Regularizing axis-aligned ensembles via data rotations that favor simpler learners

Rico Blaser Piotr Fryzlewicz

November 10, 2020

Abstract

To overcome the inherent limitations of axis-aligned base learners in ensemble learning, several methods of rotating the feature space have been discussed in the literature. In particular, smoother decision boundaries can often be obtained from axis-aligned ensembles by rotating the feature space. In the present paper, we introduce a low-cost regularization technique that favors rotations which produce compact base learners. The restated problem adds a shrinkage term to the loss function that explicitly accounts for the complexity of the base learners. For example, for tree-based ensembles, we apply a penalty based on the median number of nodes and the median depth of the trees in the forest. Rather than jointly minimizing prediction error and model complexity, which is computationally infeasible, we first generate a prioritized weighting of the available feature rotations that promotes lower model complexity and subsequently minimize prediction errors on each of the selected rotations. We show that the resulting ensembles tend to be significantly more dense, faster to evaluate, and competitive at generalizing in out-of-sample predictions.

Keywords: random rotation, regularization, ensemble learning, minimal complexity

1 Introduction

Feature rotations are ubiquitous in modern machine learning algorithms – from structured rotations, such as PCA, to random rotations and projections. For example, in computer vision, local image rotations are routinely used to obtain high-quality rotation-invariant features (e.g. Takacs et al., 2013). In the context of axis-aligned ensemble learning, rotations – and random projections, which can be decomposed into a random rotation and an axis-aligned projection – can make the difference between a highly successful classifier and an average classifier (e.g. Durrant and Kaban, 2013).

Rodriguez et al. (2006) introduced rotation forests after demonstrating that repeated PCA-rotations of random subsets of the feature space significantly improved classification performance of random forests (Breiman, 1999) and other tree ensembles. Blaser and Fryzlewicz (2016) showed that rotation forests can be outperformed using unstructured random rotations of the feature space prior to inducing the base learners. While random rotations are used with classifiers designed for high-dimensional settings, Cannings and Samworth (2017) presented a random projection ensemble, in which the high-dimensional feature space is first projected into a lower-dimensional space before applying a classifier designed for low-dimensional settings.

An important insight from the latter two papers is that the vast majority of rotations are unhelpful in improving out-of-sample classifier performance. Instead, most of the benefit of these ensembles is derived from a small number of rotations that are particularly well-suited for the specific classification problem.

In the present paper we investigate the efficacy of rotations more closely and attempt to answer the question of how we can identify or construct rotations that explicitly improve classifier performance. We hypothesise that the most beneficial rotations are those that align significant segments of the decision boundary with one of the axes and thus result in simpler and more compact base learners: we call it *rotation to simplicity*. We also believe the converse to be true: those rotations that produce less complex base learners positively impact ensemble performance. Supporting evidence for this assertion is provided in Section 5.

The remainder of the paper is organized as follows: in Section 2, we introduce the basic ensemble notation, as well as an extended loss function which takes into consideration the complexity of the base learners. This is similar to loss functions in linear regression that include penalties on the regression coefficients. In Section 3, we introduce a low-cost regularization technique, which explicitly favors rotations that are expected to produce simple base learners. Section 4 takes a step back and illustrates why certain rotations are better than others for axis-aligned learners and how these rotations differ from analytic methods, such as PCA. Next, we present performance results on a sample of well-known UCI data sets in Section 5 and conclude with our final thoughts.

