Testing backtrack search optimization algorithm as ensemble selection method

Project Report for Numerical Introductory Course submitted

to

Prof. Dr. Brenda López Cabrera

Humboldt-Universität zu Berlin
School of Business and Economics
Chair of Statistics

by

Shikhar Srivastava
(570347)

Master’s Program in Economics and Management Science (MEMS)
Berlin, July 23, 2017

Abstract

With accelerating growth in research field of parametric-modeling techniques, ensemble selection has become integral part in procedure of inductive inference. In this study, I have tested capability of a relatively new search optimization method known as "backtrack search algorithm" as a method of ensemble selection for binary classification problems. The ensemble selection is performed over thousands of model candidates with high prediction diversity. Given the large search-space, the aim of the experiment was to obtain an objective view of algorithm's search efficiency. For this, testing was performed over four different datasets obtained from UCI machine learning library, and the predictions were compared against a well established method of ensemble modeling called "stack generalization or stacking". The predictions were also compared against the base classifier's predictions. Furthermore, a pruned search-space was also used to check efficiency and resource consumption trade-off. The results of the this experiment makes a promising case in favor of the new ensemble search technique.

Contents

List of Abbreviations

BS	Backtracking search
ANN	Artificial Neural Networks
ROC	Receiver operating characteristic
GBM	Gradient Boosting Machine
LogR	Logistic Regression
ML	Machine Learning
SVM	Support Vector Machine
NB	Naive Bayes
AUC	Area under the curve
RF	Random Forest

List of Figures

List of Tables

1 Introduction

Major hurdles in decision making are usually associated with the uncertainties of proposed actions and ideas of repercussions that follows. Being a self-conscious species we are aware of our bounded rationality which limits the viable solutions derived by our own theories. Models step in when theories are too complex to understand(Frigg and Hartmann 2006). The science of predictive modeling has been of particular interest since last few decades. Other than the obvious interests in forecasting future, the increased capacity of capturing, compiling, handling and learning from magnanimous amount of data has much to contribute. Scientists and researchers from all kinds of fields are turning towards computational methods for complex calculations, and implementing well-established learning algorithms including the ones inconceivable before computer age. These data hungry algorithms are being used for exposing the hidden intertwined relationships and extracting knowledge from the data. Some of the interesting examples are: previously unrecognized, stromal morphologic structure found to be as a prognostic determinant for breast cancer (Cireşan et al. 2013), predictions of the impacts of global climate change (van Dam 2003; Cameron et al. 2000), predictions of groundwater systems (Bredehoeft and Konikow 1992; Anderson and Woessner 1992; Oreskes et al. 1994) and adoption of Neural Networks (NN) to predict human caused wildfire occurrence (Lee et al. 1996).

In frequentist regime of data modeling, a variety of classification methods have been developed for inductive inference. The diversity in these methods arose due to large number of theoretical estimate of data distributions. And, in parametric modeling the complexity increases with increase in dimensions of multivariate datasets. The predictions, thus made, from these classification methods are often dissimilar. In present day scenario, large sized sequential datasets are being produced(recorded) in all sectors of relevance. Many a times high accuracy of predictions are of prime importance (like aforementioned examples). Ensemble learning from multiple classifiers has become widely popular solution due to its proven ability of outperforming single classifier systems (Lee et al. 1996). Though computationally expensive, ensemble learning methods holds the attention of researchers due to its underlying simplicity.

This paper is a comprehensive report of experiment performed for improving prediction over multiple models produced from 8 different base classifiers. The paper serves the purpose of comparing methods of improving prediction which can be further used in real-world business cases, applied sciences etc.. One of the ensemble learning is a relatively new search optimization method called 'backtrack search algorithm' which is compared against a well established ensemble learning method called 'stacked generalization or stacking'. I have used bagging and K-fold cross validation methods to introduce high diversity and large search space for ensemble methods to optimize upon. A careful modeling approach has been applied with application of various methods to adjust variability and decrease bias in the training data for models. To get an objective estimate the test is performed on four classification based datasets.

The paper is organized as follows. Section 2 details on the background of ensemble learning. Section 3 gives an explanation of what the two ensemble methods are. Section 4 provides a brief description of Classifier algorithms used for ensemble learning. Section 5 and 6 describes the data used and experimental setup respectively. Section 6.6 provides a short description of performance metrics used for judging the results. Section 7 compares the results of ensemble methods against benchmarks. Section 8 discusses the shortcomings. Conclusion and future work with respect to the experiment is provided in section 9.

2 Ensemble Learning - basis and related work

2.1 Background

It has been well theorized (Polikar 2006; Nanculef et al. 2006; Banfield et al. 2005) that just one algorithm is not always best for all kinds of data, and multiple algorithms predict outcomes with errors in different directions. The intuitive solution, hence arises, is combining the model predictions to reach a more accurate result. Ensemble learning is one of such combining methods. Fundamentally, ensemble learning can be viewed as applying the 'wisdom of the crowd' by taking a weighted vote of each model's prediction and combining them to generate one ensemble. The process of ensemble learning, by it's nature, has two-steps of modeling. The first step is concerned with building of base candidates using classifier(s). The second step is applying ensemble technique to select(and combine) from these candidates. There are two types of ensemble learning or forecast combination methods- (a) Homogeneous, (b) Heterogeneous (Tsoumakas et al. 2009). Homogeneous ensemble algorithms use multiple candidates (models) of only one type of classifier for predicting outcomes. These multiple candidates can either be generated by using different parametric values or by creating subsamples of training set. Heterogeneous algorithms, on the other hand, combine candidates from diverse set of classifiers for predictions.

The experiment will use some of the well-established machine learning classification algorithms. Each method has a completely divergent way of understanding and predicting the data. Also, since the real world data does not exactly follow any of the theorized methods, these algorithms are prone to create errors. Ensemble learning gathers information from multiple sources (algorithms).

2.2 Literature Review

Even though there were a few contributions to this domain in the late 70s and the 80s, research in ensemble methods picked up its speed in the 90s and is still a relevant area of interest for all personnels concerned with statistics, machine learning, pattern recognition, and knowledge discovery in databases. Figure 1 is obtained from a small search on Google's Ngrams service which checks word frequencies in texts (mostly academic). Dasarathy and Sheela (1979) presented the first known work in ensemble learning where they applied k-nearest neighbor and linear models to generate classifiers for their composite classifier system to combine the classifiers for achieving higher accuracy in prediction. The paper addressed increased computation problems as the major obstacle in further development.

