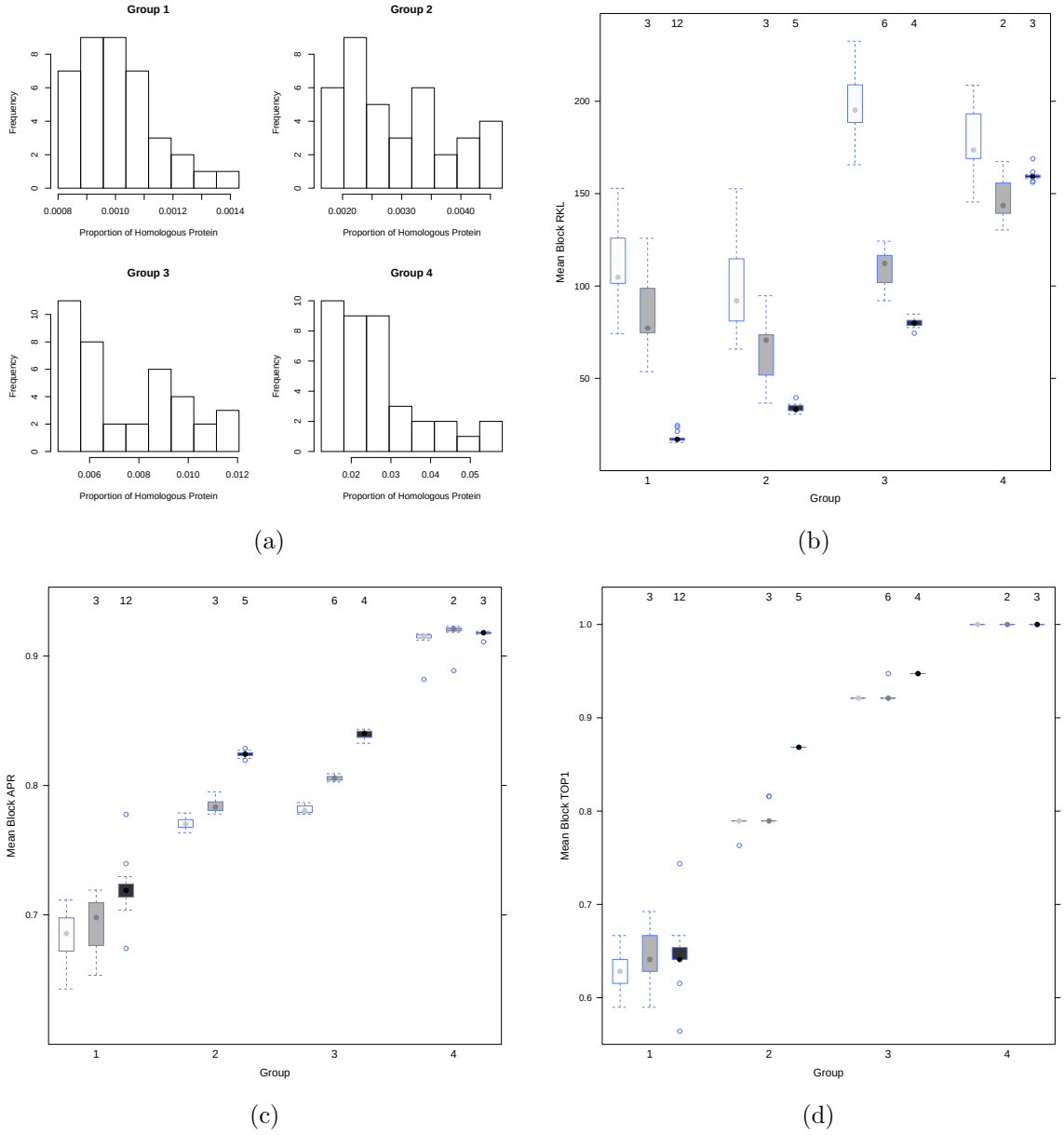



Figure 5.4: Histograms of the proportion of homologous proteins in four groups (panel *a*). Side-by-side box plots comparing the performances of the three ensembles - RF, EPX(RF) and EPX(LR) - in each of the four groups in terms of mean block RKL (panel *b*), mean block APR (panel *c*) and mean block TOP1 (panel *d*). The colors of the boxes differentiating RF, EPX(RF) and EPX(LR) are white, light grey and dark grey respectively.

APR (panel ($c$)). The most striking improvement of EPX(LR) over EPX(RF) is observed in groups 2 and 3, followed by group 1. In terms of mean block TOP1 (panel ($d$)), EPX(RF) often outperformed RF. In group 4, all of the ensembles secured the top mean block TOP1. But in groups 2 and 3, EPX(LR) substantially outperformed EPX(RF).

Clearly the ensembles of phalanxes are the winners against random forests in groups 2 and 3. Hence, we want to closely look at the performances of the three ensembles in groups 1 and 4, where our ensembles marginally win against random forests. The left and right panels of Figure 5.5 show hit curves in two typical blocks, 18 and 162, selected from groups 1 and 4, respectively. In both panels, the dotted, dashed and the solid lines are for random forests, ensemble of phalanxes using random forests and ensemble of phalanxes using logistic regression model, respectively. The legend also shows the estimated APR, RKL and TOP1 for the three ensembles. In each panel, the performances are reported for only one block so the metrics are not averaged.

