Journal of Artificial Intelligence Research 32 (2008) 355-384 Submitted 10/07; published 05/08

Spectrum of Variable-Random Trees

Fei Tony Liu TONY.LIU@INFOTECH.MONASH.EDU.AU
Kai Ming Ting KAIMING.TING@INFOTECH.MONASH.EDU.AU
Gippsland School of Information Technology,
Monash University, Australia

Yang Yu YUY@LAMDA.NJU.EDU.CN
Zhi-Hua Zhou ZHOUZH@LAMDA.NJU.EDU.CN
National Key Laboratory for Novel Software Technology,
Nanjing University, China

Abstract

In this paper, we show that a continuous spectrum of randomisation exists, in which most existing tree randomisations are only operating around the two ends of the spectrum. That leaves a huge part of the spectrum largely unexplored. We propose a base learner VR-Tree which generates trees with variable-randomness. VR-Trees are able to span from the conventional deterministic trees to the complete-random trees using a probabilistic parameter. Using VR-Trees as the base models, we explore the entire spectrum of randomised ensembles, together with Bagging and Random Subspace. We discover that the two halves of the spectrum have their distinct characteristics; and the understanding of which allows us to propose a new approach in building better decision tree ensembles. We name this approach Coalescence, which coalesces a number of points in the random-half of the spectrum. Coalescence acts as a committee of "experts" to cater for unforeseeable conditions presented in training data. Coalescence is found to perform better than any single operating point in the spectrum, without the need to tune to a specific level of randomness. In our empirical study, Coalescence ranks top among the benchmarking ensemble methods including Random Forests, Random Subspace and C5 Boosting; and only Coalescence is significantly better than Bagging and Max-Diverse Ensemble among all the methods in the comparison. Although Coalescence is not significantly better than Random Forests, we have identified conditions under which one will perform better than the other.

1. Introduction

When building ensemble-classifiers, randomisation plays an important role in forming diverse models that are generated from deterministic algorithms. Through the use of ensemble methods, diverse models are aggregated to improve the generalisation capability of the resulting classifiers.

Traditionally, ensemble methods are based on deterministic algorithms with randomisations injected to produce diverse variants. Representatives of these are Bagging (Breiman, 1996a), Random Forests (Breiman, 2001), Randomised C4.5 (Dietterich, 2000) and Random Subspace (Ho, 1998).

Recently, a completely random approach (Fan, Wang, Yu, & Ma, 2003; Fan, 2004; Liu, Ting, & Fan, 2005) is proposed using trees that are generated without any deterministic heuristic; this approach represents a departure from the traditional approaches. In this paper, we show that the complete-random approach and some of the traditional approaches can be used as two extremes to form a continuous spectrum of randomisation; and better predictive accuracy can often be found within the spectrum.

In this paper, we propose a novel algorithm which is capable of generating a range of models, end-to-end continuously from completely random to purely deterministic. The most striking fact is that though each tree-node is created either randomly or deterministically, the resulting randomness can span from completely random to purely deterministic without any modification to the ensemble method. This algorithm enables us to explore the whole spectrum between the two extremes and we show that, this new algorithm can be easily incorporated into existing ensemble methods, such as Bagging and Random Subspace. Together they generate ensembles of different degrees of randomness, which are largely unexplored until now.

We reveal that most of the existing random ensemble methods such as Bagging and Random Subspace focus on the deterministic-end of the spectrum, and ignore a major part of the spectrum. We show that Bagging, Random Subspace and the simple complete-random trees find their better counterparts inside the spectrum.

As there is no known way to measure a priori the level of randomness required for any given problem, we analyse the spectrum and discover that the two halves of the spectrum have their distinctive characteristics. With this new understanding, a new ensemble approach is proposed in this paper, which coalesces a number of points in the spectrum to form the final ensembles. Empirically, we find that this new approach performs better than any single point in the spectrum across a wide range of data sets. This new approach is an off-the-shelf solution, which provides a high level of accuracy without the need of knowing or tuning to the level of randomness required.

This paper is presented as follows. A brief overview of existing decision tree randomisation methods is provided in Section 2. It serves as a primer to decision tree randomisation. The algorithm to generate variable-random-trees is presented in Section 3. In Section 4, the different ensemble methods used in our experiment is introduced, followed by Section 5, which presents a comprehensive empirical evaluation of the spectrum as well as the proposed ensemble approach. Section 6.1 details the key differences between the randomisation framework of Random Forests and the proposed framework of variable-randomness. Other related work is provided in Section 6.2, and we conclude in the last section.

2. Randomisation Methods for Decision Trees

Many randomisation methods have been proposed to produce diverse decision trees for ensemble-classifiers. In this section, as a general introduction to decision tree randomisation, we give an overview of the ways in which they are applied. The following list of decision tree randomisation is not meant to be exhaustive, the purpose of this list is to demonstrate the mechanism and side effects of randomisation methods, which guides us in designing better approaches.

For any conventional decision tree algorithm, one deterministic model is produced for any given training set. Randomisation helps to produce multiple variants of this deterministic model to fulfil the requirement of ensemble learning. A common characteristic of popular methods is that the same heuristic is used in every tree node, which often restricts the possible range of randomness and reduces their impact on performance.

In the literature, most of the proposed randomisation methods can be grouped into three categories, depending on the dimension in which they are applied. The first category is to randomise the **instance dimension**. This includes (i) Bagging (Breiman, 1996a) and Wagging (Bauer & Kohavi, 1999), which generate different sets of training examples through random sampling or assigning randomly generated probability distributions on the given training set; (ii) Output flipping

(Breiman, 2000) in which the classes of the training examples are flipped randomly according to a ratio; and (iii) Adding random examples (Melville & Mooney, 2003) in which diverse classifiers are constructed using additional artificial training examples. A conventional decision tree algorithm is used to generate a model for each random sample of training examples. For type (i), a user has no control over the degree of randomisation applied; for types (ii) and (iii), randomness is at the expense of data integrity.

The second category is to randomise the **feature dimension** by randomly selecting a subset of features before generating each model. A representative of this method is Random Subspace (Ho, 1998) of which 50% of the features are randomly selected to produce models of an ensemble. Random Subspace is not designed to adjust the level of randomness, the default setting as mentioned is commonly used.

The third category is to randomise the **test-selection** at each decision node during the tree growing process. Since it is meant to produce variants of the deterministic model, the randomisation is usually applied in a small degree at each node while maintaining the key deterministic characteristic. Examples of this category are Randomised C4.5 (Dietterich, 2000) and Random Forests (Breiman, 2001). As reported by Breiman, the performance of Random Forests is not significantly impacted by the different values of the parameter used.

For all the methods mentioned above, deterministic models are their common starting point. Randomisations are then injected to produce diverse variants from these deterministic models. On the contrary, a totally different approach is to start with complete-random models, for example, Random Decision Trees (Fan et al., 2003) and Max-Diverse Ensemble (Liu et al., 2005). The distinction between the two starting points is the inclusion of a deterministic heuristic. For any method that uses any deterministic or a weakened heuristic in each node, their starting point is deterministic models. These two starting points seem to be mutually exclusive, however, we provide a way to connect them in order to maximize the possible range of randomness and, in turn, predictive performance gain.

In this paper, we show that a largely unexplored set of randomised models can be found between the extremes of both deterministic and complete-random models. While Random Forests provides a mean to adjust its randomness, the degrees of randomness are constrained by the number of features and the mandatory use of deterministic heuristic at each node. Details of this limitation will be discussed in Section 6.

In the next section, we propose a new algorithm that constructs trees with a controllable randomisation in test-selection. It allows us to explore the whole spectrum of variable-random trees.

3. Trees with Variable Randomness

We name a tree VR-Tree when it is generated using random test-selection in some of its nodes. In this section, we first describe the process of random test-selection and then the mechanism that induces trees with a controllable mix of random and deterministic test-selections.

In the framework of conventional tree building algorithms, random test-selection can be used as a direct replacement of deterministic test-selection. This is depicted in Algorithm 1. First, random test-selection randomly picks a feature from the list of available features to form a decision node. Then, a nominal feature of m possible values will form m branches or a continuous-valued feature with a random cut-point will form 2 branches. The random split-point selection procedure is described in Algorithm 2. This random test-selection becomes an alternative to the deterministic test-selection in the mechanism to create variable-randomness.

Algorithm 1: VR-Tree(D_t, Q, α) - Building a Variable-Random Tree

```
Input: D_t - Training set, Q - Feature set, α - probability of using deterministic test-selection
Output: node - tree node
if all classes ∈ D_t are the same or Q is empty or |D_t| < n_min then    /* n_min is the
minimum number of instances required before a split is
allowed.  */
  | return a leaf with class frequency
else
  | let r be a randomly generated value, where 0 < r ≤ 1
  | if r ≤ α then                         /* Deterministic Test-Selection.  */
  |   | node ← DeterministicTestSelection(D_t, Q)
  | else                                  /* Random Test-Selection.  */
  |   | randomly select an ν ∈ Q
  |   | construct a node with test label ν
  |   | if ν is a continuous-valued feature then           /* Handling a
  |   | continuous-valued feature.  */
  |   |   | node.splitpoint ← RandomSplit(ν, D_t)
  |   |   | D_1 ← filter(D_t, ν > node.splitpoint)
  |   |   | D_2 ← filter(D_t, ν ≤ node.splitpoint)
  |   |   | node.branch(1) ← VR-Tree(D_1, Q, α)
  |   |   | node.branch(2) ← VR-Tree(D_2, Q, α)
  |   | else                    /* Handling a discrete feature.  */
  |   |   | let {v_1...v_m} be possible values of ν
  |   |   | for i ∈ m do                             /* m-ary split.  */
  |   |   |   | D_i ← filter(D_t, ν == v_i)
  |   |   |   | node.branch(i) ← VR-Tree(D_i, Q − ν, α)
  | return node
```

Algorithm 2: RandomSplit(ν, D_t) - Random split point selection

```
Input: ν - a continuous-valued feature, D_t - training data
Output: a split point
r_1 ← randomly select a value of ν in D_t
r_2 ← randomly select a value of ν in D_t
while r_1 == r_2 do
  | r_2 ← randomly select a value of ν
return the mid point between r_1 and r_2
```

To generate variable-randomness, the test-selection process is split into two stages at each node. The first stage decides which test-selection to use, either *random* or *deterministic* test-selection. The second stage proceeds with the selected test-selection to produce the actual test for the node. An α parameter is provided to control the probability of choosing deterministic test-selection over the random one, where $0 \leq \alpha \leq 1$. α also approximates the percentage of deterministic nodes generated in trees. Note that by setting α = 1, this procedure generates trees which are identical to conventional decision trees; and by setting α = 0, it generates complete-random trees. The procedure of the above mechanism can be found in Algorithm 1.