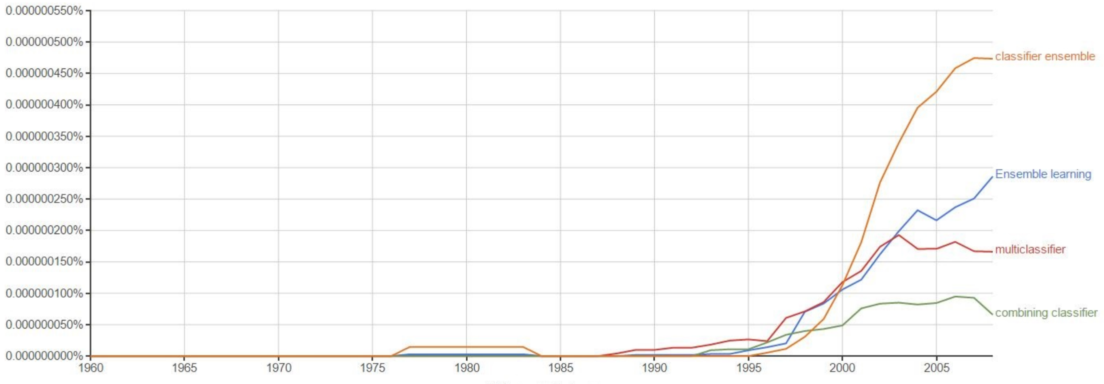


Figure 1: Ensemble learning related work in literature history

With the growth in computational systems, research on this subject increased abruptly. Schapire (1990) used 'Boosting' as ensemble of weak models. The research was a response to question by Kearns (1988). The work by Schapire (1990) had a significant impact on future research in the field and defined structures of many other children algorithms (Freund et al. 1999). On the other hand, researchers saw the potential in combining models created from different classification methods. However, recognizing the problem of exponential increase in number of models, sub-selection or 'pruning' of candidate models' library grabbed considerable interest. According to Tsoumakas et al. (2009), there are three major pruning techniques i.e. *ranking, clustering* and *optimizing*. *Ranking* involves comparing the models on the basis of their individual performances and choosing them in that order. This method has failed to produce any significant improvement over base classifiers (Yang et al. 2005). *Clustering* method uses traditional clustering techniques such as hierarchical or k-means for discovering the groups with low intra-group and high inter-group diversity in predictions (Giacinto et al. 2000; Qiang et al. 2005). Usually, one representative of each group is then used for ensemble averaging. There can be many variations of this using different distance method in-between the clusters. However, an interesting method was applied by Bakker and Heskes (2003), they trained a new model for every cluster and used them for ensemble learning. Finally, *optimizing* involves pruning or weight based aggregating using certain sort of maximization function. These methods are especially interesting because they borrow normative optimization techniques (not specifically for machine learning domain) and use them as ensemble methods. For example, Zhou and Tang (2003) used Gasen-b method of Genetic Algorithm class for optimal model combinations, Zhang et al. (2006) used semi-definite programming method which formulated ensemble sub-selection as mathematical problem and last but not the least, hill-climbing methods (both forward and backwards) have been used widely for ensemble selection which have shown promising results(Caruana et al. 2006; Fan et al. 2002; Banfield et al. 2005; Yang et al. 2005). Other than these methods, there are also some statistical techniques to identify degree of diversity amongst the model library as tool for pruning (Tsoumakas et al. 2004, 2005).

3 Ensemble Learning - methods used

3.1 Stacked Generalization

Stacked Generalization or stacking applies one or many of the supervised classification algorithm on the candidates library produced using diverse set of classifiers (Wolpert 1992). Classifiers are also called as generalizers and hence, the name of this method. In application, the base level models will be generated first using the requirements of final output of stacked method. For example, if only certain models are selected as ensemble after applying stacking method then only those specifics candidates will be generated. Second layer of algorithm will then be applied on this base set (meta-level) to generate the final predictions.

This experiment used *Random Forest* for stacking over the base library of models. Briefs about working methodology of *Random Forest* have been provided in section 4.

3.2 Backtracking Search

Backtracking Search (BS) is a search optimization technique proposed by Civicioglu (2013). The algorithm is a development over Evolutionary Algorithms (EAs) and works on similar concept. EAs require global exploration and local exploitation abilities. The ability of effectively using entire search space and optimizing for the best solution over previously discovered solution differentiates them from other classical optimization techniques. The success entirely depends upon exploration abilities of the algorithm.

Like other EAs, BS also optimizes using selection, mutation and crossover strategies over trial population. And unlike other EAs, BS uses a non-uniform crossover strategy. In this experiment, I have used BS as ensemble search optimization technique. Using diversified classifiers and data modeling techniques (as explained later in section 4), thousands of models were generated. This created the search space for BS in form of a weight matrix with each cell value representing the weight of a model. As initialization, two parents (P1 and P2) are created with random weights alloted in each cell in accordance with $P_{(i,j)} \sim U(low_j, up_j)$ where (low, up) are the boundary for weights and (i, j) is the cell position. One parent undergoes mutation via $P_{mutated} = P + F \times (P_{old} - P)$, where F is the transformation parameter of BS and used as defined by author of BS i.e. $F = 3 \times rnd$ and $rnd \sim N(0, 1)$. The crossover of this $P_{mutated}$ with the left over parent is performed via creating a map. The map selects the 'genes' or cells of this matrix would undergo the crossover. The selection of these cells decides the number of transformations and acts as the only hyper-parameter of this algorithm. Can be selected from a range of 0 to 1 (1 meaning all cells are mutated), the parameter acts as 'mix-rate'. For each row of the matrix, the mix-rate is multiplied with a random number from $U(0, 1)$ (*runif* function in R, Becker et al. 1988) and the number of columns, to calculate the number of columns to be transformed. A random selection from this uniform distribution selects which cells will get transformed. This mix-rate is overridden randomly (again using the *runif* function) and in this case, only one cell per row is transformed.

The candidate produced is checked for the accuracy and is assigned as new parent when a higher accuracy is achieved.

4 Classification Algorithms

Models are the tools to find out about causal relations of certain processes (Woodward 2003). For example, in 2013, a group of machine learning experts, with no medical knowledge, were able to identify cancerous areas under a human tissue as accurately as a trained pathologist (Cireşan et al. 2013). In this experiment, I aim to exploit such properties of observable reality using available data mining methods. The experiment includes six different algorithms and a variant of two of them. These popular algorithms are thoroughly researched and used for forecasting in various fields. A brief understanding of each would help in understanding how diverse these classification methods are, and also in future development or usage of them. I have explained Random Forest in more detail than the others because it has been used as the generalization algorithm in stacking ensemble method.