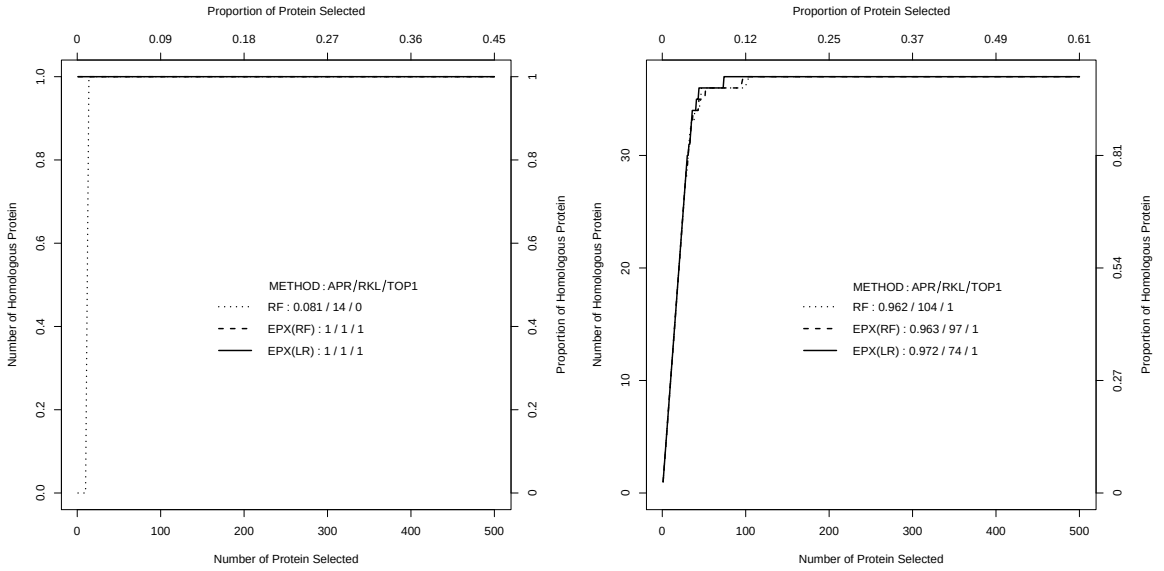
Panel ($a$) of Figure 5.5 is for block 18 which is chosen from group 1. Ranking of the homologous protein is challenging as this block contains only 1 homolog out of 1114 candidate proteins. Using the three evaluation metrics, the top performers are our ensembles EPX(LR) and EPX(RF). They ranked the only homologous protein in the top position, and all of the three metrics achieved the top score. Random forests could not rank the homologous protein in the top position and scored 0 for TOP1. For random forests, the estimated APR and RKL were 0.081 and 14, respectively.

Block 162 is chosen from group 4 and contains a total of 37 homologous proteins out of 816 candidate proteins (panel $b$ of Figure 5.5). Ranking of the homologous proteins is least challenging in this block compared to the blocks in other three groups. All of the three ensembles perfectly rank the first 28 homologous proteins scoring 1 for TOP1. The top performer is EPX(LR) followed by EPX(RF) and RF in terms of RKL and APR.

When there are many homologous proteins in a block, all of the three ensembles might rank a homologous protein in the top position to provide the best possible score for TOP1. Hence, the comparison of the ensembles using TOP1 might appear problematic especially when there are many homologous proteins. However using TOP1, our ensembles EPX(LR) and EPX(RF) often outperform RF if the blocks contain a few homologous proteins. Our favorite performance metrics are RKL and APR, and using them, the ensembles EPX(LR) and EPX(RF) are clear winner over random forests. Moreover, for the most part, the adapted ensemble EPX(LR) outperforms EPX(RF).

### 5.6.2 Evaluation on the Test Set

In this section, the three ensembles are evaluated on the test set. The organizers of the 2004 KDD cup decided not to disclose the status of the response variable for the test set - however, the results can be obtained by submitting intermediate results to their website. Hence, the ensembles of phalanxes and random forests were learned on the 153 training




(a) (b)

Figure 5.5: Hit curves comparing the performances of three ensembles - RF, EPX(RF) and EPX(LR) - in blocks 18 (panel *a*) and 162 (panel *b*) chosen from the groups 1 and 4, respectively. The respective block sizes are 1114 and 816. The legend in each figure also shows the estimated APR, RKL and TOP1 for the three ensembles.

blocks, for which the response was known, and were applied to the test set to obtain the probabilities of being homologous to the native protein. The estimated probabilities were preserved in a specific format suggested by the KDD cup organizers and were submitted to their website for the result. The KDD cup website did check the format of the submitted probabilities automatically and reported back the final results. It is important to mention that we submitted one set of probabilities corresponding to an ensemble to compute all of the evaluation metrics, i.e. the methods were not tuned specifically for each criterion.

Table 5.3 shows the performances of the three ensembles - RF, EPX(RF) and EPX(LR) - using the three evaluation metrics: mean block RKL, APR and TOP1. We obtain 4 and 5 phalanxes using EPX(RF) and EPX(LR), respectively. In terms of mean block RKL, APR and TOP1, the top performer is EPX(LR) followed by EPX(RF) and RF. In terms of mean block TOP1, the performances of RF and EPX(RF) are found similar.