In the next section, we introduced the three ensemble methods used in our experiment based on VR-Trees.

4. Ensemble Methods

Using VR-Tree as the base learner, we explore three ensemble methods that are employed in this investigation. They are listed as follows:

- *Aggregating*, in which trees are generated from the same training data using the full set of features.
- *Subspacing*, in which trees are generated with subsets of randomly selected features. γ parameter is used to determine the percentage of features to be used.
- *Bagging*, in which trees are generated from a bootstrap sample using the full set of features.

The details of these ensemble methods are shown in Algorithms 3, 4 and 5.

Algorithm 3: Agg.VR-Trees(D_t, Q, N, α)

Input: D_t - Training set, Q - Feature set, N - Number of trees, α - probability of using deterministic test-selection
Output: E - a collection of trees
for $i \in N$ **do**
  $E \leftarrow E \cup$ VR-Tree(D_t, Q, α)
return E

Algorithm 4: Subspace.VR-Trees$(D_t, Q, N, \alpha, \gamma)$

Input: D_t - Training set, Q - Feature set, N - Number of trees, α - probability of using deterministic test-selection, γ - the percentage of features used, where $0 < \gamma \leq 1$
Output: E - a collection of trees
for $i \in N$ **do**
  $Q_s \leftarrow$ randomly generate a set percentage γ of features from Q
  $E \leftarrow E \cup$ VR-Tree(D_t, Q_s, α)
return E

While none of these ensemble methods are new, the incorporation of VR-Tree as the base learner help to unleash the potentials of these methods. The predictive performance gain is shown in Section 5. Note that Subspacing is equivalent to the Random Subspace method (Ho, 1998) when γ=50%. Since γ=50% provides the maximum number of distinct subspaces, which is an important factor to increase diversity, we will use γ=50% as the default setting for Subspacing. Also, notice that Bag.VR-Trees with α = 1 is equivalent to the conventional Bagging method (Breiman, 1996a).

Algorithm 5: Bag.VR-Trees(D_t, Q, N, α)

Input: D_t - Training set, Q - Feature set, N - Number of trees, α - probability of using deterministic test-selection
Output: E - a collection of trees
for $i \in N$ **do**
  $D_b \leftarrow$ generate a bootstrap sample from D_t
  $E \leftarrow E \cup$ VR-Tree(D_b, Q, α)
return E

We use probability averaging to combine the outputs from individual models of an ensemble. In order to predict a class given a test case, the predicted class is obtained by:

$$arg \max_y \left(\sum_{i=1}^{N} \frac{n_{i,y}}{n_i}\right), y \in Y \tag{1}$$

where N is the number of trees in an ensemble, $n_{i,y}$ is the number of class y training instances and n_i is the total number of training instances at a leaf of a tree i in which the test case falls into.

5. Empirical Study of The Spectrum

We design our experiment in four parts. The first part investigates the predictive performance spectrum of Aggregating, Bagging and Subspacing using VR-Trees. We then use the result to characterize the two-halves of the spectrum. The second part examines the diversity of base learners generated by these ensembles using the strength and correlation curves as defined by Breiman (2001). This part highlights the range of randomness one can achieve by using VR-Trees. The third part explores an alternative to using only a single α value to produce models in an ensemble. This alternative combines a number of points in the spectrum, which is our proposed ensemble method in this paper. The fourth part investigates the strengths and weaknesses of the proposed method.

Forty-five data sets from the UCI repository (Asuncion & Newman, 2007) are used in this paper. The characteristics of all these data sets are provided in Appendix A. Ten-fold cross-validation is conducted for each data set and the average error rate is reported. 100 trees are used for each ensemble. Random Forests and C5 Boosting (www.rulequest.com) are used as benchmarks, in addition to Bagging, Subspacing and Aggregating of VR-Trees. We use the Friedman test with the Bonferroni test as the post-hoc test at 95% confidence level to compare classifiers (Demšar, 2006).

For the Random Forests implementation used in this paper, the default settings of `mtry = floor(sqrt(q))` and `nodesize=1` is used, where `mtry` is the number of features randomly sampled before each split, `q` is the number of features and `nodesize` is the minimum size of terminal nodes. The implementation is taken from R (www.r-project.org).

Our implementation including of VR-Trees is based on C4.5 (Quinlan, 1993). The default C4.5's stop-splitting rules are applied: (i) the minimum number of training samples $n_{min} = 4$ are required before splitting is considered, and (ii) the deterministic test-selection stops splitting when all possible splits report negative scores. *Gain ratio* is used as the test-selection criterion. Probability averaging is implemented with curtailment (Zadrozny & Elkan, 2001), where the minimum leaf size for probability estimation is always greater than one. These default settings are used for VR-Trees in this paper.

5.1 The Predictive Performance Spectrum of Aggregating, Bagging and Subspacing

Figure 1: The spectrum of predictive performance for Aggregating, Bagging and Subspacing, as well as Bagging plus Subspacing: error rates against α, average over forty-five data sets.

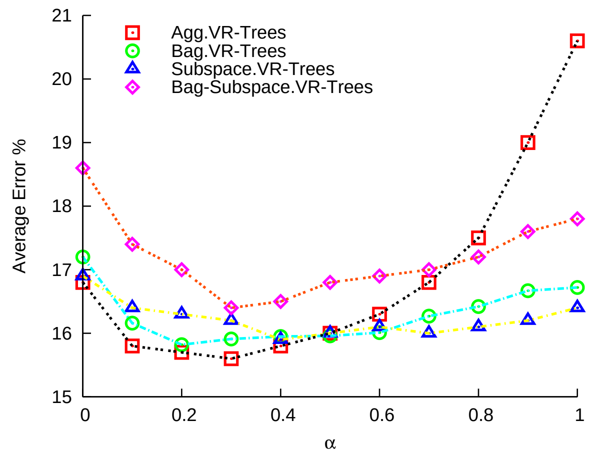


Figure 1 shows the spectrum of performance [1] for four ensemble methods using VR-Trees as the base models. Note that the conventional Bagging and Random Subspace are two points on the deterministic-end (α = 1) of the spectrum for Bag.VR-Trees and Subspace.VR-Trees; and Max-Diverse ensemble is at the complete-random-end (α = 0) of the Aggregating spectrum. Figure 2 shows the results of Friedman tests for each of Aggregating, Bagging and Subspacing and we have the following observations.

i From Figure 2, it is interesting to note that the best operating region for Agg.VR-Trees is having α values between 0.1 and 0.6. This shows that Max-Diverse ensemble which operating at α = 0 can improve its performance by moving further into the middle of the spectrum. The best operating region for Bag.VR-Trees is between 0.1 and 0.6, which is mainly in the first half of the spectrum. This is significantly different from what the conventional Bagging is normally applied at, namely α = 1. The best operating region for Subspace.VR-Trees is between 0.4 and 0.8. This is also different from what Random Subspace is normally applied at α = 1.

ii A balanced mix of random and deterministic heuristics, i.e., α = 0.5, produces the best ensemble or no significantly difference to the best ensemble for any one of the three ensemble methods.

iii Out of the four curves shown in Figure 1, Agg.VR-Trees have the largest swing in performance, followed by Bag.VR-Trees and Subspace.VR-Trees. This is expected as the two end-points in the Agg.VR-Trees' curve represents a single deterministic model and an ensemble of complete-random models. As α decreases from 1 to 0.5, a substantial improvement in predictive performance for Agg.VR-Trees, takes effect due to the increased diversity in the ensemble, which reduces the average error rate from 20.6% to 16.0%.

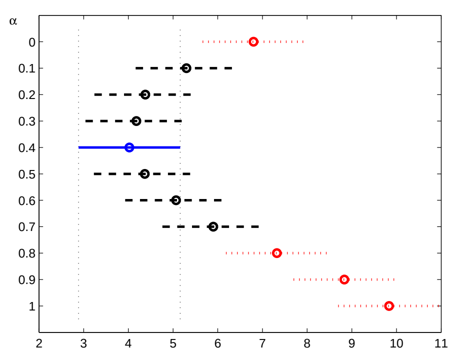
iv Although both Bag.VR-Trees and Agg.VR-Trees perform best in the region of $0 \leq \alpha \leq 0.5$, the result in Figure 1 indicates that they have no significant difference in terms of predictive

1. Averaged over forty-five data sets.

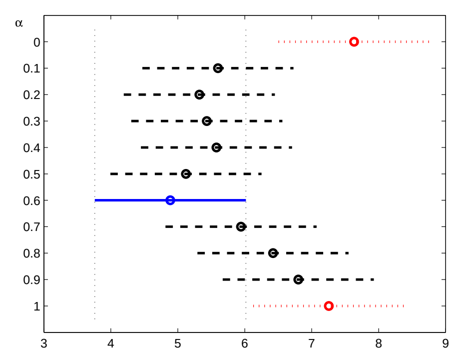
accuracy. Because of additional computational requirement to generate bootstrap samples for Bagging, Aggregating becomes the preferred method in the first half of the spectrum. Bagging and Subspacing are preferred to Aggregating in the second half of the spectrum because the latter is uncompetitive in that region.

v The use of both Bagging and Subspacing in a single ensemble is not recommended, as shown by the result in Figure 1 that Bag-Subspace.VR-Trees always performs worse than its parents, Bag.VR-Trees and Subspace.VR-Trees.