4.1 Random Forest (RF)

Based on the functioning of Decision tree learning, Random Forest, a method developed by Ho (1995), is also an ensemble learning algorithm albeit a homogeneous one with decision trees as building blocks in it. If the relationship between features and the response is non-linear and complex then decision trees have significant improvement over classical methods. Decision trees predict the target variable via stratification of the feature space. There are three major steps involved:

1. The set of predictors (input variables) $X_1, X_2, X_3 \ldots X_n$ are divided into K distinct and non-overlapping regions: $R_1, R_2, R_3 \ldots R_n$ in the order of most commonly occurring classes. Construction of these regions can be visualized from Figure 2. Classification error parameter is used for deciding the values on which the regions are divided. Classification error rate is fraction of training observations in a region that do not belong the most common class. The divisions are greedy in nature i.e. rather for values which would end up in minimal overall error rate, the algorithm looks for values which minimizes error rate at the instance of division.

2. For every observation that falls into the region R_k where $k = \{i \mid i \in (1, 2, 3 \ldots n), i > 0\}$, we make the same prediction, which is simply the class prediction for the training observations in R_k.

3. This recursive binary splitting of trees continued till final prediction error is minimized or regions are left with specified number of training data. It is important to note that the more number of trees increases the complexity or in this case specificity of model. Thus, making it possible to get a 0% error prediction on training set with large over-fitting. Third option of stopping the growth of trees is to prune them. Optimal pruning value largely depends on size of the dataset which could be identified using iterative testing.

Fig. 2 represents an abstract view of tree formations.

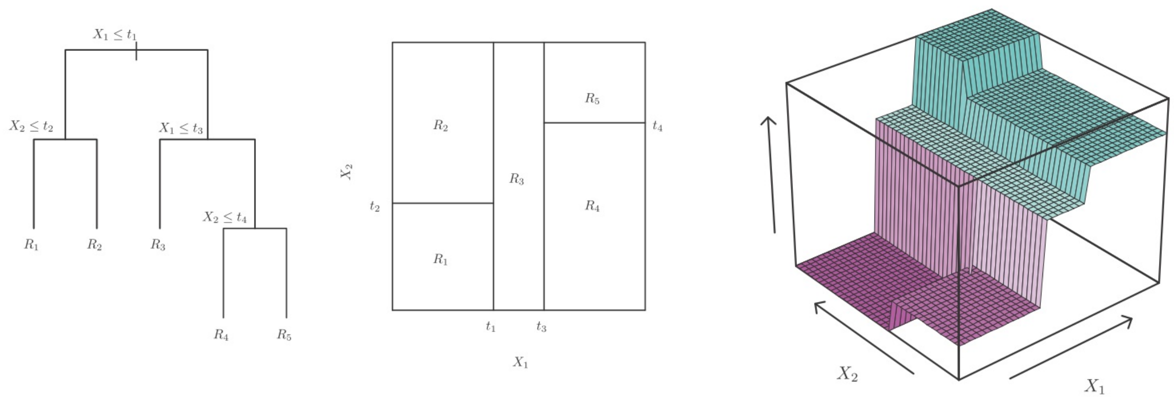


Figure 2: Visualizations of tree based divisions of regions, a tree is split with on individual input's value t. (source: James et al. 2013)

The resultant decision trees suffer from high variance. Basically, this means that if the input data is divided into different parts and decision trees are built on them, the result could be quite different. Bagging or bootstrap aggregation is a well-researched and accepted solution for reducing variance. The mean of n independent observation sets $Y_1, Y_2, Y_3 \dots Y_n$ each with variances σ^2 has a variance of σ^2/n (James et al. 2013). Built upon this concept, the bootstraps are sample sets of original training data with approximately 70% original values and rest 30% repeated samples. The original values are randomly selected in different bootstraps. Decision trees are built upon each of these bootstraps and an average of final predictions $\hat{f}_b(x)$ is calculated by eq. 1 where B is the total number of bags.

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x) \tag{1}$$

Finally, Random Forests, which are further improved over bagging of decision tree method, 'de-correlates' predictions of different bags. It randomly considers m out total p predictors for building decision trees in every split. So, every split gets different set of predictors to build upon. The intuition behind doing this is also the basic idea of ensemble learning. Suppose if one feature is extremely strong as compared to others in creating division. This would result in decision trees of every bag basically being built on this particular feature, which further would mean that the resulting predictions will be highly correlated. Taking an average correlated values will not be useful given the data has high variance. Using random m predictors at each split will allow other predictors have more chance. On average $(p-m)/p$ of the splits will not have that strong predictor. And hence, resulting in uncorrelated predictions. An aggregated result of all the bags thus produced act as a powerful homogeneous ensemble classifier.

Explanation of meta-parameters used-
ntree: Maximum number of trees allowed to be grown.
Mtry: Fraction of predictors randomly selected from all predictions at each split.

4.2 Gradient Boosting (GB)

As explained in previous section, RFs are an improvement over simple bagging method of decision trees. Boosting is yet another aggregation method, like RF, which uses decision trees as base candidates to build a strong aggregated prediction model. However, instead of sub-selection of predictors in each bag for creating the tree, GB uses all data together for building trees but works in a sequential manner. The trees built uses information from previously built tree and instead of building on sub-sampled bags following trees are built on modified data working on the residuals of prediction.

The algorithm starts with $\hat{f}(x) = 0$ and $r_i = y_i$ where f is the fitting function, r is residual and y is the target. A tree $\hat{f}^b$, with pre-defined splits d is built using predictors X and residual r. It is shrunken by a factor of λ for cumulatively updating initial $\hat{f}(x)$. The same shrunken values are also cumulatively subtracted from residual r, setting target values for the next iteration. The final model is hence, a summation of all the trees built through user defined number of iterations. The shrinkage factor λ decides how slow or fast the model will learn. This implies a lower value would require more number of iterations for model to learn better.

Explanation of meta-parameters used-
ntree: Number of trees i.e. the iterations
shrinkage: shrinkage factor λ
minobsinnode: Minimum number of observations in the trees terminal nodes
nTrain: Number of observations on which to train. This adds cross-validation technique over sampling in building final model.

4.3 Support Vector Machine (SVM)

Developed in 1990s, SVMs have been popular for its "out of the box" approach for classification problems. SVM develops over the methodology of Maximal margin classifier, which finds a hyperplane. A hyperplane aims to differentiate between the data corresponding to respective classes. However, real-world data works in a highly non linear manner, and usually such planes doesn't exit. SVM solves this problem by providing a cushion over the optimization constraints of Maximal margin algorithm. In SVM, feature space of input $x \in \mathbb{R}^A$ is non-linearly enlarged to a higher dimension m using 'kernels'.