Table 5.4 shows the number of processors used, amount of memory allocated and the amount of time recorded (in minute) for parallel execution of the three ensembles. For EPX(RF) and EPX(LR), a total of 32 processors of the Bugaboo machine were *parallelized* in the Western Canada Research Grid (WestGrid) computing network (`http://www.westgrid.ca/support/quickstart/bugaboo`). We allocated 8GB memory to each processor while running EPX(RF) and 4GB memory to each while running EPX(LR). When EPX(RF) took a total of 26 hours and 9 minutes for learning on the training set and estimating the probabilities of being homologous to the native protein on the test set, the EPX(LR) took only 1 hour

Table 5.3: The performances of the three ensembles - RF, EPX(RF) and EPX(LR) - using three evaluation metrics: mean block RKL, APR and TOP1. The ensembles are learned on the 153 training blocks and are evaluated on the 150 test blocks. The top performance corresponding to each evaluation metric is marked by fold face.

| Ensembles | Number of Phalanxes | Mean Block | | |
|---|---|---|---|---|
| | | RKL | APR | TOP1 |
| RF | $\cdots$ | 143.733 | 0.8089 | 0.8733 |
| EPX(RF) | 4 | 82.3067 | 0.8140 | 0.8733 |
| EPX(LR) | 5 | **71.4733** | **0.8274** | **0.8867** |

and 20 minutes to finish the same task. Using random forests, the task was completed assigning only 1 processor with 8GB memory. The total running time was 1 hour and 7 minutes. However, the goal was to reduce the computational time for EPX(RF) through EPX(LR). Here, EPX(LR) runs much *faster* than EPX(RF) and gives *better results* as well.

Table 5.4: The number of processors used, amount of memory allocated and the elapsed time recorded (in minute) for parallel execution of the three ensembles.

| Ensembles | Number of Processors | Memory allocation (GB) | Elapsed time (Minute) |
|---|---|---|---|
| RF | 1 | 8 | 67 |
| EPX(RF) | 32 | 8 | 1569 |
| EPX(LR) | 32 | 4 | 80 |

Now let us compare the results of our ensemble with the winner of the 2004 KDD Cup. For the protein homology section, the winner of the 2004 KDD Cup was the Weka Group (Pfahringer, 2004). Weka is a collection of machine learning algorithms for data mining tasks (`http://www.cs.waikato.ac.nz/ml/weka/`). Weka supports a large number of algorithms out of the box, and the winning group tried all of them. The Weka group then selected only the top three classifiers, which were: (1) a boosting ensemble of 10 unpruned decision trees (Boost10), (2) a linear support vector machines with a logistic model fitted to the raw outputs for improved probabilities (LinSVM), and (3) 100000 or more random rules ($10^5$ RR). The top three performers were aggregated to obtain the winning ensemble. Table 5.5 shows the performance of the top three constituent classifiers as well as the performance of the winning ensemble. The performance of the ensemble of phalanxes using logistic regression, EPX(LR), is more or less like the LinSVM of Weka group which was one of the top performers selected from many classifiers. However, our target was not to change much of the original ensemble of phalanxes – but to develop a general purpose ensemble with good results. Despite changing a little of the algorithm of phalanx formation, the adapted ensemble EPX(LR) does provide impressive results.

Table 5.5: The results from the Weka solution to the 2004 KDD Cup.

| Classifiers | Mean Block | | |
|---|---|---|---|
| | RKL | APR | TOP1 |
| Boost10 | 500.68 | 0.6858 | 0.7933 |
| LinSVM | 64.41 | 0.8258 | 0.8867 |
| $10^5$ RR | 53.77 | 0.8373 | 0.8933 |
| Ensemble | 52.45 | 0.8412 | 0.9067 |

## 5.7 Summary

The ensemble of phalanxes is easily adapted to the logistic regression model replacing random forests. This proves that the scope of the ensemble of phalanxes is not limited to random forests only; some other simple models could also be used. One potential candidate is obviously the *classification tree* which is a simple model in nature and fast in terms of computational complexity. A potential question in using a classification tree would be to decide how deep to grow the trees. This problem might be addressed with an extra computational effort of employing cross-validation, however.

The ensemble of phalanxes is adapted to the logistic regression model by introducing filtering in step $4(b)i$ during phalanx formation. If the weak performing variable (group of variables) of the most promising pair of variables (groups of variables) to merge harms the top performing variable (group of variables), the modified algorithm filters the weak performer. This change is crucial for the logistic regression model – but not for random forests. In random forests, merging two groups of variables usually does not harm the top performing group unless the weak performing group contains too many noise variables. Moreover, we did not encounter any initial screening in this application as all of the 74 predictors, either alone or together with other variable, beat the threshold of random ranking.

In order to ensure the good performance of EPX(LR), the adaptation through the intermediate step of filtering was necessary. However, the goal was not to change much of the algorithm as we wanted to develop a general purpose ensemble. We have observed that the adapted ensemble EPX(LR) outperforms EPX(RF) with a big margin using three evaluation metrics specific to ranking rare homologous proteins. However, some thoughtful modifications could further improve the performance of EPX(LR). For example, the use of appropriate weights to aggregate the phalanxes might help improving the performances in terms of predictive ranking.

It is shown that the EPX(LR) is much faster than EPX(RF) in terms of computational time. Although parallel computation is used to learn EPX(LR), we expect that the ensemble could also be trained using a single processor in a reasonable amount of time by coding the algorithm in C/C++ or in Fortran. Having finished the coding, we plan to embed the program in an **R** package in the near future.

# Chapter 6

# Conclusion and Future Work

This thesis tackles a very special type of two-class classification problem. The specialty is that the frequency of one of the classes is very small (i.e., rare) comparing to the other class. Moreover, it is the rare, not the majority, class for which the correct classification is required. For example, in medicinal chemistry, a chemist might be interested in classifying the rare active compounds only in a chemical library of many compounds.

Minimizing the misclassification error rate can be unhelpful to such problems. To adapt classification to unbalanced classes, statisticians or machine learners often maximize the following evaluation metrics: precision, recall, G-mean (geometric mean of precision and recall) and F-measure (harmonic mean of precision and recall). An alternative is to sequence all of the observations in a dataset in such a way that the rare class observations are found at the top of a shortlist. The shortlist comprises the top-ranked observations which are intended to include most, if not all, of the rare-class observations. The length of the shortlist would depend on the available resources in a project. As such, after shortlisting the compounds in a large chemical library, the chemist only needs to go through the shortlist to find most of the active compounds in the library and thus to save time and money.