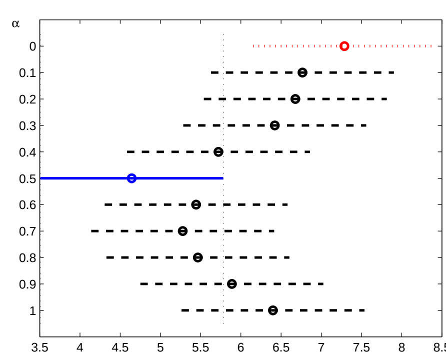
Figure 2: Friedman test results for classifiers produced by (a) Aggregating, (b) Bagging, (c) Subspacing of VR-Trees, from eleven α values in the spectrum over forty-five data sets. We use Bonferroni test as the post-hoc test and $\alpha = 0$ as the control condition for each comparison. The vertical axis indicates the α values, and the horizontal axis indicates the rank values. The circle is the average rank for each α value and the bar indicates the critical values for a two-tailed test at 95% significance level. When two classifiers having no overlapping bars, it indicates that they are significantly different. Significantly worse results are presented in dotted bars (coloured in red) located on the right-hand-side of the diagram. The best results are presented in solid bars (coloured in blue) or the left-most bar in each diagram.



(a) Agg.VR-Trees, $\alpha = 0.4$ has the highest ranking



(b) Bag.VR-Trees, $\alpha = 0.6$ has the highest ranking



(c) Subspace.VR-Trees, $\alpha = 0.5$ has the highest ranking

We summarise the characteristics of the two halves of the spectrum in Table 1. In the next subsection, we continue our analysis over the various points of the spectrum in relation to generalisation error.

Table 1: Characteristics of the two halves of the spectrum.

Complete-Random-end, $\alpha \in [0, 0.5]$	**Deterministic-end,** $\alpha \in (0.5, 1]$
• Models at this extreme end are generated in a completely random fashion.	• Models are variants of a deterministic model.
• Candidate models are all possible trees of larger sizes.	• Candidate models are models of smaller sizes (because each model reaches pure leaves early).
• Models have a higher diversity.	• Maintaining high predictive accuracy while providing some degree of diversity.
• Aggregating is preferred in this region.	• Subspacing and Bagging are preferred in this region.

5.2 Strength-Correlation Analysis

In this section, we examine the strength and correlation profiles produced by Aggregating, Bagging and Subspacing. Firstly, we illustrate the relationship between generalisation error, strength and correlation using a map. Secondly, we plot the strength-correlation curves for the actual data sets to characterise their behaviours. Thirdly, we continue the analysis from Section 5.1 to further explore the two halves of the spectrum in light of the strength-correlation analysis.

Breiman's (2001) inequality, $PE \geq \rho(1-s^2)/s^2$ is useful for discerning the relationship between generalisation error PE, strength s and correlation ρ. Briefly, the strength of an ensemble measures the expected margin function of the ensemble. A margin function is the probability of correct classification minus the maximum probability of incorrect classification of an instance. Correlation of an ensemble is a measure on how similar the predictions are among individual trees. A high reading of correlation indicates that individual trees are making similar predictions. We use the strength and correlation estimation procedures of Buttrey and Kobayashi (2003) as they are the corrected version of Breiman (2001). To make this paper self-contained, the estimation procedures are given in Appendix C.

Figure 3 shows a map of strength-correlation profiles with grey scale indicating the generalisation error. We can see that the error rate is lower at the high-strength and low-correlation corner (at the bottom-right), where the error rate is lower with darker grey level. As for all the ensemble classifiers in general, their goal is to get themselves to a region where the estimated error rate can be as low as possible. Aggregating with different values of α typically spans in the fashion of either curve (a) or curve (b) as shown in Figure 3, where each curve starts from α = 0 (at the bottom-left) to α = 1 (at the top-right). For (a), the lowest error rate can be found at α = 0. For (b), lower error rates can be found with a larger value of α. However, error rates also increase when α approaches 1. In this case, a search would be necessary to determine the optimal α.

Figure 3: Ensemble generalisation error distribution using Breiman's inequality on strength and correlation. Curves (a) and (b) represent two typical spans of Agg.VR-Trees with $0 \leq \alpha \leq 1$.

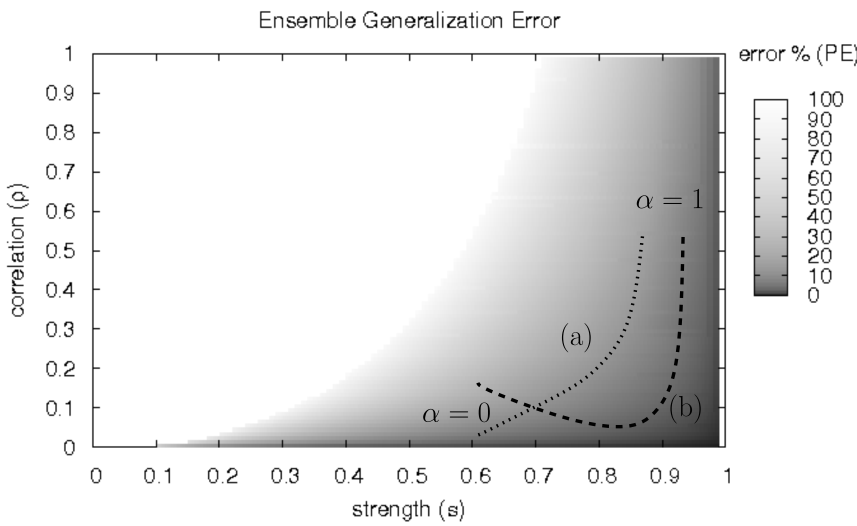


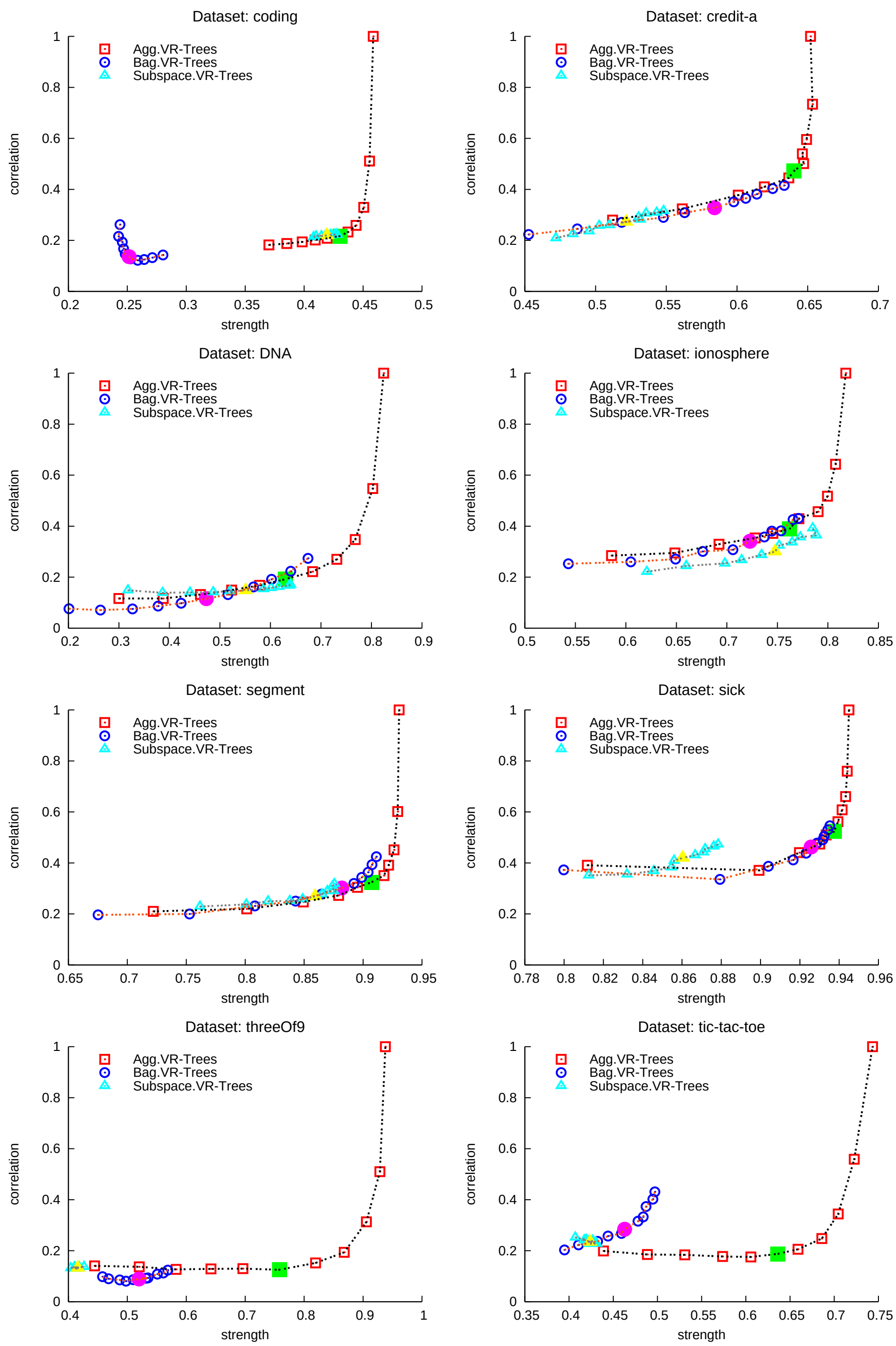
Figure 4 shows that the three different ensemble methods provide different ranges of strength and correlation profile. The most effective way to use VR-Trees is Aggregating. This is because it consistently produces the longest span of strength-correlation curve in each of the data sets we used. Bagging usually has the second longest span, followed by Subspacing. Note that Aggregating usually spans in both strength and correlation dimensions; whereas Bagging and Subspacing have significantly smaller span, especially in the correlation dimension. Table 2 shows the range between the minimum and the maximum values of strength as well as correlation, averaged over forty-five data sets. The result shows that Aggregating produces the largest range of trees in comparison with Bagging and Subspacing. Most interestingly, the best of Aggregating is usually located in a lower or similar error region to those of Bagging and Subspacing.

Table 2: Average ranges of strength and correlation over the forty-five data sets.