This transformation is used for finding the linear hyperplane in the feature space using eq.2 where $\phi_i(x)$ is transformation as per kernel function which describes the similarity of two observations and is generalized as given in eq. 3

$$\hat{f}(x) = w_o + \sum_{i=1}^{m} w_i \phi_i(x) \tag{2}$$

$$K(x, x') = \sum_{i=1}^{m} \phi_i(x)\phi_i(x') \tag{3}$$

A linear kernel would mean the classifier is linear in features and in that case, in eq. 3, $\phi_i(x) = x$. On the other hand, as a popular choice, kernel can also use radial basis function as given in equation 4

$$K(x, x') = exp(-\gamma \sum_{i=1}^{m} (x_i - x_{i'})^2) \tag{4}$$

Explanation of meta-parameters used-
degree: parameter needed for kernel of type polynomial
cost: cost of constraints violation, a trade-off between the model complexity and the amount up to which deviations.

4.4 Logistic Regression (LogR)

Logistic regression is the choice algorithm for binary classification problems. Providing prediction output in probability of a class, assumes function (5) and uses Maximum Likelihood method to fit the model.

$$p(x) = \frac{1}{1 + e^{(\beta_0 + \beta_1 X)}} \tag{5}$$

The value of coefficients, here β_0 and β_1, are chosen so as the final result is as close to $p(X)$ as possible. In other words, the output values will be close to 1 for class with 1 value and 0 for observations with 0 values in class. The likelihood function takes the form as given in equation 6.

$$l(\beta_0, \beta_1) = \prod_{i':y_{i'}=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_i')) \tag{6}$$

The coefficient values are chosen to maximize aforementioned function. Mathematical details of maximum likelihood functions are beyond the scope of this experiment report. Explanation of meta-parameters used-
lambda: regularization factor which penalizes the large weights value of coefficients during the iterative fitting process of finding the maximum likelihood function. This helps in reducing overfitting of the model since large weights means model is trying to fit the noise which is not desirable.

4.5 Artificial Neural Networks (ANN) and Extreme Learning Machine (ELM)

Taking its inspiration from design of nervous system in brain for decision making artificial neural networks were conceptually developed in early-1940s but the major improvements of working back-propagation algorithm were not done till mid-1980s (Yadav et al. 2015). In a simplified ANN of two layers, the information is processed via a mathematical activation function (non-linear function) at Neuron. Figure 3 represents this with x_i where components $i = 1, 2, ., n$ as the input features in $\mathbb{R}^n$ and w_{ij},where $i = 1, 2, ., n$ being random weights to each input and j is the respective neuron in hidden layer. At the neuron, activation function is applied on aggregation of the inputs and output is received. The transfer function here is the aggregate with weights applied and activation function of used in this experiment is sigmoid function which is given by eq. 7 where $f(x)$ is linear fit on input variables .

$$\bar{y} = \frac{1}{1 + e^{-f(x)}} \tag{7}$$

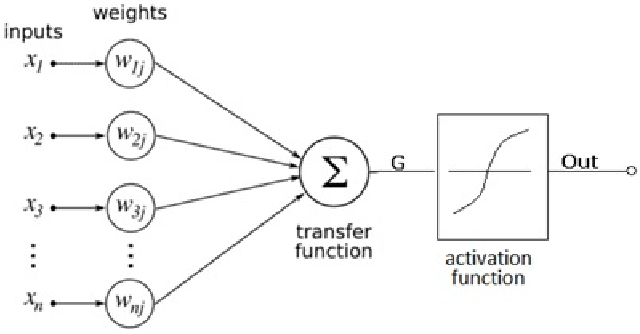


Figure 3: Abstract representation of ANN model.

Now, consider that there are more neurons with different weight distributions for the input and these neurons are further acting as hidden layers. These organized artificial neurons send their signals forward to a final neuron. This system is called a 3 layer (input, hidden and output) neural network. The final error $||y - \hat{y}||_2$ is propagated backwards and revises the random weights given to each neuron of hidden layer. The difference of weights further revises the weight given to input. Hence, the term back-propagation was derived. Figure 4 is a graphic representation of this.

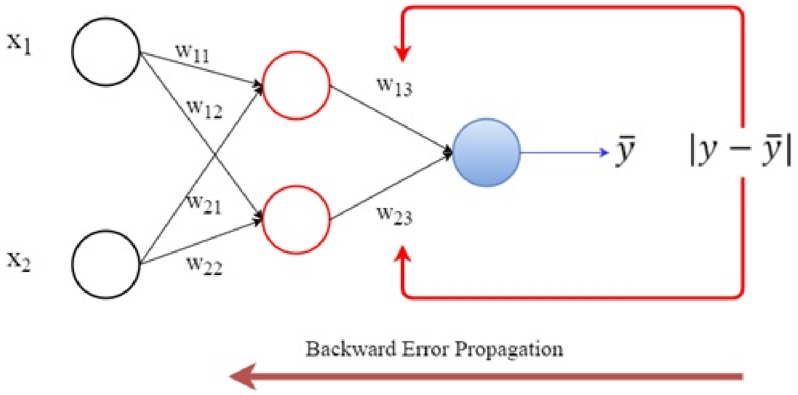


Figure 4: Single Layer Backward Propagation in NN

Hornik et al. (1989) proved that this system can approximate any continuous function which means that the system can learn highly non-linear (complex) relationships in data.
Explanation of meta-parameters used-
Size: Number of neurons in hidden layer
Decay: parameter of weight decay, for removing noise from inputs. Also, large weights of inputs indicate model complexity. Complex models are prone to overfitting. This parameter also controls overfitting.
Maxit- Maximum number of iterations for back propagation.

Extreme learning machine (ELM) are ANNs with only a single hidden layer and instead of a full gradient descent in back-propagation, weights connecting inputs to hidden nodes are randomly assigned and never updated. Explanation of meta-parameters used-
nhid: Number of neurons in hidden layer

4.6 Naive Bayes (NB)

Using the simple bayes rule with an assumption that features are independent of each other, Naive Bayes classifiers have shown efficiency in variety of problems (Sahami et al. 1998; Zhang 2004; Al-Aidaroos et al. 2012). This naive assumption of independence (hence the name) works well for small feature space, however, with increase in features the model accuracy declines. For classification problems the class conditional feature probabilities are defined for all features. So, for input $X = (x_1, x_2, ., x_n)$ with features $i =$

$1, 2, ., n$ a product of conditional probabilities $p(x_i|C_k)$ (with the naive assumption) and the class probability $p(C_k)$ gives probability of the input X belonging to a class C_k where k is the class number. Eq. 8 represents the mathematical notation of this classification $\hat{C}$ prediction. Explanation of meta-parameters used-
fL: Factor for Laplace correction, default factor is 0, i.e. no correction.

$$\hat{C} = \underbrace{argmax}_{k \in \{1,2,.,n\}} p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{8}$$

5 Datasets

For a robust comparative study of learning algorithms I used four datasets. The datasets were downloaded from online machine learning repository of University of California, Irvine. All four datasets belong to binary classification problem. Table 1 details on the description of these datasets. There are many benefits to using this repository because datasets are small and easy-to-load, pre-processed by the researchers, and widely used (for baseline comparisons).