I chose four bioassay datasets with chemical compounds that are either active or inactive against a biological target, e.g., active against lung tumor cells. The predictors are the numerically represented chemical structures of the compounds. It is possible to use many different types of classifiers to rank the compounds. However, classification-tree based ensembles are used in this thesis for two reasons: (i) ensembles usually possess better predictive power than non-ensembles, and (ii) classification-tree based methods possess good predictive power with inherent variable selection properties, and are applicable to mixed type of high-dimensional datasets. Specifically, the popular state-of-the-art ensemble random forests is used because of its superiority over other ensemble and non-ensemble classifiers.

In each of the four bioassays, I use five sets of predictor variables. In Chapter 2, the ensemble of descriptor sets (EDS) was proposed by aggregating five random forests fitted to the five descriptor sets. The resulting EDS outperforms the top performing random forests applied to any of the single set of descriptors. It is the natural diversity and strength of the random forests built across the descriptor sets which help to develop a powerful ensemble like EDS. I name such descriptor sets as *natural phalanxes*, where the variables within a phalanx are good when used together and the variables between phalanxes are good to help each other in an ensemble.

The presence of such natural phalanxes in the data motivates me to unmask *data-adaptive phalanxes* in a variable-rich descriptor set. For this purpose, an algorithm of phalanx formation (Algorithm 3.1) is proposed in Chapter 3. The ensemble random forests itself is used to unmask *data-adaptive phalanxes* in each of the five descriptor sets. The algorithm filters the weak variables and forms the phalanxes using unfiltered variables. As expected, the large descriptor sets tend to supply many phalanxes.

The ensemble of phalanxes (EPX) is obtained by fitting a random forest in each phalanx and aggregating probabilities across the phalanxes. The performance of EPX was compared with random forests and regularized random forests. The former performs better in terms of predictive ranking than the latter two ensembles. The advantage of EPX over the two ensembles is large when there are many variables in a descriptor set and when an assay contains a small proportion of active compounds. The ensemble of descriptor sets is also improved replacing random forests in each descriptor set by the ensemble of phalanxes. The replacement of random forests by a more powerful ensemble like EPX further improves the performances of the resulting ensemble of descriptor sets.

In Chapter 3, the phalanxes are formed using the probabilities of belonging to the rare class obtained from the out-of-bag (OOB) samples in a random forests, whereas in Chapter 5, the phalanxes are formed using the probabilities obtained from a cross-validation. The datasets used in Chapters 3 and 5 are the bioassay datasets and the protein homology dataset, respectively. The ensemble of phalanxes performs better than the random forests in those applications. The ensemble of phalanxes is also adapted to the logistic regression model in Chapter 5. The resulting ensemble of phalanxes runs faster in terms of computational time and performs better in terms of predictive ranking than the original ensemble of phalanxes using random forests. This shows that the ensemble of phalanxes is generalizable to any suitable classification methods.

Last, but not the least, the protein homology dataset provides a *real test set* without disclosing the status of the response variable for the candidate proteins. However, it is possible to evaluate the performances of the ensembles by submitting the probabilities of being homologous to the KDD cup website. The test results prove the superiority of the ensemble of phalanxes over random forests. The evaluation of the ensemble of phalanxes in the test set is completely honest and confirms the cross-validated results found in Chapters 3 and 4.

## 6.1 Parallel Computation

In this thesis, instead of executing the computations sequentially, I choose to execute them in parallel using many processors/cores in a computer or using many nodes in a cluster. The primary goal is to reduce the computational time.

Specifically, I used the four `MAC processors` (Grouse, Baker, Seymour and Cypress) in the computing servers of the department of statistics at the University of British Columbia. Those

processors provide high computing power with large distributed memory in each and are good for high performance computing within each. But the interconnection of the processors does not support large data transfer from one to another. In order to facilitate the transfer of large data and results from one node to another as well as to get high performance computing, I moved to the `western Canada research grid` (WestGrid) cluster *Bugaboo*.

To provide parallel backend for the **R** commands which support parallel execution, I used the **R** packages **doSNOW** and **doMPI**. If the processors are not connected as a cluster due to various reasons, one can build a Simple Network Of Workstations (SNOW) using the **R** package **doSNOW**. The processors in the computing servers of the department of statistics at UBC fall in this category, and I used **doSNOW** to create parallel backend. On the other hand, the computing nodes in Bugaboo are connected as clusters, and I used **doMPI** for creating parallel backend.

Both of the **R** packages **doSNOW** and **doMPI** create parallel backend for the **R** package *foreach*. The **R** package *foreach* gives a parallel programming framework and provides a looping construct for executing **R** codes repeatedly on multiple processors/cores on a computer or on multiple nodes of a cluster. The package *foreach* also supports sequential execution of repeating statements with a minor change in the command line. This attractive feature of *foreach* enables a programmer writing and debugging codes in a personal computer and then executing the final codes in parallel in a cluster.

As stated earlier, I moved from the statistics department's servers to WestGrid's cluster Bubaboo in order to facilitate transfer of large data files and results from one node to another. Yet I employed another **R** package *iterators* to further decrease the overhead data transfer. As the data communication overhead was heavy, I considered chunking and transferring the appropriate portion of the data needed for computing using *iterators*. Even though there are several built-in statements in *iterators*, I created my own *iterators* to transfer appropriate chunk of data suitable to my **R** program.