	Strength	Correlation
Agg.VR-Trees($\alpha \in [0, 1]$)	0.135	0.696
Bag.VR-Trees($\alpha \in [0, 1]$)	0.136	0.160
Subspace.VR-Trees($\alpha \in [0, 1]$)	0.069	0.088

It is also interesting to note that α=0.5 is always close to or at the changing corner between strength-varying leg and correlation-varying leg in all the examples shown in Figure 4. This means that α = 0.5 is either close to or at the lowest generalisation error region, if the ensemble exhibits

Figure 4: Strength-Correlation distribution of Aggregating, Bagging and Subspacing for different values of α. The solid-filled marks represent α=0.5. For Aggregating, the first half of the α range ($\alpha \in [0, 0.5]$) is characterised by lower correlation and lower strength, and it is located at the bottom-left corner of the diagrams. The second half ($\alpha \in [0.5, 1]$) is characterised by higher correlation and higher strength, and it is located at the top-right corner of the diagrams.



curve (b) like behaviour as in Figure 3. Thus, α = 0.5 often serves as an upper limit in a range of α values that performs well.

Combining the above result with the analysis on the two halves of the spectrum from Section 5.1, we find that Aggregating exhibits the following characteristics in most of the data sets in Figure 4:

- At the first-half of the spectrum, strength increases rapidly from α = 0 and slows down at about α = 0.5; however, correlation only varies in a small degree or not vary at all.

- At the second-half of the spectrum, correlation increases rapidly from α = 0.5 and peaks at α = 1. In this range, both strength and correlation are very high, error rates are not optimal.

In summary, α for better performing Aggregating models can be found in the range between 0 and 0.5. Most single operating points $\alpha \in [0, 0.5]$ have been shown to work well in Section 5.1.

In the next section, we show an alternative approach that achieves even better result by using the range of $\alpha \in [0, 0.5]$ that we have identified thus far.

5.3 Coalescence of Different Operating Points

In this section, we show that combining single models from multiple points in the spectrum will achieve similar or better performance as compared to using any single point in the spectrum. We coalesce VR-Trees with α sampled at a fixed interval. For example, to form a 100-model ensemble, we construct trees with $\alpha = \{0, 0.005, 0.01, ..., 0.495\}$. We call this approach *Coalescence*.

The Coalescence approach is appealing because we do not need to know which α value produces the most accurate ensemble for any given data. By introducing members of different "talents" from the random-half of the spectrum; Coalescence forms a committee of different members. In this committee, as far as we know, some members are good at approximating non-axis-parallel boundary and some members are good at avoiding over-fitting. The make-up of this committee helps to handle unforeseeable conditions that arise in training sample.

As a result, Coalescence provides a comparable predictive performance to a search-for-the-best-α approach (Liu & Ting, 2006) without the cost of searching for the optimal α value. In Figure 5, Coalescence is shown to be better than any single operating point in the first-half of the spectrum using the Friedman test, over forty-five data sets.

An additional comparison is also conducted with some other well known tree ensemble classifiers. The complete result for Coalescence, Aggregating VR-Trees (α=0) (i.e. Max-Diverse Ensemble), Bagging, Subspacing (α=1) (i.e. Random Subspace), C5 Boosting and Random Forests are presented in Table 3. The average error rates over forty-five data sets are provided in the last row of the table. Figure 6 shows the result of the Friedman test and we have the following observations:

- Coalescence ranks the highest among the five benchmarking ensembles in the Friedman test.

- The second highest ranking ensembles: Random Forests and C5 Boosting have almost identical ranking in the Friedman test, and similar average error rates.

- The third highest ranking ensemble is Random Subspace; and the lowest ranking ensembles are Bagging and Aggregating VR-Trees (α = 0). Both ensemble methods have similar average error rates.

Figure 5: Friedman test result for comparing Coalescence with five operating points of Agg.VR-Trees in the first-half of the spectrum. Horizontal axis indicates the rank values. Coalescence ranks top as compared to different points in the first-half of the spectrum.

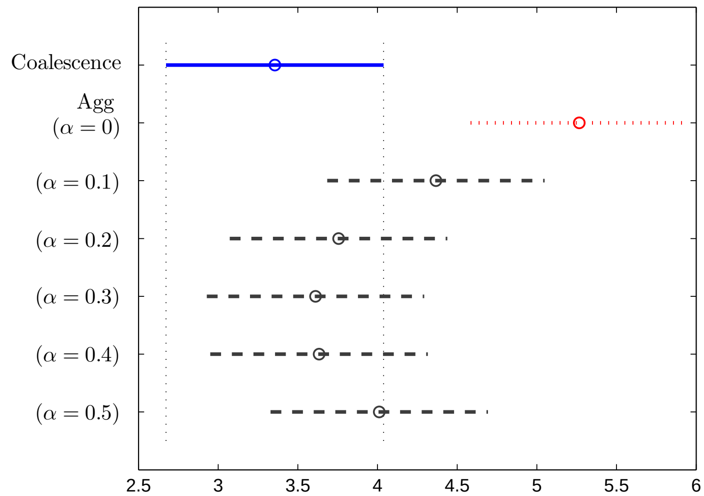


Figure 6: Friedman test result for comparing Coalescence with five benchmarking methods. Horizontal axis indicates the rank values. Notice that Coalescence is the only method that is significantly better than Agg.α = 0 (Max-Diverse Ensemble) and Bag.$\alpha = 1$ (Bagging).

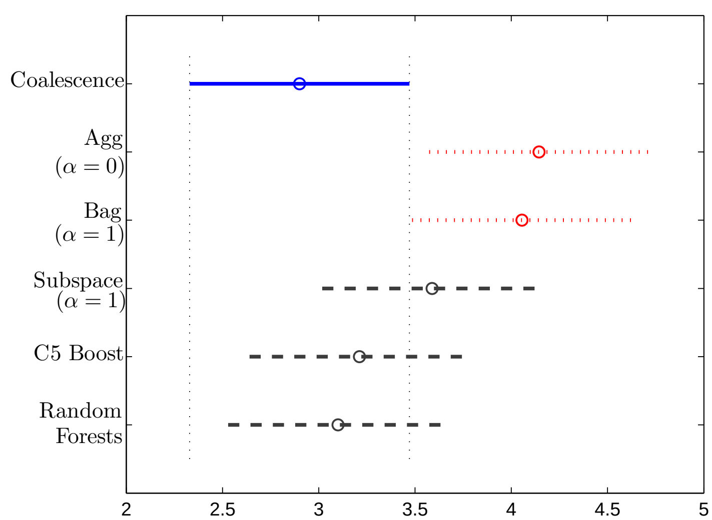


Table 3: Experimental results: Average error rate of ten-fold cross-validation.

data sets	Coalescence	Agg. VR-Trees (α=0)	Bag. VR-Trees (α=1)	Subspace. VR-Trees (α=1)	C5 Boost	Random Forests
abalone	30.0	30.2	30.7	30.0	31.1	30.9
anneal	1.5	1.4	3.5	3.2	5.0	9.7
audiology	17.2	17.7	16.2	20.3	15.0	20.8
autos	18.1	22.5	19.5	14.7	15.6	15.2
balance	14.4	12.3	18.1	11.4	18.9	16.3
breastw	3.0	2.4	3.4	3.0	3.1	3.4
breasty	28.7	25.9	27.3	25.1	26.9	29.7
chess	0.7	1.6	2.3	1.8	0.3	1.1
cleveland	42.9	41.6	44.2	41.9	41.6	41.9
coding	16.4	16.8	24.2	16.7	15.4	17.5
credit-a	12.3	13.0	12.8	13.0	14.3	13.0
credit-g	23.1	25.7	22.7	24.8	22.4	23.2
dna	5.3	26.5	6.0	3.7	4.8	3.4
echo	33.5	34.2	29.0	32.0	37.4	32.0
flare	18.6	19.2	17.7	17.1	17.5	18.5
glass	21.0	22.9	21.5	21.5	21.4	21.0
hayes	18.1	21.9	17.5	17.5	16.9	16.9
hepatitis	18.0	15.5	20.0	18.6	14.1	17.3
horse	13.3	17.9	15.2	16.0	22.5	14.1
hypo	0.8	1.7	0.8	1.3	0.8	1.0
ionosphere	5.7	8.5	5.4	5.4	5.4	6.5
iris	4.7	4.7	4.0	6.0	4.0	3.3
labor	7.0	3.3	10.7	12.0	15.7	5.0
led24	27.9	30.3	28.1	29.1	28.1	28.5
led7	26.7	26.9	26.3	27.5	27.8	26.3
liver	27.0	27.9	27.3	31.9	29.6	26.1
lymph	14.9	14.3	20.4	15.6	19.1	16.9
nursery	0.9	2.2	3.7	5.9	0.9	2.3
pima	23.4	24.6	24.0	23.2	25.0	23.2
post	40.0	36.7	37.8	32.2	30.0	32.2
primary	55.2	57.2	56.6	51.9	56.9	56.9
satimage	8.8	10.4	9.0	8.3	8.1	8.1
segment	2.1	3.1	2.4	2.3	1.8	2.1
sick	2.1	5.7	2.2	4.3	2.2	2.1
solar	29.4	30.3	27.2	27.8	25.7	27.2
sonar	15.9	15.9	24.0	18.7	15.9	13.9
soybean	5.4	6.0	6.6	5.3	6.2	5.7
threeOf9	0.0	0.6	1.4	10.0	0.0	0.8
tic-tac-toe	3.0	9.7	14.3	22.3	1.2	1.3
vehicle	24.5	27.1	24.9	25.4	23.3	26.1
vote	4.1	5.3	4.6	4.6	4.8	4.1
waveform21	14.5	14.7	16.1	14.7	15.6	14.7
waveform40	15.7	17.0	16.5	15.3	15.1	14.9
wine	3.4	1.1	3.4	2.3	5.6	2.3
zoo	1.0	2.0	3.0	4.0	3.0	5.0
mean	15.6	16.8	16.7	16.4	15.9	15.6

- Of all the ensembles, only Coalescence is shown to be significantly better than Aggregating VR-Trees (α = 0) and Bagging in the Friedman test.

It is interesting to note that when employing an oracle to find the best α, the optimal error rate is 14.6% [2] (Liu & Ting, 2006). Thus, the Coalescence approach comes close to this optimal result without using any oracle.

Figure 7 shows the predictive performance of Coalescence in relation to Aggregating in eight data sets. Coalescence sometimes outperforms its 'parents' ($\alpha \in [0, 0.5]$), and often comes close to the best performing Aggregating.