Table 1: Description of datasets used

Dataset	Instances	Attributes	Description
Australian Credit	690	14	credit card application data
German Credit	1000	20	credit history information
Indian Pima Diabetes	770	8	forecasts the onset of diabetes mellitus among Pima Indian women using various health measures
Breast Cancer	680	9	classifying breast cancer as benign or malignant

6 Experimental Setup

I have tried to follow a careful approach to minimize forced errors. The datasets are divided into three subsets after performing ’pre-processing’. The three subsets are called as training, validation and test datasets. Test dataset is left out dataset which is never touched until the final models are built. Traditionally, validation dataset is used to not only estimate but also regulate test error rates (by tweaking meta-parameters and predictors). However, in this experiment the validation dataset was used as training set for ensemble learning algorithms. The ensemble outputs were then used against test data to check their performance. The relevant codes and datasets used to perform this experiment can be found at https://github.com/shikhar-hu/Ensemble-Selection

6.1 Data Pre-Processing

Pre-processing is the first step in predictive modeling. The major reason for doing it is the fact that real world datasets are never clean or model-ready. The techniques of imputation are also often debated as scientists have different point of views. In the ideal scenario, the data collectors/managers should be the one handling these issues as they have the field experience and know the reasons for every missing or outliers. However, such scrutiny is often not possible. In the chosen datasets, there were no missing values. I refrained from performing outliers' analyses since (1) they could mean some extreme scenario in these small datasets, (2) and also due to my limited domain knowledge. Normalization of the predictors is often performed. In the experiment, the predictors were scaled to (-1,1). Dummy or one-hot encoding on categorical variable was performed for better comprehensibility of data to algorithms. Categorical variables have non-incremental values which are often repeated. Dummy encoding transforms a categorical variable into multiple binary (0,1) variables with each representing a unique category of parent variable.

6.2 Data Partitioning

Due to the small size of datasets, a thorough stratified sampling (on each variable) was not possible. Using R's *sample* function, which randomly selects from uniform distribution, with a control on target variable distribution I divided the data in proportions of 60:20:20 for Train, Validation and Test samples respectively.

6.3 Bias and Variance Minimization

After dividing the data, the validation and test sets were kept aside. Considering only the training data, on which the classifier algorithms will be applied to generate library of candidates, we can see that number of observations are quite lesser now. Re-sampling methods are essential tools for obtaining additional information about the fitted model.

K-Fold Cross Validation and Bootstrap Aggregation (bagging) are the two most common and well established statistical methods for doing this.

- K-Fold Cross Validation- In principle, cross validation is performed to minimize uncertainty in test error rates value. Average of all folds (models built by leaving one out) are taken to minimize bias. However, in this experiment for maximizing the learning space from datasets for ensemble algorithms, instead of taking average of models, each model was considered as a separate candidate. The value of K chosen is 5 (because of computational resource limits) which meant there will be a total 5 training samples of 80% the initial training sample size on which the further operations were performed. It is worth mentioning that the random sampling to create folds was applied along with stratifying the target variable class ratio in each sample.
- Bootstrap Aggregation- Also known as bagging, it is a well acclaimed statistics method used for reducing variance and over-fitting which has a simple procedure of random sampling with replacement. Each fold created in previous step was subjected to bagging by randomly selecting 70% from original data and insertion of rest of the 30% by randomly selecting them from the chosen 70% population. This process was done 5 times (again, because of computational resource limits) to make sure that randomness in each sample. Going above 5 bags caused the ensemble algorithm to bloat more than available RAM and hence, resulted in memory error. In total, 30 training sets (25 bags and 5 folds) were used for building candidate library.

6.4 Classifier Specifications

One major time consuming factor in building right predictive models is usually the process of finding out the optimal value of meta-parameters of respective algorithms. This issue can be easily neglected in ensemble learning process. Heterogeneous ensemble algorithms crave for diversity in input data which should ideally improve the output of these algorithms. Using varied values of meta-parameters ensure that the algorithms are trying from understand data from varied perspectives. Thus, creating multiple candidates of each classifier using combination of different meta-parametric values is desirable. And, the problem of finding optimal values can be ignored; only a basic understanding of numerical values taken by the meta-parameters is required. Table 5 in appendix mentions the values of meta-parameters used for building of multiple candidates of each classifier. So, the library finally had 10710 candidates with each bootstrapped sample of each fold used to build all the models using specified parametric combinations.

6.5 Pruning

Because of the large size of search space (model library) the ensemble selection was expected to be computationally expensive. To save computational resources, it is often ideal to sub-select or prune the model library before applying ensemble learning on it. I wanted to check BS and Stacking's efficiency on both kind of datasets i.e full search space and pruned. For pruning, there are various methods like ranking, clustering or hill-climbing (optimization). However, to save time I checked only for Pearson correlation coefficient amongst models and eliminated the ones with 0.85 or higher correlation value with any other model candidate. On average over four datasets, this resulted in a small library of only 1050 candidates.

6.6 Performance metrics

The performance of output models built using different classification algorithms and ensemble methods were compared for finding out the best method. I have used the 'Area under the receiver operating characteristic curve' (AUC) metric which is a widely used method for test evaluation. The predicted values (true or false) and actual values (true or false) are grouped as shown in Table 2. This table is also known as contingency or confusion metric. Sensitivity or True positive rate or recall is given by equation 9. And, (1-specificity) or fall-out rate is given equation 10.