## 6.2 Future Work

Throughout this thesis, the phalanxes are aggregated to form an ensemble using unweighted average of the probabilities belonging to the rare class. This is common both in case of forming the phalanxes and aggregating the final phalanxes. I propose that the strong phalanxes get larger weights than the weak phalanxes to improve prediction performance of the resulting ensemble. Thus, determining appropriate weights for the phalanxes would be a good extension of the current ensemble of phalanxes. To determine the optimal weight, I plan to optimize the performance of the ensemble by examining individual strengths of the phalanxes as well as the covariance structure of the probabilities belonging to the rare class between the phalanxes. This process of determining optimal weight would be adapted not only to aggregate final phalanxes but also to form the phalanxes. In addition to optimal weights, I plan to determine

an appropriate bound on prediction performance of the ensemble by studying the strengths and correlations between constituent classifiers constructed using the phalanxes.

One of the objectives of this thesis is to sort the rare class observations through ranking. To evaluate the performance of a classifier in terms of predictive ranking, I mainly used average hit rate (i.e., average precision) as a single number summary of a hit curve. The metric "average hit rate" is also used to form the phalanxes. The formation of phalanxes is also possible by optimizing other evaluation metrics, for example, optimizing rank last (RKL). Such generalization of the ensemble of phalanxes would be a good extension of this current method.

A straightforward extension of this ensemble would be in *regression*, where the response variable is continuous. For example, the four assay datasets used in Chapters 2 to 4 typically resulted in continuous responses (percent inhibition) from a primary screen and binary responses (active versus inactive) from a secondary screen. Had I used the continuous response variables from those assays, I would have dealt with models related to classical regression. To deal with continuous response variable as in classical regression, I would have to optimize the metric *squared error loss* to form the phalanxes as well as to evaluate the resulting ensemble. Having predicted the response *percent inhibition*, one may rank to sort out the compounds with high activity.

Unlike the rare class problem, the frequencies of the classes could be approximately equal to each other in many applications. An extension of this ensemble of phalanxes is possible in such balanced classification problem by employing the metric *misclassification error*. Hopefully, the ensemble of phalanxes would outperform random forests in balanced classification problems too, if a dataset contains many useful feature variables with signals for the response.

The *socialization* of the predictor variables in the screened phalanxes can be made by a simple modification of the algorithm 1.1 of random forests presented in Section 1.3.2. The proposed modification is as follows: at each node of a tree in a forest, randomly choose $m_{\text{try}} = \sqrt{D_i}$ variables to find the split-point of that node. Here, $D_i$ ; $i = 1, \cdots p$, is the number of feature variables in the $i$th phalanx. As a result of this modification, the resulting trees in the modified random forests would be more diverse and stronger than the trees generated by regular random forests. For the above mentioned reasons, the modification to random forests might perform better in terms of predictive ranking than regular random forests.

The algorithm for the ensemble of phalanxes is coded in **R**, the freely available statistical software. The original C codes for the three evaluation metrics – APR, RKL, and TOP1 - are downloaded from the 2004 KDD Cup website and are used in my programs. Besides, the whole **R** program is designed in such a way that the program can be executed using as many nodes as possible in a cluster of processors. With the vision of running the codes in a single processor, I plan to rewrite the entire program in C/C++ or in Fortran and wrap it up in an **R** package. In this **R** package, there would be opportunity for employing most of the popular evaluation metrics to form the phalanxes and to build the ensemble as well.

# Bibliography


Aoyama, T., Suzuki, Y., and Ichikawa, H. (1990), "Neural networks applied to quantitative structure-activity relationships," *Journal of Medicinal Chemistry*, 33, 2583–2590. → pages 15

Bolton, R. and Hand, D. (2002), "Statistical fraud detection: a review," *Statistical Science*, 17, 235–255. → pages 2

Boser, B., Guyon, I., and Vapnik, V. (1992), "A training algorithm for optimal margin classifiers," in *5th Annual ACM Workshop on COLT*, ed. Haussler, D., Pittsburgh, PA: ACM Press, pp. 144–152. → pages 7

Breiman, L. (1996a), "Bagging predictors," *Machine Learning*, 26, 123–140. → pages 2, 8

— (1996b), "Heuristics of instability and stabilization in model selection," *The Annals of Statistics*, 24, 2350–2383. → pages 9

— (1996c), "Out-of-bag estimation," Tech. rep., Department of Statistics, University of California, Berkley, USA. → pages 38

— (2001), "Random forests," *Machine Learning*, 45, 5–32. → pages 2, 9, 10, 20, 28, 33, 38, 47, 51, 60

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*, The Wadsworth Statistics/Probability Series, Belmont, California, USA: Wadsworth International Group. → pages 4

Bruce, C. L., Melville, J. L., Pickett, S. D., and Hirst, J. D. (2007), "Contemporary QSAR classifiers compared," *Journal of Chemical Information and Modeling*, 47, 219–227. → pages 16, 20

Burden, F. R. (1989), "Molecular identification number for substructure searches," *Journal of Chemical Information and Computational Sciences*, 29, 225–227. → pages 19, 52

Carhart, R. E., Smith, D. H., and Ventkataraghavan, R. (1985), "Atom pairs as molecular features in structure-activity studies: definition and application," *Journal of Chemical Information and Computational Sciences*, 25, 64–73. → pages 19, 52

Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. (2003), "SMOTEBoost: Improving prediction of the minority class in boosting," in *7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, Croatia, pp. 107–119. → pages 50

Chen, C., Liaw, A., and Breiman, L. (2004), "Using random forest to learn imbalanced data," Tech. rep., Department of Statistics, University of California, Berkeley, CA. → pages 16, 50

Cortes, C. and Vapnik, V. (1995), "Support-vector networks," *Machine Learning*, 20, 273–297. → pages 7