5.4 Strengths and Weaknesses

This section will examine the strengths and weaknesses of VR-Tree. Although Coalescence is not significantly better than Random Forests in the Friedman test, Table 3 clearly shows that Coalescence is better than Random Forests in some data sets but worse in others. This section will also examine some of the conditions under which Coalescence performs better than Random Forests and vice versa.

We find that the ensembles of complete-random-trees have the following strengths: 1) capable of *approximating non-axis-parallel boundary*, and 2) they are highly stable learners in terms of *Pointwise Hypothesis Stability* (Bousquet & Elisseeff, 2002). An analysis that is based on *Pointwise Hypothesis Stability* can be found in Appendix B. To verify the complete-random-trees' ability to approximate non-axis-parallel boundary, in Figure 8, we provide a visualization example using a Gaussian mixture data set (Hastie, Tibshirani, & Friedman, 2001). Figure 8 shows that an ensemble of complete-random-trees (using 100 trees) is able to better approximate the non-axis-parallel boundary as compared to a single deterministic tree.

Moreover, as described in the analysis in Appendix B, one of the strengths of complete-random trees is also one of their weaknesses: complete-random trees trend to *over-fit*; and this problem stems mainly from the ability to approximate non-axis-parallel boundary. We also find that the overfitting problem is aggravated by irrelevant attributes, class noise and small training size as shown in the following empirical examination.

We denote $\mathcal{X}$ as the input space and $\mathcal{Y}$ as the output space. A learning algorithm outputs a function that approximates the underlying true function f by inspecting the training set. A training set $S = \{z_i\}_{i=1}^{m}$, where $z_i = (x_i, y_i)$ and $y_i = f(x_i)$, contains m i.i.d. examples in $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ that are drawn from uniform distribution in the instance space. We define the input space with two variables, $x_i = \langle \nu_{i,a}, \nu_{i,b} \rangle$ and the output space with two possible classes $\mathcal{Y} = \{+1, -1\}$. In this case, the instance space is a square from point $(\nu_a = 1, \nu_b = 1)$ to point $(\nu_a = -1, \nu_b = -1)$.

We define the two concepts as follows:

$$\text{Concept } \mathbf{A}\text{: } f^A(x_i) = y_i = \begin{cases} +1 & \text{if } (\nu_{i,a} > \nu_{i,b}) \\ -1 & \text{else} \end{cases}, x_i = \langle \nu_{i,a}, \nu_{i,b} \rangle$$

$$\text{Concept } \mathbf{B}\text{: } f^B(x_i) = y_i = \begin{cases} +1 & \text{if } (\nu_{i,a} > 0) \\ -1 & \text{else} \end{cases}, x_i = \langle \nu_{i,a}, \nu_{i,b} \rangle$$

2. Averaged over the same forty-five data sets used in this paper.

Figure 7: Detail results comparing Coalescence with Aggregating in eight data sets.

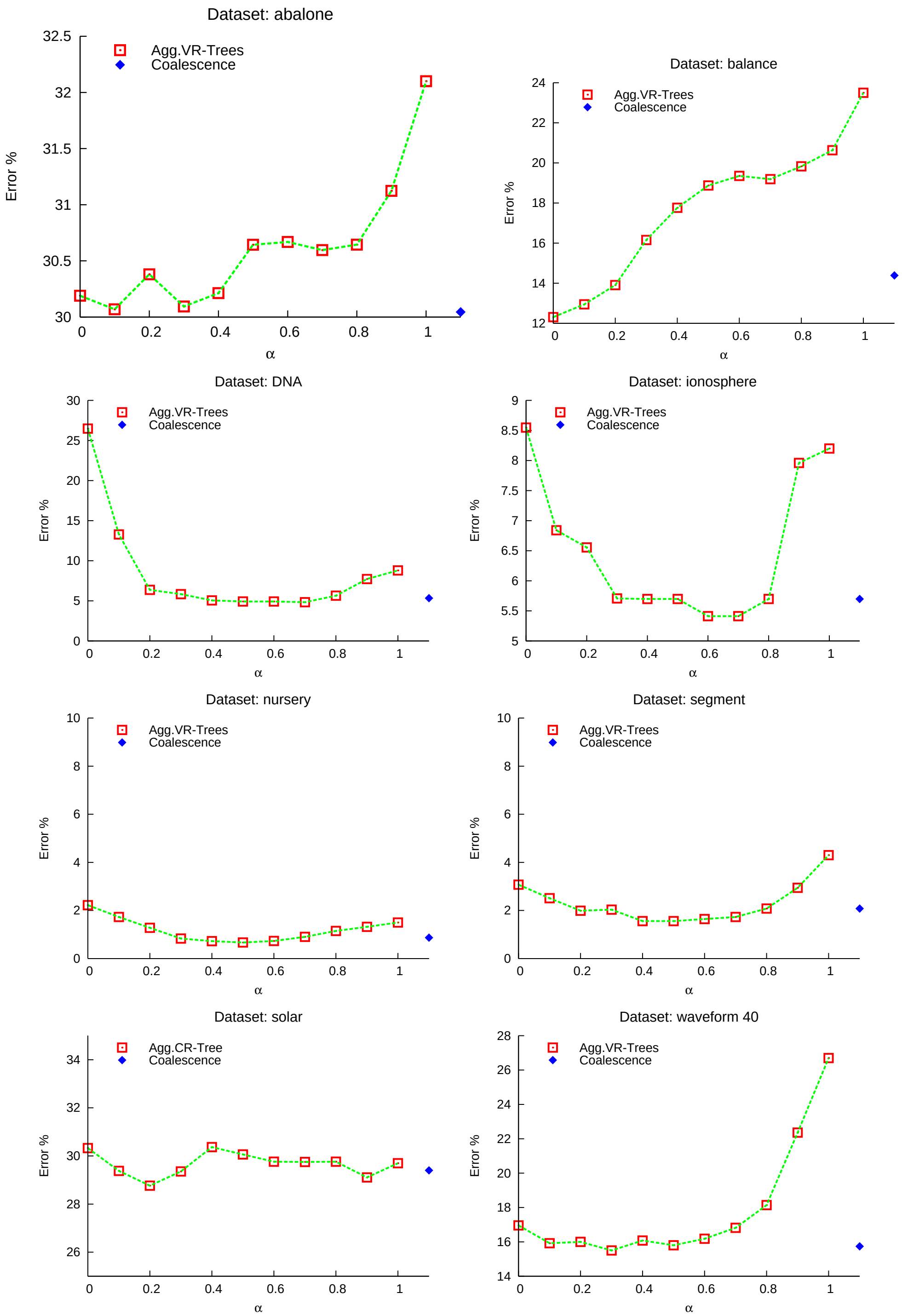
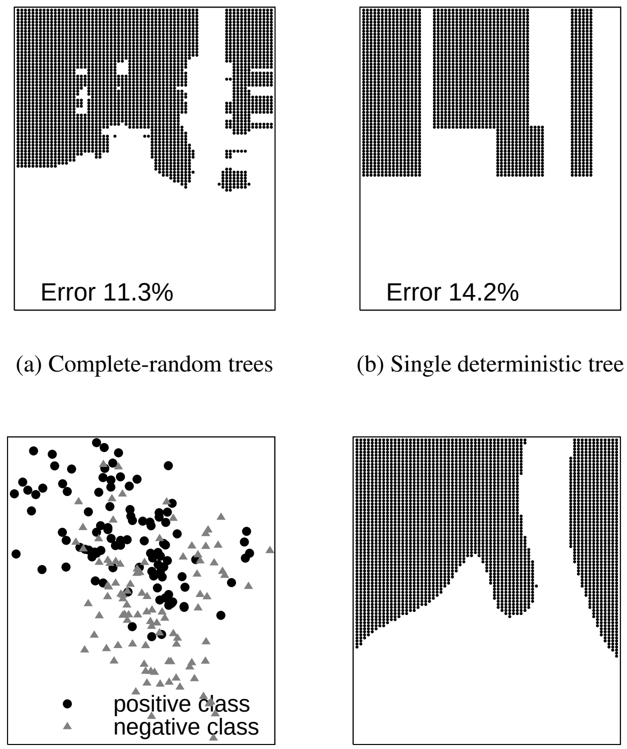


Figure 8: Visualisation of a non-axis-parallel decision boundary on Gaussian mixture data using an ensemble of complete-random-trees as compared to a single deterministic tree (C4.5).



(a) Complete-random trees (b) Single deterministic tree

(c) Training sample (d) Actual decision boundary

Concept **A** is useful for illustrating the ability to approximate non-axis-parallel boundary. Concept **B** is an axis-parallel concept, which is used as a control condition in this experiment. We conduct four experiments using (a) a training sample of $k = 1024$ instances, (b) a training sample with irrelevant attributes, (c) a training sample with class noise and (d) a size-reduced training sample of $k = 64$. We train ensemble models using Coalescence, VR-Trees ($\alpha = 0$, $\alpha = 0.5$) and Random Forests. We use 100 trees for each ensemble. Finally, we evaluate the models by 10000 lattice samples from the instance space, and the average error rate over 10 runs is reported.

Table 4 shows that:

- VR-Trees ($\alpha = 0$) is the best performer in approximating non-axis-parallel boundary in concept **A**, even with small training size of $k = 64$. However, it is the worst performer in axis-parallel concept **B** in both $k = 1024$ and $k = 64$.

- VR-Trees ($\alpha = 0.5$) performs the best under class noise and irrelevant attribute conditions in both concepts **A** and **B**. When irrelevant attributes are added, the error rate of VR-Trees (α = 0) increases at a faster rate as compare to VR-Trees (α = 0.5). Similarly, class noise affects the complete-random-end (α = 0) of the spectrum more severely than the middle point of the spectrum (α = 0.5).

Table 4: Averaged error rates of VR-Trees ($\alpha = 0$, $\alpha = 0.5$), Coalescence and Random Forests on Concepts **A** and **B**. The default number of training samples is $k = 1024$, unless otherwise specified. The best error rate in each row is bold faced.