Table 2: Confusion matrix

	Predicted True	Predicted False
Actual True	True positive	False negative
Actual False	False positive	True negative

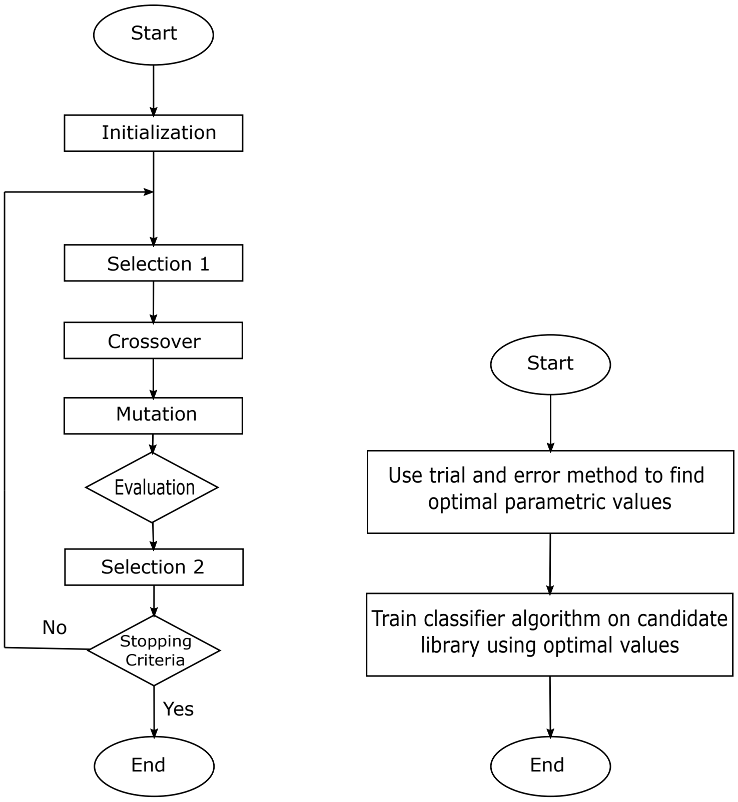
$$True\ positive\ rate = \frac{\sum True\ Positive}{\sum Actual\ True} \tag{9}$$

$$False\ positive\ rate = \frac{\sum False\ Positive}{\sum Actual\ False} \tag{10}$$

Receiver operating characteristic curve plots true-positive rate against false positive rates. The area under this curve represents the probability of correctly classifying 'true' or 'false' classes by the model. The AUC value is one the most popular metric of evaluating model performance. In this experiment I used this metric to compare the models finally created. Also, the BS algorithm used this metric to optimize the ensemble selections.

6.7 Ensemble Learning Setup

Figures 5a and 5b shows graphical representations of both working of ensemble learning algorithms. For Backtrack search algorithm, fig 5a, initialization defines creating of the two parents, 'selection 1' refers to the selection of cells in the weight matrix (see 3.2 for details). 'Selection 2' on the other hand, refers to updating the old parent with new offspring if offspring's AUC values are higher than the parents'. The stopping condition for whole dataset (full search space of 10710 candidates) was set to 1 hour and for the pruned dataset it was 30 minutes. The only parameter, which is dim rate, was set to 0.6 after multiple tests. Figures 6 and 7 in appendix explains the working and behavior of this parameter. For stacking, even though the final training time was less, the search of optimal parametric value was manual task and time consuming.


Start
Initialization
Selection 1
Crossover
Mutation
Evaluation
Selection 2
No
Stopping Criteria
Yes
End
Start
Use trial and error method to find optimal parametric values
Train classifier algorithm on candidate library using optimal values
End

(a) BS

(b) Stacking

Figure 5: Flow of ensemble methods used

7 Results

The top performing candidates of base classifiers were considered as benchmarks. The experiment was performed in R programming language and software environment for statistical computing for Windows x64. Multiple functional packages were used for the setup. The supervising algorithm (RF) applied for stacking ensemble learning required fine tuning. On the other hand, BS method also required more time due to its nature of search. As mentioned before the comparison was performed over two candidate libraries (full and pruned). For pruned dataset the champion of this test problem turns out to be SVM classifier with Linear function, getting best result in two out of the four base datasets.

Table 3: AUC values from Test dataset (Pruned search space)

Algorithm	**Dataset**			
	Australian Credit	German Credit	Indian Pima Diabetes	Breast cancer
BS	0.9107	0.9926	0.8106	0.9889
Stack-RF	0.9289	0.9888	**0.8423**	0.9972
RF	0.9194	0.9869	0.8116	0.9947
GBM	0.9026	**0.9938**	0.8163	0.9977
SVM-L	**0.9301**	0.9818	0.8308	**0.9985**
SVM-R	0.9298	0.9849	0.8224	0.9883
LogR	0.9242	0.9870	0.8363	0.9982
ANN	0.9300	0.9866	0.8163	0.9899
ELM	0.9021	0.9806	0.8359	0.9883
NB	0.9164	0.9787	0.8196	0.9975

However, when full search was provided to ensemble methods, BS showed primacy in three of the four dataset. The comparison of performance scores of all the methods for pruned library and full library are summarized in Table 3 and 4 respectively.

Table 4: AUC values from Test dataset (Full search space)

Algorithm	**Dataset**			
	Australian Credit	German Credit	Indian Pima Diabetes	Breast cancer
BS	**0.9372**	**0.9952**	**0.8444**	0.9970
Stack-RF	0.9072	0.9757	0.8116	0.9972
RF	0.9194	0.9869	0.8123	0.9947
GBM	0.9026	0.9938	0.8163	0.9977
SVM-L	0.9301	0.9818	0.8308	**0.9985**
SVM-R	0.9298	0.9849	0.8224	0.9883
LogR	0.9242	0.9870	0.8363	0.9982
ANN	0.9300	0.9866	0.8163	0.9899
ELM	0.9021	0.9806	0.8359	0.9883
NB	0.9164	0.9787	0.8196	0.9975

8 Discussion

8.1 Effects of pruning

A major part of this experiment was to test the efficiency of BS algorithm against small and large search spaces. When given a large search space, BS algorithm manages to achieve highest prediction accuracy. However, BS performed sub-optimally when highly correlated models were deleted from the library. It is clear that BS algorithm efficiently exploits over small additive improvements. On the other hand, stacking ensemble method performed better than BS in 3 out of the 4 cases on pruned library but single classifiers predictions were better than both in this scenario. One take-away from this result about stacking is that with highly diverse set of candidate's library, stacking promises to perform better than BS algorithm.

8.2 Computational costs and Real-time applications

In the experimental setup, I used 5 number of folds for k-fold sampling and 5 bags for each sample. I planned on creating 10 folds and 8 bags initially. However, the BS algorithm searches for optimal ensemble while loading the whole candidate library in RAM. Even after deploying two 8GB windows machine in parallel, the algorithm was running out of memory and hence, I had to shorten the size of dataset. Whereas, stacking algorithm did not take as much resource. On an average, with different supervisors, it took 70% of RAM memory but not more than 300 seconds of learning.

Whereas, on the applicative side, both models when built can be applied and used quite easily. Only hassle for the users is the initial ensemble building process. The initial setup requires significant resource consumptions but once final model is ready, even real-time datasets, can be used to calculate their predictive scores. The BS ensemble output is a one degree polynomial equation with inputs of all the base candidates. The base candidates in turn use input data over their own equations to calculate the scores. There is no need of rebuilding of models unless the users feel the model has become out-dated.