Cover, T. and Hart, P. (1967), "Nearest neighbour pattern classification," *IEEE Transaction on Information Theory*, 13, 21–27. → pages 5

DeBarr, D. and Wechsler, H. (2012), "Spam detection using random boost," *Pattern Recognition Letters*, 33, 1237–1244. → pages 1

Deng, H. (2012), *R Package Regularized Random Forest*, Version 1.2; R Foundation for Statistical Computing: Viena, Austria. → pages 10, 42

Deng, H. and Runger, G. (2012), "Feature selection via regularized trees," in *The 2012 International Joint Conference on Neural Networks (IJCNN), IEEE.* → pages 10, 33, 51

Dietterich, T. G. (2000), "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, 1857, 1–15. → pages 20

Doniger, S. and Hofmann, T. (2002), "Predicting CNS permeability of drug molecules: comparison of neural network and support vector machine algorithms," *Journal of Computational Biology*, 9, 849–864. → pages 15

Efron, B. and Tibshirani, R. (1994), *An Introduction to the Bootstrap*, Chapman & Hall/CRC. → pages 8

Fienberg, S. E. (2004), "Homeland insecurity: Data mining, terrorism detection, and confidentiality," Tech. Rep. 148, National Institute of Statistical Sciences, 19 T. W. Alexander Drive, Research Triangle Park, NC. → pages 1

Fisher, R. (1936), "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, 7, 179–188. → pages 6

Freund, Y. (1995), "Boosting a weak learning algorithm by majority," *Information and Computation*, 121, 256–285. → pages 10

Freund, Y. and Schapire, R. (1996a), "Experiments with a new boosting algorithm," in *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kaufmann, San Francisco, pp. 148–156. → pages 2, 10, 11

— (1996b), "Game theory, online prediction and boosting," in *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pp. 325–332. → pages 10

— (1997), "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, 55, 119–139. → pages 10

Friedman, J. (1989), "Regularized discriminant analysis," *Journal of the American Statistical Association*, 84, 165–175. → pages 6

Hall, P., Park, B., and Samworth, R. (2008), "Choice of neighbor order in nearest-neighbor classification," *The Annals of Statistics*, 36, 2135–2152. → pages 4

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2nd ed. → pages 6, 7, 8, 12

Hughes-Oliver, J. M., Brooks, A. D., Welch, W. J., Khaledi, M. G., Hawkins, D., Young, S. S., Patil, K., Howell, G. W., and Ng, R. T. (2011), "ChemModLab: A web-based cheminformatics modeling laboratory," *In Silico Biology*, 11, 61–81. → pages 2, 15, 16, 19, 20, 22, 29, 34, 54

Johnson, R. A. and Wichern, D. W. (2002), *Applied Multivariate Statistical Analysis*, Prentice Hall, 5th ed. → pages 25

Kauffman, G. W. and Jurs, P. C. (2001), "QSAR and k-nearest neighbor classification analysis of selective cyclooxygenase-2 inhibitors using topologically based numerical descriptors," *Journal of Chemical Information and Computational Sciences*, 41, 1553–1560. → pages 15

Kearsley, S. K., Sallamack, S., Fluder, E. M., Andose, J. D., Mosley, R. T., and Sheridan, R. P. (1996), "Chemical similarity using physiochemical property descriptors," *Journal of Chemical Information and Computational Sciences*, 36, 118–127. → pages 22, 34

Koonin, E. V. and Galperin, M. Y. (2003), *Sequence Evolution Function: Computational Approaches in Comparative Genomics*, Boston: Kluwer Academic. → pages 14

Leach, A. R. and Gillet, V. J. (2007), *An Introduction to Chemoinformatics*, Dordrecht, Netherlands: Springer. → pages 13, 16

Liaw, A. and Wiener, M. (2011), *R Package Random Forest*, Version 4.6-2; R Foundation for Statistical Computing: Viena, Austria. → pages 20, 41, 42

Liu, K., Feng, J., and Young, S. S. (2005), "PowerMV: A software environment for molecular viewing, descriptor generation, data analysis and hit evaluation," *Journal of Chemical Information and Modeling*, 45, 515–522. → pages 19

McCulloch, W. and Pitts, W. (1943), "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, 5, 115–133. → pages 6

Meier, L., van de Geer, S., and Buhlmann, P. (2008), "The group lasso for logistic regression," *Journal of the Royal Statistical Society, Series B*, 70, 53–71. → pages 32, 59

Merkwirth, C., Mauser, H., Schulz-Gasch, T., Roche, O., Stahl, M., and Lengauer, T. (2004), "Ensemble methods for classification in chemiinformatics," *Journal of Chemical Information and Computational Sciences*, 44, 1971–1978. → pages 15

Mezey, P., Carbo, R., and Girones, X. (2001), *Fundamentals of molecular similarity*, New York: Kluwer Academic/Plenum Publishers. → pages 13

Pearlman, R. S. and Smith, K. M. (1999), "Metric validation and the receptor-relevant subspace concept," *Journal of Chemical Information and Computational Sciences*, 39, 28–35. → pages 19, 52

Pfahringer, B. (2004), "The weka solution to the 2004 KDD cup," *SIGKDD Explorations*, 6, 117–119. → pages 73

Ripley, B. D. (2011), *R Package Tree*, Version 1.0-29; R Foundation for Statistical Computing: Viena, Austria. → pages 28

Rosenblatt, F. (1962), *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, Washington, D.C. → pages 6

Rusinko, A., Farman, M. W., Lambert, C. G., Brown, P. L., and Young, S. S. (1999), "Analysis of a large structure/biological activity data set using recursive partitioning," *Journal of Chemical Information and Computational Sciences*, 39, 1017–1026. → pages 15