	VR-Trees ($\alpha = 0$)	Coalescence	VR-Trees ($\alpha = 0.5$)	Random Forests
A	**1.4%**	1.6%	2.0%	1.9%
A, 8 irr. att.	7.6%	3.4%	**3.0%**	3.2%
A, 40% class noise	30%	21.9%	**14.7%**	33.4%
A, $k = 64$	**6.7%**	8.9%	10.4%	7.3%
B	0.3%	**0%**	**0%**	**0%**
B, 8 irr. att.	5.2%	**0%**	**0%**	**0%**
B, 40% class noise	30.7%	4.2%	**2.6%**	32.5%
B, $k = 64$	2.4%	1.1%	1.1%	**0.8%**

- Between Coalescence and Random Forests, Coalescence performs better under class noise condition; but Random Forests is a better performer under small training size in both concepts **A** and **B**.

The above empirical results show that, while complete-random trees, i.e. VR-Trees (α = 0) are good at approximating non-axis-parallel boundary, they are easily over-fitted to irrelevant attributes, class noise and small training sample. Although it is the case, we find that Coalescence is a way to manage these propensities without a search for a specific α value. In Table 4, we find that Coalescence tends to have performance closer to the better performing learner of either VR-Trees (α = 0) or VR-Trees (α = 0.5); the only exception is when learning the non-axis-parallel concept **A** with a small training sample.

As to further our understanding on how irrelevant attributes affect Coalescence, we perform a simple check on the eighteen data sets in which Random Forests performs better (ignoring two artificial data sets: *led* and *waveform* in which we have already known the results with and without irrelevant attributes).

We remove the less-important half of all attributes according to Random Forest's variable-importance (Breiman, 2001). We then evaluate these eighteen data sets again using Coalescence and Random Forests under the same 10-fold cross validation. The result in Table 5 shows that Coalescence performs better in ten out of the eighteen data sets; and Random Forests performs better only in four data sets. This result indicates the influence of irrelevant attributes on Coalescence is greater than that on Random Forests. Among the eighteen data sets, Coalescence's error rates are reduced by more than half in *labor*, *hayes* and *tic-tac-toe* data sets. The result indicates that further management of irrelevant attributes can indeed improve the performance of Coalescence.

6. Related Work

In this section, we first highlight the differences between VR-Trees and Random Forests in their capability to vary the degree of randomness, so to distinguish VR-Trees from others. Then, in order to better position VR-Trees, we discuss various different decision tree ensembles that are related to VR-Trees.

Table 5: Evaluation of Coalescence and Random Forests with respect to the condition of irrelevant attribute: eighteen data sets are listed with the less-important half of all attributes removed. Boldfaced indicates improvement in error rate using the reduced number of attributes

	Coalescence		Random Forests	
data sets	Full att.	Half att.	Full att.	Half att.
autos	18.1	**13.6**	15.2	**14.6**
cleveland	42.9	45.2	41.9	43.9
dna	5.3	**3.9**	3.4	**3.2**
echo	33.5	39.0	32.0	42.8
flare	18.6	19.0	18.5	18.9
hayes	40.6	**18.1**	41.3	**16.9**
hepatitis	18.0	**16.7**	17.3	17.4
iris	4.7	**4.0**	3.3	3.3
labor	7.0	**3.3**	5.0	5.0
liver	27.0	33.6	26.1	33.1
pima	23.4	24.5	23.2	24.5
post	40.0	**37.8**	32.2	36.7
satimage	8.8	9.4	8.1	9.3
solar	29.4	30.3	27.2	28.7
sonar	15.9	**13.0**	13.9	16.4
tic-tac-toe	18.8	**3.0**	18.3	**1.3**
vote	4.1	4.4	4.1	4.4
wine	3.4	**1.7**	2.3	2.8

6.1 Relationship with Random Forests

It is interesting to note that Breiman (2001) found that Random Forests' accuracy is not overly sensitive to the value of F, which is the parameter intended to vary the degree of randomness. The F parameter corresponds to the α parameter in VR-Tree. Yet, our experiments in section 5.1 clearly shows that varying the degree of randomness (using α) has a significant impact on the predictive performance of the resulting ensembles. It is thus important to identify the differences between the two ensembles that cause the different behaviours. We will do so in the following paragraphs.

Algorithm 6: The random feature selection framework of Random Forests

Input: D_t - Training set, F - number of features
Output: $node$: tree node
randomly select F features from all available features
D' = a subset of D_t according to the F features
$node$ = DeterministicTestSelection(D')
return $node$

On the surface, Random Forests is very similar to Agg.VR-Trees because both of them use the deterministic and random test-selections in the tree induction process. However, they differ in the way in which the test-selection is applied in each decision node. The randomisation framework of Random Forests is described in Algorithm 6.

In applying test-selection, Random Forests applies both random feature selection and deterministic test-selection in each node; however VR-Tree only applies either random or deterministic test-selection in each node. α controls the probability of whether deterministic or random test-selection is applied in each node; whereas the mixed application of the two selection processes in each node constrains the 'amount' of randomness that can be introduced to Random Forests. In case of Random Forests, F only controls the number of features to be randomly selected. Once selected, the deterministic test-selection chooses the best feature. Thus, if the 'best' feature is readily selected in the first place, then no matter what the F is, the 'best' feature will always be chosen by the deterministic test-selection. This agrees with Breiman's observation that the error rate is not overly sensitive to the different values of F in Random Forests.

The accessibility to the different degrees of randomness directly affects the diversity of models that are produced, Figure 9 shows the strength-correlation curves for Random Forests (using all the available F values, 19 for *segment* data and 21 for *waveform21* data), in comparison with Agg.VR-Trees using eleven α values sampled with an equal interval. We find that Random Forests produces ensembles which are highly correlated to each other and many of which have similar strength. The same result is also reported by Breiman (2001). Note that the fitted curves for Random Forests are visual aids which do not mean to represent accessibility between points. In contrast, Agg.VR-Trees produces ensembles which are accessible along the curve and spread along a wider range.

In a nutshell, the randomisation framework used in Random Forests significantly limits its ability to scale to the different levels of randomness when the total number of features is small. On the other hand, VR-Trees is able to scale to different levels of randomness regardless of the number of features.

6.2 Other Related Work

An approach to search for the best α value is proposed (Liu & Ting, 2006). This approach searches the optimal α value based on average progressive training errors. An estimated optimal $\widehat{\alpha}$ for a task is generated as follows:

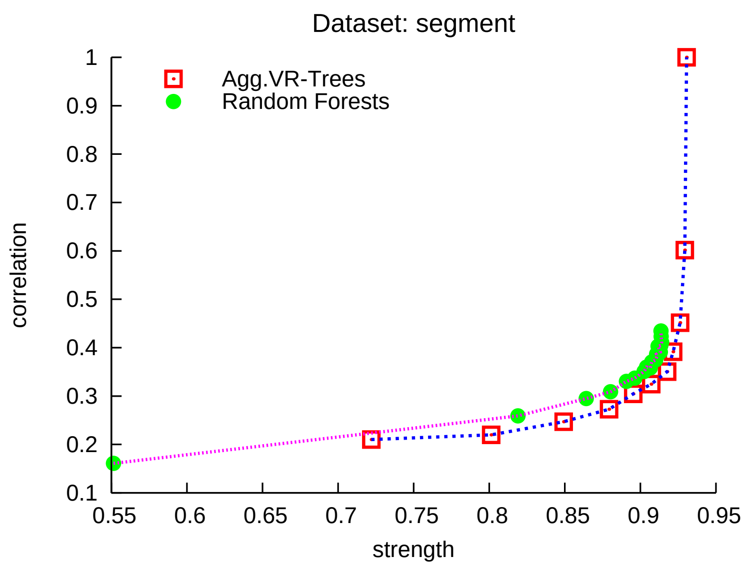
$$\widehat{\alpha} = \underset{0 \leq \alpha \leq 0.5}{\arg\min} [\frac{1}{N} \sum_{i=1}^{N} err(\alpha, i, D_t)] \tag{2}$$

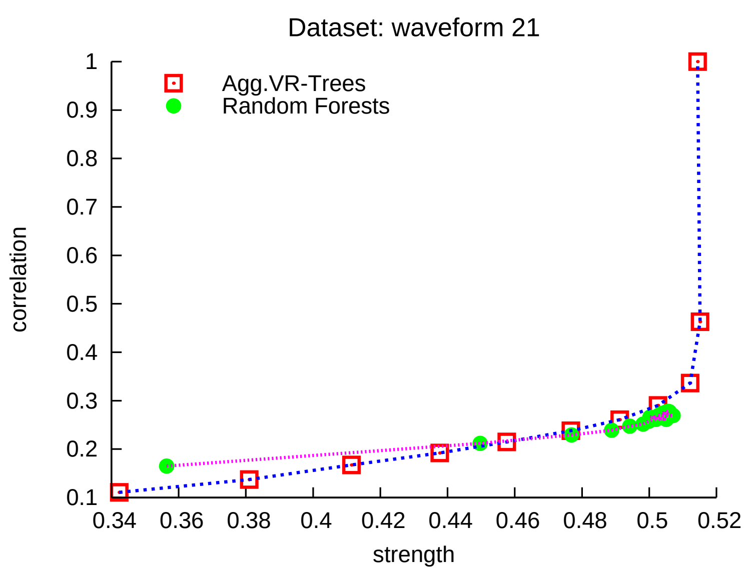
where N is the total number of trees in an ensemble, $err()$ returns the training error rate of an ensemble of size i, while Agg.VR-Trees is set at α with training set D_t. After obtaining $\widehat{\alpha}$, i.e., the best performing α, the ensemble employs the model with $\widehat{\alpha}$ for actual predictive tasks. Note that each unpruned tree in the ensemble stops growing if the number of training examples in the node is four or less (the default setting as in C4.5.) This avoids generating zero training error trees. Though the method has comparable prediction performance to that of the Coalescence approach, it requires a substantial computational overhead where the ensembles of all α values in the search must be produced.

In contrary to common belief, Fan et al (2003) first propose the use of complete-random trees to produce accurate predictive models. Fan (2004) explains the reason why the combination of complete-random trees and probability averaging produces accurate models. Using complete-random trees, Liu et al. (2005) show that their ensembles perform comparably to Bagging and Random Forests.