8.3 Drawbacks of each methodology

From results, we see that individual classifiers (especially SVM-L) outperformed ensemble algorithm when there is a small search library for ensemble methods. However, building an optimal model from individual classifier has it's own hassles.ML based predictive modeling heavily relies on optimal combination of meta-parameters of the base algorithm. Finding out the specific combination depends on the nature of problem, dataset, and selected base learners. This, in turn, makes the optimal values swerve and difficult to find. Consequently, it requires a trial and error method and, hence, increased resource/time consumption. The larger the number of meta-parameters the harder it gets to locate the optimal choices.

On the other hand, BS algorithm outperformed every other method given the large search space. The downside of using this method that it is expensive as well because of its dependency on large distribution of prediction diversity in the search space. However, a higher predictive accuracy gives it the upper-hand for cases where a quality of forecast is of prime importance.

9 Conclusion

The obvious criticism of heterogeneous ensemble learning is its resource or time consumption. However, given the on-going growth in computational systems, and the increasing competition in almost every business domain, high investment in prediction science has become a norm as it facilitates in taking pro-active decisions. This experiment tested and compared a new optimization method, backtrack search algorithm, and provided an objective outlook towards its performance. In three out of four cases studied, when given with a full search space the algorithm outperformed 10710 single classifier models and an optimized stacked model. However, it is also proven from this experiment that the efficiency of BS algorithm degrades with shortening of this search space. In other words, even a little diversity among base predictions is exploited by this method.

It is worth mentioning that given the fact BS algorithm did well without any algorithmic structural changes and it is expected to show significant improvements with even little ensemble selection based changes to it. For example, instead of initiating with random parents, if it is force fed with high performing candidate, the convergence time would significantly reduce. Another example would be to create bags of search spaces to reduce biasness in the training for better search and final aggregation. There is ample amount of scope rising from this research to make a version of ensemble specific BS algorithm for reaching the optimum.

References

AL-AIDAROOS, K., A. BAKAR, AND Z. OTHMAN (2012): "Medical Data Classification with Naive Bayes Approach," *Information Technology Journal*, 11, 1166–1174.

ANDERSON, M. P. AND W. W. WOESSNER (1992): "The role of the postaudit in model validation," *Advances in Water Resources*, 15, 167–173.

BAKKER, B. AND T. HESKES (2003): "Clustering ensembles of neural network models," *Neural networks*, 16, 261–269.

BANFIELD, R. E., L. O. HALL, K. W. BOWYER, AND W. P. KEGELMEYER (2005): "Ensemble diversity measures and their application to thinning," *Information Fusion*, 6, 49–62.

BECKER, R. A., J. M. CHAMBERS, AND A. R. WILKS (1988): "The new S language," *Pacific Grove, Ca.: Wadsworth & Brooks, 1988*.

BREDEHOEFT, J. AND L. KONIKOW (1992): "Groundwater models cannot be validated-Reply," .

CAMERON, D., K. BEVEN, AND P. NADEN (2000): "Flood frequency estimation by continuous simulation under climate change (with uncertainty)," *Hydrology and Earth System Sciences Discussions*, 4, 393–405.

CARUANA, R., A. MUNSON, AND A. NICULESCU-MIZIL (2006): "Getting the most out of ensemble selection," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*, IEEE, 828–833.

CIREŞAN, D. C., A. GIUSTI, L. M. GAMBARDELLA, AND J. SCHMIDHUBER (2013): "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, 411–418.

CIVICIOGLU, P. (2013): "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, 219, 8121–8144.

DASARATHY, B. V. AND B. V. SHEELA (1979): "A composite classifier system design: concepts and methodology," *Proceedings of the IEEE*, 67, 708–713.

FAN, W., F. CHU, H. WANG, AND P. S. YU (2002): "Pruning and dynamic scheduling of cost-sensitive ensembles," in *AAAI/IAAI*, 146–151.

FREUND, Y., R. SCHAPIRE, AND N. ABE (1999): "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, 14, 1612.

FRIGG, R. AND S. HARTMANN (2006): "Models in science," .

GIACINTO, G., F. ROLI, AND G. FUMERA (2000): “Design of effective multiple classifier systems by clustering of classifiers,” in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, IEEE, vol. 2, 160–163.

HO, T. K. (1995): “Random decision forests,” in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, IEEE, vol. 1, 278–282.

HORNIK, K., M. STINCHCOMBE, AND H. WHITE (1989): “Multilayer feedforward networks are universal approximators,” *Neural networks*, 2, 359–366.

JAMES, G., D. WITTEN, T. HASTIE, AND R. TIBSHIRANI (2013): *An introduction to statistical learning*, vol. 112, Springer.

KEARNS, M. (1988): “Thoughts on hypothesis boosting,” *Unpublished manuscript*, 45, 105.

LEE, B., P. WOODARD, S. TITUS, ET AL. (1996): “Applying neural network technology to human-caused wildfire occurrence prediction.” *AI applications*.

NANCULEF, R., C. VALLE, H. ALLENDE, AND C. MORAGA (2006): “Ensemble learning with local diversity,” *Artificial Neural Networks–ICANN 2006*, 264–273.

ORESKES, N., K. SHRADER-FRECHETTE, K. BELITZ, ET AL. (1994): “Verification, validation, and confirmation of numerical models in the earth sciences,” *Science*, 263, 641–646.

POLIKAR, R. (2006): “Ensemble based systems in decision making,” *IEEE Circuits and systems magazine*, 6, 21–45.

QIANG, F., H. SHANG-XU, AND Z. SHENG-YING (2005): “Clustering-based selective neural network ensemble,” *Journal of Zhejiang University-Science A*, 6, 387–392.

SAHAMI, M., S. DUMAIS, D. HECKERMAN, AND E. HORVITZ (1998): “A Bayesian approach to filtering junk e-mail,” in *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62, 98–105.

SCHAPIRE, R. E. (1990): “The strength of weak learnability,” *Machine learning*, 5, 197–227.

TSOUMAKAS, G., L. ANGELIS, AND I. VLAHAVAS (2005): “Selective fusion of heterogeneous classifiers,” *Intelligent Data Analysis*, 9, 511–525.

TSOUMAKAS, G., I. KATAKIS, AND I. VLAHAVAS (2004): “Effective voting of heterogeneous classifiers,” *Machine Learning: ECML 2004*, 465–476.

TSOUMAKAS, G., I. PARTALAS, AND I. VLAHAVAS (2009): “An ensemble pruning primer,” *Applications of supervised and unsupervised ensemble methods*, 1–13.