Schapire, R. (1990), "The strength of weak learnability," *Machine Learning*, 5, 197–227. → pages 10

Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008), "Credit card fraud detection using hidden markov model," *IEEE Transactions on Dependable and Secure Computing*, 5, 37–48. → pages 1

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., and Feuston, B. R. (2003), "Random forest: A classification and regression tool for compound classification and QSAR modeling," *Journal of Chemical Information and Computational Sciences*, 43, 1947–1958. → pages 15, 16, 20

Tebbutt, S., Opushnyev, I., Tripp, B., Kassamali, A., Alexander, W., and Andersen, M. (2005), "SNP chart: An integrated platform for visualization and interpretation of microarray genotyping data," *Bioinformatics*, 21, 124–124. → pages 32

Tibshirani, R. (1996), "Bias, variance, and prediction error for classification rules," Tech. rep., Department of Statistics, University of Toronto, ON, Canada. → pages 38

Todeschini, R. and Consonni, V. (2000), *Handbook of Molecular Descriptors*, Weinheim, Germany: WILEY-VCH. → pages 13, 16

Venables, W. and Ripley, B. (2002), *Modern Applied Statistics with S*, Springer, 4th ed. → pages 4

Wang, Y. (2005), "Statistical Methods for High Throughput Screening Drug Discovery Data," Ph.D. thesis, University of Waterloo, Waterloo, Canada. → pages 2, 11, 12, 13, 20, 23

Warmuth, W., Ratsch, G., Mathieson, M., Liao, J., and Lemmen, C. (2001), "Active learning in the drug discovery process," *NIPS*. → pages 1

Widrow, B. and Hoff, M. (1960), "Adaptive switching circuits," *IRE WESCON Convention record*, 4, 96–104. → pages 6

Wolpert, D. and Macready, W. (1999), "An efficient method to estimate baggings generalization error," *Machine Learning*, 35, 41–55. → pages 38

Yuan, M. and Lin, Y. (2007), "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society, Series B*, 68, 49–67. → pages 32, 59

Zhang, Q., Hughes-Oliver, J. M., and Ng, R. T. (2009), "A model-based ensembling approach for developing QSARs," *Journal of Chemical Information and Modeling*, 49, 1857–1865. → pages 16, 17, 20

Zhu, M. (2004), "Recall, precision and average precision," Tech. rep., Department of Statistics and Actuarial Science, University of Waterloo, ON, Canada. → pages 23, 34

Zhu, M., Su, W., and Chipman, H. (2006), "LAGO: A computationally efficient approach for statistical detection," *Technometrics*, 48, 193–205. → pages 8, 34

# Appendix A

# Supplementary Tables

Table A.1: The number of variables, initial groups, screened groups, candidate phalanxes, and screened/army of phalanxes for the 3 runs of the algorithm to AID362 assay.

| DS | Run | Number of | | | | |
|---|---|---|---|---|---|---|
| | | Variables | Groups | | Phalanxes | |
| | | | Initial | Screened | Candidate | Screened |
| AP | 1 | 360 | 77 | 50 | 1 | 1 |
| | 2 | | | 54 | 6 | 4 |
| | 3 | | | 57 | 12 | 4 |
| BN | 1 | 24 | 24 | 24 | 4 | 4 |
| | 2 | | | 24 | 5 | 5 |
| | 3 | | | 24 | 6 | 6 |
| CAP | 1 | 1319 | 325 | 281 | 10 | 10 |
| | 2 | | | 289 | 12 | 12 |
| | 3 | | | 288 | 14 | 14 |
| FP | 1 | 563 | 102 | 92 | 5 | 5 |
| | 2 | | | 95 | 6 | 5 |
| | 3 | | | 97 | 4 | 4 |
| PH | 1 | 112 | 21 | 18 | 2 | 2 |
| | 2 | | | 16 | 1 | 1 |
| | 3 | | | 18 | 2 | 1 |

Table A.2: Average hit rate (AHR) for an ensemble of phalanxes (EPX), a random forests (RF), and a regularized random forests (RRF) averaged over 16 repeats of balanced 10-fold cross-validation for the AID362 assay. Larger AHR values are better. The last two columns show the number of times EPX has larger AHR among the 16 repeats of cross-validation relative to RF and RRF.

| DS | Run | Mean AHR | | | EPX beats | |
|---|---|---|---|---|---|---|
| | | EPX | RF | RRF | RF | RRF |
| AP | 1 | 0.300 | 0.280 | 0.256 | 16/16 | 16/16 |
| | 2 | 0.306 | | | 15/16 | 16/16 |
| | 3 | 0.295 | | | 13/16 | 15/16 |
| BN | 1 | 0.261 | 0.242 | 0.238 | 16/16 | 16/16 |
| | 2 | 0.299 | | | 16/16 | 16/16 |
| | 3 | 0.285 | | | 16/16 | 16/16 |
| CAP | 1 | 0.363 | 0.267 | 0.171 | 16/16 | 16/16 |
| | 2 | 0.355 | | | 16/16 | 16/16 |
| | 3 | 0.368 | | | 16/16 | 16/16 |
| FP | 1 | 0.315 | 0.266 | 0.174 | 16/16 | 16/16 |
| | 2 | 0.323 | | | 16/16 | 16/16 |
| | 3 | 0.306 | | | 16/16 | 16/16 |
| PH | 1 | 0.227 | 0.216 | 0.168 | 14/16 | 16/16 |
| | 2 | 0.212 | | | 7/16 | 16/16 |
| | 3 | 0.218 | | | 9/16 | 16/16 |

Table A.3: The number of variables, initial groups, screened groups, candidate phalanxes, and screened/army of phalanxes for the 3 runs of the algorithm to AID364 assay.