Cutler and Zhao (2001) propose PERT (Perfect Random Tree Ensembles) which randomises the test-selection for continuous-valued features to achieve higher randomisation. At each potential

Figure 9: Strength-Correlation plots for Random Forests and Aggregating VR-Trees at different F and α values. Aggregating VR-Trees has a wider range of correlation as compared to Random Forests.





split, PERT first randomly selects two examples of different classes from the local training set. A feature is then selected randomly. A cut point on the feature is randomly selected between the values of that two samples. A leaf is formed if two examples of different classes cannot be found after ten trials. PERT is also shown to be competitive to Bagging and Random Forests. We believe that this method is likely to be close to the complete-random-end of the spectrum. However, it is unclear on how the different degrees of randomisation can be introduced to the PERT framework.

Extra-Trees (Geurts, Ernst, & Wehenkel, 2006) relies on the same framework as in Algorithm 6. However, a random split-selection is used instead of the deterministic split-selection. Different from PERT, Extra-Trees' cut points are randomly selected between the maximum and the minimum values of the given samples. As compared to PERT, Extra-Trees requires an additional data-scan at every node of a tree to find the maximum and minimum values, which can be a disadvantage in terms of computational complexity. For categorical features, a random subset split is used in Extra-Trees. This method is also shown to be competitive to Random Forests.

Robnik-Šikonja (2004) reports an improved version of Random Forests by using five different deterministic test-selection criteria instead of one. This achieves the effect of increased diversity by producing different variants of deterministic models — the same starting point as Random Forests, Bagging and Random Subspace.

MultiBoosting (Webb, 2000; Webb & Zheng, 2004) is another approach that combines more than one type of ensembles. MultiBoosting is a tightly-coupled method that incorporates bagging (random sample weight assignment) into the main Boosting procedure (the incremental sample weight modifier) in order to increase model diversity. Like Bagging and Random Forests, it is focusing on increasing diversity at the deterministic-end of the spectrum.

There are many algorithms reported in the literature, other than those listed in this paper, suggesting different ways to combine models generated from one algorithm or different algorithms, (e.g., see Breiman, 1996b; Ting & Witten, 1999; Perrone & Cooper, 1993) . All of these require some kind of learning or estimation in order to either selectively choose some available models or build a meta-model to combine them. The Coalescence approach is more simple than these approaches because it does not require to learn a meta-model or/and some kind of estimation.

7. Conclusions

In this paper, we make the following contributions:

- Propose a new algorithm, which generates a spectrum of VR-Trees that span between complete-random trees and deterministic trees. We show that different points in the spectrum can be significantly different in terms of predictive accuracy. This opens up new opportunities for decision tree ensembles.

- Show that existing ensemble methods such as Bagging, Random Subspace, and Max-Diverse Ensemble are only at either end of the spectrum. The performance of these ensembles can be improved by moving towards the middle of the spectrum and the improvements can be significant.

- Discover that the two halves of the spectrum have their distinctive characteristics, separated by a critical point of α=0.5. This point has two interesting characteristics. First, it produces an equal percentage of random and deterministic decision nodes in VR-Trees. Second, it often lies on the lowest generalisation error region (or close to it) in a typical strength-correlation curve. Ensembles generated from $\alpha \in [0.0, 0.5]$ often out-perform those generated from $\alpha \in [0.5, 1.0]$.

- Propose a new approach in building better performing ensembles. The Coalescence approach coalesces a number of points in the first-half of the spectrum. We show that it ranks better than any single operating point in the whole spectrum.

- Identify the key differences between ensembles constructed under the frameworks of Random Forests and VR-Tree and explain why the predictive accuracy is sensitive to a parameter in the VR-Tree framework, but not in the Random Forests framework.

In our empirical evaluation, Coalescence is compared with five benchmarking ensemble methods: Max-Diverse Ensemble, Bagging, Random Forests, Random Subspace and C5 Boosting. The study reveals that Coalescence ranks top and it is the only ensemble method that is significantly better than Bagging and Max-Diverse Ensemble using the Friedman test.

Although Coalescence is not significantly better than Random Forests, we have identified that: while Random Forests performs better than Coalescence under the conditions of irrelevant attribute and small training size, Coalescence performs better in learning non-axis-parallel concepts and under class noise.

Acknowledgments

Y. Yu and Z.-H. Zhou were supported by the National Science Foundation of China under Grant Nos. 60635030 and 60721002. Part of the research was conducted when K. M. Ting was visiting the LAMDA group, Nanjing University.

Appendix A. Data characteristics of all data sets

Table 6: Data characteristics of all forty-five data sets used in the experiments. Data are taken from the UCI repository (Asuncion & Newman, 2007).

data sets	*size*	*#att.*	*#class*	*description*
abalone	4177	1n, 7c	2	Abalone growth
anneal	898	13n, 6c, 19b	6	Steel annealing
audiology	226	8n, 61b	23	Standardised Audiology Database
auto	205	6n, 15c, 4b	7	1985 Auto Imports Database
balance	625	4c	3	Balance Scale weight and Distance Database
breast-w	699	10c	2	Winconsin breast cancer database
breast-y	286	6n, 3b	2	Ljubljana Breast cancer database
chess	3196	35n, 1b	2	Chess end games
cleveland	303	4n, 6c, 3b	5	Cleveland heart disease database
coding	20000	15n	2	Coding database
credit-a	690	4n, 6c, 4b	2	Australian Credit database
credit-g	1000	12c, 12b	2	German credit database
dna	3186	60n	3	Primate splice-junction gene sequences
echo	133	6c, 1b	2	Echocardiogram data
flare	1066	3n, 2c, 5b	2	Predicting solar flare
glass	214	9c	7	Glass identification database
hayes	160	4c	3	Hayes-Roth & Hayes-Roth database
hepatitis	155	6c, 13b	2	Hepatitis Domain
horse	368	13n, 7c, 2b	2	Horse colic database
hypo	3163	7c, 18b	2	Thyroid disease database
ionosphere	351	34c	2	Radar returns from the ionosphere
iris	150	4c	3	Iris plants database
labor	57	5n, 8c, 3b	2	Final settlements in labour negotiations
led24	3200	24b	10	LED display + 17 irrelevant attributes
led7	3200	7b	10	LED display with no irrelevant attribute
liver	345	6c	2	BUPA liver disorder
lymph	148	6n, 3c, 9b	4	Lymphography domain
nursery	12960	8n	5	Nursery database
pima	768	8c	2	Diabetes of female Pima Indians
post	90	7n, 1c	3	Postoperative patient data
primary	339	3n, 14b	22	Primary tumor domain
satimage	6435	36c	7	Satellite image data set from NASA
segment	2310	19c	7	Image segmentation data
sick	3163	7c, 18b	2	Sick-euthyroid data
solar	323	3n, 3c, 6b	6	Solar data set
sonar	208	60c	2	Classification of sonar signals
soybean	683	19n, 16b	19	Soy bean disease diagnosis
threeOf9	512	9b	2	The concept of three of nine
tic-tac-toe	958	9n	2	Tic-Tac-Toe board configurations
vehicle	846	18c	4	Vehicle silhouette data set
vote	435	16n	2	Votes for U.S. Congressmen
waveform21	5000	21c	3	Waveform data
waveform40	5000	40c	3	Waveform data with 19 noise attributes
wine	178	13c	3	Wine recognition data
zoo	101	1n, 15b	7	Zoo database

Attribute type is indicated by **n**: nominal, **c**: continuous, and **b**: binary.

Appendix B. Theoretical Analysis of VR-Trees ($\alpha = 0$)

The notations below are similar to those of Bousquet and Elisseeff (2002). Denote $\mathcal{X}$ as the input space and $\mathcal{Y}$ as the output space. A learning algorithm is a function $T : \mathcal{Z}^k \rightarrow \mathcal{F}$, where $\mathcal{F} \in \mathcal{Y}^{\mathcal{X}}$ is a function space. We denote f as an function in $\mathcal{F}$. The learning algorithm outputs a function that approximates the underlying true function f^* by inspecting the training set. A training set $S = \{z_i\}_{i=1}^k$, where $z_i = (x_i, y_i)$ and $y_i = f^*(x_i)$, contains k i.i.d. examples in $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ drawn from an unknown distribution $\mathcal{D}$. We consider the case that $\mathcal{X}$ is a bounded $\mathbb{R}$eal space and $\mathcal{Y} = \{-1, +1\}$. Denote the training set after *removing* the i-th example as

$$S^{\setminus i} = \{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_k\}$$

Given a learning algorithm T trained on S, we denote its *generalisation error* as

$$R(T, S) = \mathbb{E}_{z=(x,y)}[\ell(f_{T,S}(x), y)] \ ,$$

and its *empirical error* as

$$R_e(T, S) = \frac{1}{k} \sum\nolimits_{i=1}^{k} \ell(f_{T,S}(x_i), y_i) \ ,$$

where $f_{T,S} = T(S)$ and $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a loss function.

We study the generalisation ability from the viewpoint of the *Pointwise Hypothesis Stability* (Bousquet & Elisseeff, 2002).

Definition 1 (Pointwise Hypothesis Stability) *An algorithm T has pointwise hypothesis stability β with respect to the loss function ℓ if it holds for all $i \in \{1, \ldots, k\}$ that*

$$\mathbb{E}_{S\sim\mathcal{D}}[|\ell(f_{T,S}(x_i), y_i) - \ell(f_{T,S^{\setminus i}}(x_i), y_i)|] \leq \beta$$

Theorem 11 of Bousquet and Elisseeff (2002) reveals the relationship between the *Pointwise Hypothesis Stability* and the *generalisation error*. To be self-contained, we write the theorem as Lemma 1.

Lemma 1 *For any learning algorithm T with pointwise hypothesis stability β with respect to a loss function ℓ such that for some M, $0 \leq \ell(\cdot, \cdot) \leq M$, we have with probability $1 - \delta$,*

$$R(T, S) \leq R_e(T, S) + \sqrt{\frac{M^2 + 12Mk\beta}{2k\delta}}$$

Assume that *(i)* there is only one attribute in $\mathcal{X}$ and the attribute has $\mathbb{R}$eal values, *(ii)* every training example has unique attribute values, *(iii)* each internal node has non-empty subsets, *(iv)* the tree building process stops at nodes which contain only one example, and *(v)* the output of a VR-Tree is $+1$ or -1 for an input instance in the case of binary classification. When building an ensemble of VR-Trees, we run the VR-Tree algorithm N times to produce a set of trees $\{f_i\}_{i=1}^N$. Given a test instance $z = (x, y)$, the output of the ensemble is:

$$f^N(x) = \frac{1}{N} \sum\nolimits_{i=1}^{N} f_i(x)$$

whose range is $[-1, +1]$. In the following analysis, we let N approach *infinity*.