VAN DAM, J. C. (2003): *Impacts of climate change and climate variability on hydrological regimes*, Cambridge University Press.

WOLPERT, D. H. (1992): "Stacked generalization," *Neural networks*, 5, 241–259.

WOODWARD, J. (2003): "Making things happen: A causal theory of explanation," .

YADAV, N., A. YADAV, AND M. KUMAR (2015): *An introduction to neural network methods for differential equations*, Springer.

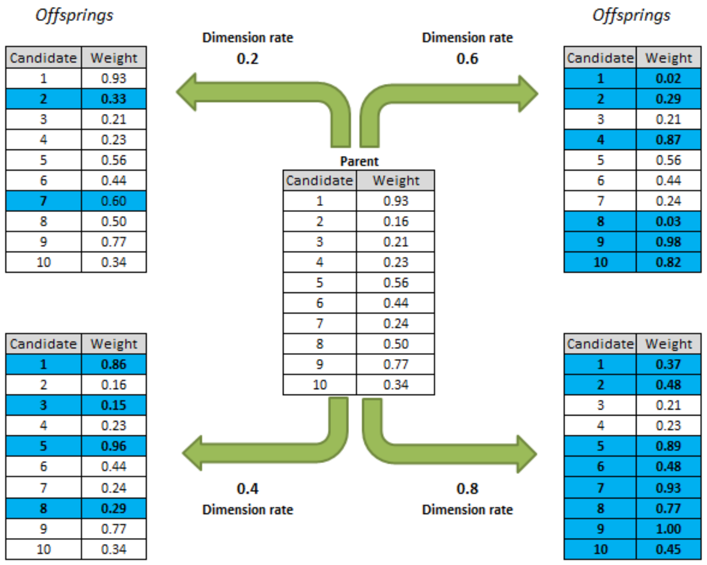
YANG, Y., K. KORB, K. M. TING, AND G. I. WEBB (2005): "Ensemble selection for superparent-one-dependence estimators," in *Australasian Joint Conference on Artificial Intelligence*, Springer, 102–112.

ZHANG, H. (2004): "The optimality of naive Bayes," *AA*, 1, 3.

ZHANG, Y., S. BURER, AND W. N. STREET (2006): "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, 7, 1315–1338.

ZHOU, Z.-H. AND W. TANG (2003): "Selective ensemble of decision trees," *Rough sets, fuzzy sets, data mining, and granular computing*, 589–589.

A Figures and Tables



Candidate	Weight
1	0.93
2	**0.33**
3	0.21
4	0.23
5	0.56
6	0.44
7	**0.60**
8	0.50
9	0.77
10	0.34

Candidate	Weight
1	**0.02**
2	**0.29**
3	0.21
4	**0.87**
5	0.56
6	0.44
7	0.24
8	**0.03**
9	**0.98**
10	**0.82**

Parent

Candidate	Weight
1	0.93
2	0.16
3	0.21
4	0.23
5	0.56
6	0.44
7	0.24
8	0.50
9	0.77
10	0.34

Candidate	Weight
1	**0.86**
2	0.16
3	**0.15**
4	0.23
5	**0.96**
6	0.44
7	0.24
8	**0.29**
9	0.77
10	0.34

Candidate	Weight
1	**0.37**
2	**0.48**
3	0.21
4	0.23
5	**0.89**
6	**0.48**
7	**0.93**
8	**0.77**
9	**1.00**
10	**0.45**

Figure 6: Simplified example for effect of dim rate value on offspring creation

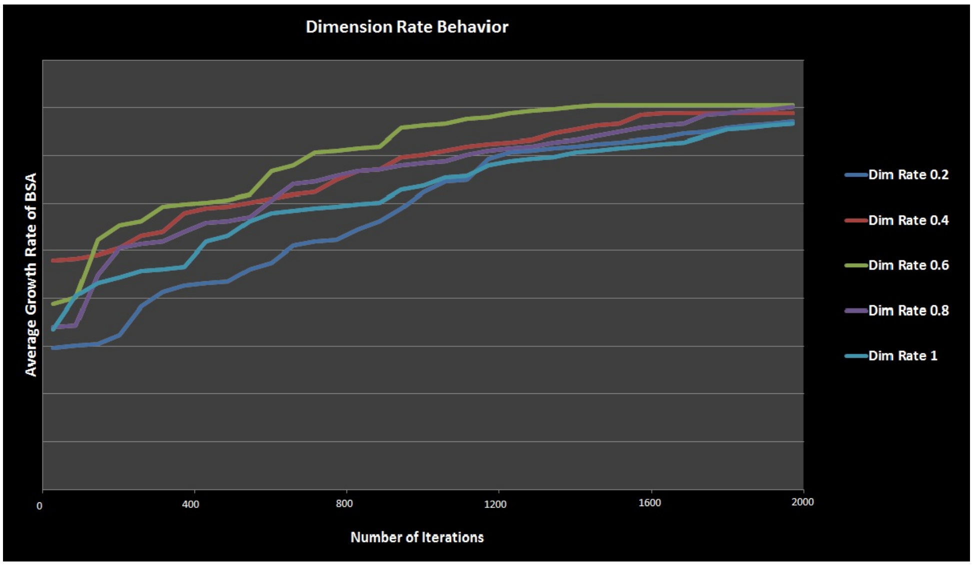


Figure 7: Average learning behavior of different dim rate values on test datasets. 0.6 turned out to be optimum value

Table 5: Classifier model specifications

Classifier	Parametric values	Number of models	Number of candidates
ELM	Activation Function Sigmoid, Pure linear, radial basis nhid 10,20,50, 100 to 1000 with diff of 100	39	1170
Gradient Boosting	ntree 100 to 2200 with diff of 300 shrinkage 0.005,0.01,0.02 minobsinnode 9,11 ntrain 0.7,0.9	96	2880
Logistic regression	lambda 2-̂19 to 26̂ with diff in power of 1 cp aic,bic	26	780
Naive bayes	fL 0,1,2,5,10 UseKernel F and T	10	300
ANN	size 1 to 15 with diff 2 decay 10-̂4 to 100̂ with diff in power of 2 maxit 100,200,1000	72	2160
Random Forest	ntree 100 to 1800 with diff of 100 mtry sqrt(m)/2,sqrt(m),sqrt(m)*2 nodesize ceiling (n/200)	54	1620
SVM Linear	degree 2,3 cost 2-̂10 to 24̂ with diff in power of 2	16	480
SVM Radial	degree 1,2,3,4 cost 2-̂10 to 21̂0 with diff in power of 2	44	1320