| DS | Run | Number of | | | | |
|---|---|---|---|---|---|---|
| | | Variables | Groups | | Phalanxes | |
| | | | Initial | Screened | Candidate | Screened |
| AP | 1 | 380 | 78 | 63 | 6 | 5 |
| | 2 | | | 71 | 6 | 6 |
| | 3 | | | 65 | 5 | 5 |
| BN | 1 | 24 | 24 | 24 | 4 | 4 |
| | 2 | | | 24 | 3 | 3 |
| | 3 | | | 24 | 4 | 4 |
| CAP | 1 | 1585 | 394 | 316 | 7 | 7 |
| | 2 | | | 321 | 10 | 10 |
| | 3 | | | 380 | 10 | 7 |
| FP | 1 | 580 | 104 | 100 | 4 | 4 |
| | 2 | | | 101 | 3 | 3 |
| | 3 | | | 102 | 6 | 6 |
| PH | 1 | 120 | 21 | 18 | 3 | 3 |
| | 2 | | | 16 | 1 | 1 |
| | 3 | | | 16 | 2 | 2 |

Table A.4: Average hit rate (AHR) for an ensemble of phalanxes (EPX), a random forests (RF), and a regularized random forests (RRF) averaged over 16 repeats of balanced 10-fold cross-validation for the AID364 assay. Larger AHR values are better. The last two columns show the number of times EPX has larger AHR among the 16 repeats of cross-validation relative to RF and RRF.

| DS | Run | Mean AHR | | | EPX beats | |
|---|---|---|---|---|---|---|
| | | EPX | RF | RRF | RF | RRF |
| AP | 1 | 0.291 | 0.265 | 0.230 | 16/16 | 16/16 |
| | 2 | 0.292 | | | 16/16 | 16/16 |
| | 3 | 0.310 | | | 16/16 | 16/16 |
| BN | 1 | 0.371 | 0.327 | 0.300 | 16/16 | 16/16 |
| | 2 | 0.365 | | | 16/16 | 16/16 |
| | 3 | 0.373 | | | 16/16 | 16/16 |
| CAP | 1 | 0.379 | 0.334 | 0.252 | 16/16 | 16/16 |
| | 2 | 0.390 | | | 16/16 | 16/16 |
| | 3 | 0.390 | | | 16/16 | 16/16 |
| FP | 1 | 0.318 | 0.305 | 0.261 | 15/16 | 16/16 |
| | 2 | 0.320 | | | 16/16 | 16/16 |
| | 3 | 0.317 | | | 14/16 | 16/16 |
| PH | 1 | 0.278 | 0.275 | 0.219 | 11/16 | 16/16 |
| | 2 | 0.276 | | | 9/16 | 16/16 |
| | 3 | 0.282 | | | 14/16 | 16/16 |

Table A.5: The number of variables, initial groups, screened groups, candidate phalanxes, and screened/army of phalanxes for the 3 runs of the algorithm to AID371 assay.

| DS | Run | Number of | | | | |
|---|---|---|---|---|---|---|
| | | Variables | Groups | | Phalanxes | |
| | | | Initial | Screened | Candidate | Screened |
| AP | 1 | 382 | 78 | 58 | 5 | 4 |
| | 2 | | | 59 | 5 | 4 |
| | 3 | | | 61 | 3 | 3 |
| BN | 1 | 24 | 24 | 24 | 3 | 3 |
| | 2 | | | 24 | 5 | 5 |
| | 3 | | | 24 | 3 | 3 |
| CAP | 1 | 1498 | 362 | 192 | 7 | 7 |
| | 2 | | | 196 | 9 | 9 |
| | 3 | | | 202 | 7 | 6 |
| FP | 1 | 580 | 104 | 85 | 5 | 5 |
| | 2 | | | 85 | 2 | 2 |
| | 3 | | | 88 | 3 | 3 |
| PH | 1 | 119 | 21 | 15 | 5 | 5 |
| | 2 | | | 15 | 4 | 4 |
| | 3 | | | 14 | 3 | 3 |

Table A.6: Average hit rate (AHR) for an ensemble of phalanxes (EPX), a random forests (RF), and a regularized random forests (RRF) averaged over 16 repeats of balanced 10-fold cross-validation for the AID371 assay. Larger AHR values are better. The last two columns show the number of times EPX has larger AHR among the 16 repeats of cross-validation relative to RF and RRF.

| DS | Run | Mean AHR | | | EPX beats | |
|---|---|---|---|---|---|---|
| | | EPX | RF | RRF | RF | RRF |
| AP | 1 | 0.327 | 0.315 | 0.281 | 16/16 | 16/16 |
| | 2 | 0.331 | | | 16/16 | 16/16 |
| | 3 | 0.328 | | | 16/16 | 16/16 |
| BN | 1 | 0.342 | 0.335 | 0.333 | 16/16 | 16/16 |
| | 2 | 0.354 | | | 16/16 | 16/16 |
| | 3 | 0.338 | | | 13/16 | 13/16 |
| CAP | 1 | 0.390 | 0.347 | 0.310 | 16/16 | 16/16 |
| | 2 | 0.384 | | | 16/16 | 16/16 |
| | 3 | 0.378 | | | 16/16 | 16/16 |
| FP | 1 | 0.358 | 0.362 | 0.338 | 3/16 | 15/16 |
| | 2 | 0.358 | | | 4/16 | 14/16 |
| | 3 | 0.364 | | | 12/16 | 16/16 |
| PH | 1 | 0.277 | 0.277 | 0.282 | 9/16 | 5/16 |
| | 2 | 0.284 | | | 15/16 | 10/16 |
| | 3 | 0.279 | | | 9/16 | 6/16 |