In order to study the Pointwise Hypothesis Stability of VR-Trees (α = 0) ensembles, we need to bound

$$\mathbb{E}_{S\sim\mathcal{D}}[|\ell(f_{T,S}(x_i), y_i) - \ell(f_{T,S^{\setminus i}}(x_i), y_i)|].$$

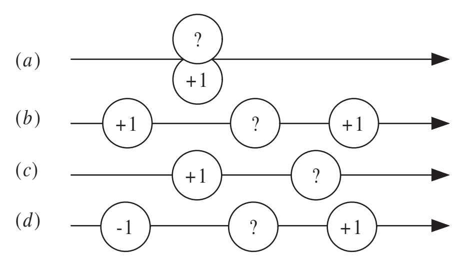
We specify the loss function as

$$\ell(y_1, y_2) = \begin{cases} 1, & |y_1 - y_2| > 1 \\ 0.5, & |y_1 - y_2| = 1 \\ 0, & |y_1 - y_2| < 1 \end{cases}.$$

Let μ_S denote the number of *class transitions*, which are example pairs (z_j, z_k) in S such that $y_j \neq y_k$ and there is no example between x_j and x_k. μ_S will be used to bound β later.

Now, when a training instance $z' = (x', y') \in S$ is being held out as the test example and leaving $S^{\setminus i}$, we want to know how a VR-Tree ensemble makes its predictions on z'. We find that one of the following four events will happen when classifying z', as illustrated in Figure 10.

Figure 10: Illustration of four possible places of a test instance. Circles of $+1$, -1 and ? denote positive, negative and test instances, respectively.



a) z' is a duplicate of a training example $z_i = (x_i, y_i)$, as illustrated in Figure 10(a). We have $f^n(x') = y_i$, because every tree leaf is pure, which means the empirical error is always zero. Since we assume that every instance has a unique location, we ignore this case for μ_S.
b) z' is located between two training examples $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$ with $y_i = y_j$, as illustrated in Figure 10(b). In this case, z' will fall into a leaf node containing z_i or z_j. Therefore, we have $f^n(x') = y_i = y_j$. When z' is wrongly classified, that is, z' has a different label from y_i and y_j, two counts are added to μ_S.
c) z' is located *outside* the training set and $z_i = (x_i, y_i)$ is the nearest training example, as illustrated in Figure 10(c). In this case z' will fall into a leaf node that must contain z_i, thus $f^n(x') = y_i$. When z' is wrongly classified, one count is added to μ_S.
d) z' is located between two training examples $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$ with $y_i \neq y_j$. When z' is wrongly classified, that is, z' has a different label from either y_i or y_j, one count is added to μ_S.

Therefore, given an example $z_i = (x_i, y_i) \in S$, we have $\ell(f_{T,S}(x_i), y_i) = 0$, because y_i is included in S, which is the case of Figure 10(*a*). By the above analysis, we also have the number of errors upper-bounded by μ_S. Then we have

$$\begin{aligned} \beta &= \mathbb{E}_{S\sim\mathcal{D}}[|\ell(f_{T,S}(x_i), y_i) - \ell(f_{T,S^{\setminus i}}(x_i), y_i)|] \\ &= \mathbb{E}_{S\sim\mathcal{D}}[\ell(f_{T,S^{\setminus i}}(x_i), y_i)] \\ &\leq \mathbb{E}_{S\sim\mathcal{D}}[\mu_S]/k \ . \end{aligned}$$

Note that μ_S varies on training sets, but it is upper bounded by the class transitions of the underlying true function f^*, that is, let μ^* be

$$\mu^* = \left|\{x \in \mathcal{X} | \forall \sigma > 0 \ \exists x' : f^*(x) f^*(x') < 0 \wedge \|x - x'\| < \sigma\}\right| \ ,$$

which is a constant. We then have

$$\forall S \sim \mathcal{D} : \mu_S \leq \mu^*,$$

and $\mu_S = \mu^*$ when S is large enough. Therefore, we have

$$\beta \leq \mu^*/k \ .$$

Since μ^* is a constant, the ensemble of VR-Trees (α = 0) is a stable learner, whose generalisation error is bounded, with probability $1 - \delta$,

$$R(T, S) \leq \sqrt{\frac{1 + 12\mu^*}{2k\delta}} \ .$$

It can be observed that features with large μ^* weaken the predictive performance.

Moreover, we find that irrelevant attributes, class noise and insufficient training sample can cause a large μ^*. For irrelevant attributes, they perform an almost random projection from $\mathcal{X}$ to $\mathcal{Y}$. If f^* is a random projection, i.e. $P(f^*(x) = +1) = 0.5$, the probability that there are μ_S class transitions in training set S is

$$P(\mu_S) \leq (1/2)^{|S|/\mu_S - 1}$$

by considering that the μ_S transitions divide S into equal segments. This implies that irrelevant attributes weaken the predictive performance of VR-Trees (α = 0). For class noise and insufficiency of training samples, it is not hard to see that they also increase μ^* and hence also weaken the performance of VR-Trees (α = 0).

Appendix C. Estimation of Strength and Correlation

To make this paper self contained, we provide the estimation of strength and correlation which is defined by Breiman (2001) and corrected by Kobayashi (2002). Given an ensemble of N trees $\{f_1, ..., f_N\}$, a hold-out set $D_h = \{(x_1, y_1), ..., (x_k, y_k)\}$ and k is the number of test cases, we can estimate the followings:

- Strength - it is estimated by the average of margin over D_h.
- Correlation - it is estimated by taking the variance of margin over the standard deviation of random vectors, which represent trees.

C.1 Estimation of Strength $\hat{s}$

The estimation of Strength is given by:

$$\hat{s} = \frac{1}{k} \sum_{i=1}^{k} s_i \tag{3}$$

The margin s_i is given by:

$$s_i = \frac{1}{N} \sum_{j=1}^{N} I(f_j(x_i) = y_i) - \frac{1}{N} \sum_{j=1}^{N} I(f_j(x_i) = c(i)) \tag{4}$$

For each test case i, $c(i)$ is the class label that receives the maximum votes among N trees and $c(i)$ can be any class label other than the true label y_i. $I(.)$ is the indicator function that returns 1 when it is true, 0 otherwise.

C.2 Estimation of Correlation $\hat{\bar{\rho}}$

The estimation of Correlation is given by:

$$\hat{\bar{\rho}} = \frac{var(s)}{\frac{1}{N} \sum_{j=1}^{N} sd(j)} \tag{5}$$

The variance of margin $var(s)$ is given by:

$$var(s) = \frac{1}{k} \sum_{i=1}^{k} s_i^2 - \hat{s}^2 \tag{6}$$

The standard deviation of each random vector $sd(j)$ is given by:

$$sd(j) = [p_1 + p_2 - (p_1 - p_2)^2]^{1/2} \tag{7}$$

$$p_1 = \frac{1}{k} \sum_{i=1}^{k} I(f_j(x_i) = y_i) \tag{8}$$

$$p_2 = \frac{1}{k} \sum_{i=1}^{k} I(f_j(x_i) = c(i)) \tag{9}$$

References

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository.. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*(1-2), 105–139.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, *2*, 499–526.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (1996b). Stacked regressions. *Machine Learning*, *24*(1), 49–64.

Breiman, L. (2000). Randomizing outputs to increase prediction accuracy. *Machine Learning*, *40*(3), 229–242.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Buttrey, S. E., & Kobayashi, I. (2003). On strength and correlation in random forests. In *Proceedings of the 2003 Joint Statistical Meetings, Section on Statistical Computing*, San Francisco, CA. American Statistical Association.

Cutler, A., & Zhao, G. (2001). PERT - perfect random tree ensembles. In *Computing Science and Statistics*, Vol. 33, pp. 490–497, Costa Mesa, Orange Country, California.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal Machine Learning Research*, *7*, 1–30.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139–157.

Fan, W. (2004). On the optimality of probability estimation by random decision trees. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence (AAAI)*, pp. 336–341, California, USA. AAAI Press / The MIT Press.

Fan, W., Wang, H., Yu, P. S., & Ma, S. (2003). Is random model better? on its accuracy and efficiency. *ICDM '03: Proceedings of the Third IEEE International Conferenceon Data Mining*, 51–58.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning : Data mining, Inference, and Prediction*. Springer-Verlag.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Kobayashi, I. (2002). *Randomized Ensemble Methods for Classification Trees*. Ph.D. thesis, Naval Postgraduate School, Monterey, CA.

Liu, F. T., & Ting, K. M. (2006). Variable randomness in decision tree ensembles. In *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference (PAKDD 2006)*, pp. 81–90, Singapore.

Liu, F. T., Ting, K. M., & Fan, W. (2005). Maximizing tree diversity by building complete-random decision trees. In *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference (PAKDD 2005)*, pp. 605–610, Hanoi, Vietnam.

Melville, P., & Mooney, R. (2003). Constructing diverse classifier ensembles using artificial training examples. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 505–510, Mexico.

Perrone, M. P., & Cooper, L. N. (1993). When networks disagree: ensemble methods for hybrid neural networks. *Artificial Neural Networks for Speech and Vision*, 126–142.

Quinlan, R. J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, Calif.

Robnik-Šikonja, M. (2004). Improving random forests.. In Boulicaut, J.-F., Esposito, F., Giannotti, F., & Pedreschi, D. (Eds.), *Proceedings of The 15th European Conference on Machine Learning (ECML 2004)*, Vol. 3201 of *Lecture Notes in Computer Science*, pp. 359–370, Pisa, Italy. Springer.

Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artifical Intelligence Research (JAIR)*, *10*, 271–289.

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machince Learning*, *40*(2), 159–196.

Webb, G. I., & Zheng, Z. (2004). Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, *16*(8), 980–991.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 609–616.