2 Motivation

A decision tree divides the predictor space into disjoint regions G_j, where $1 \leq j \leq J$, with J denoting the total number of leaf nodes of the tree. Borrowing the notation from Hastie et al. (2009), the binary decision tree is represented as

$$T(x;\Omega) = \sum_{j=1}^{J} c_j I(x \in G_j), \tag{1}$$

where $\Omega = \{G_j, c_j\}_1^J$ are the optimisation- or tuning-parameters and $I(\cdot)$ is an indicator function. Inputs x are mapped to a constant c_j, depending on which region G_j they are assigned to. A tree ensemble consisting of M trees can then be written as

$$E_M(x) = \sum_{m=1}^{M} T(x; \Omega_m). \tag{2}$$

In this paper, we assume that trees are grown independently and that no co-dependence exists between the tuning parameters of different trees. This restriction implicitly excludes boosted tree ensembles (Friedman, 2001). Our goal is then to optimise the tuning parameters Ω_m for each tree in such a way as to minimise a given loss function, $L(y_i, f(x_i))$, that is

$$\hat{\Omega}_m = \arg\min_{\Omega_m} \sum_{i=1}^{N} L(y_i, T(x_i;\Omega_m)). \tag{3}$$

It should be noted that the general tree-induction optimization problem in equation (3) is NP-complete (Hyafil and Rivest, 1976) even for two-class problems in low dimensions (Goodrich et al., 1995) and an axis-aligned, greedy tree induction algorithm such as CART (Breiman et al., 1984) is typically used to find a reasonable approximation.

At this point we depart from the standard tree ensemble setting in two aspects: (1) we add a penalty P to the loss function and (2) we add rotations R_k to the input data. Hence, the loss function gets modified to

$$L(y_i, f(x_i)) = \underbrace{V(y_i, f(R_k(x_i)))}_{accuracy} + \underbrace{P(R_k(x_i))}_{complexity}, \tag{4}$$

where the regularisation term $P(\cdot)$ penalises rotations that lead to more complex base learners. $V(y_i, f(x_i))$ is a typical loss function – such as square-, hinge-, or logistic loss – which does not take model complexity into account (see e.g. James et al., 2013). Minimizing this combined loss function resembles constrained regression problems, such as Ridge- or Lasso-regressions (Tibshirani, 1996), but instead of constraining coefficients, we actively regularize the base learners. Lastly, the subscript k denotes the specific rotation; we typically grow multiple trees per rotation, depending on the efficacy of the rotation: this is described in detail in Section 3.

With the addition of the regularisation term, we have made the problem even more challenging to solve. Since tree induction was already NP-complete to begin with, we discuss an algorithm in the following section which strictly separates the weighting of favorable rotations that reduce model complexity from the tree induction optimization, which improves accuracy. Using this approach, we implicitly assume that simpler models do not lead to lower prediction accuracy, a hypothesis we show to be empirically valid in Section 5.

3 Regularization

In this section, we introduce our proposed algorithm for generating an ensemble that optimizes the use of available rotations.

Given a set of R feature rotations, we would like to build an ensemble consisting of M base learners. In order to accomplish this, the algorithm first builds tiny micro-forests of U unconstrained trees on each rotation, a low-cost operation because $U \ll M$ and $U < M/R$. Based on the statistical properties of these micro-forests, the full ensemble is constructed. Here we present the generic algorithm; in Section 5 we demonstrate several ways of leveraging the available statistics. For tree-based ensembles, the trees in the micro-forests can frequently be reused for the full ensemble, further reducing the amortized cost of building the micro-forests.

Algorithm 1 describes the regularized rotation procedure in detail. The integer inputs denote the desired total number of trees M in the complete ensemble, the number of available (or generated) rotations R, and the number of trees U created for each micro-forest.

In line 2 of Algorithm 1, the available rotations are stored in an array named *rotations*. It is important to include the identity rotation here to make sure the procedure returns high-quality results when the problem is already optimally rotated to begin with. If too few rotations are available, the procedure can generate random rotations in addition to the identity rotation (Anderson et al., 1987).

In lines 4-5, an unconstrained, unpruned micro-forest consisting of U trees is grown. The recommended default value of U is of the order of 10-20 trees. The purpose of these trees is merely to obtain a reliable estimate of the median complexity of a representative tree that will be grown on the particular rotation, with minimal interference from outliers.

Our main proposal in this paper is to apply a complexity measure for base learners and use it to rank the obtained rotations from the best one which corresponds to the least complex learners to the worst one that corresponds to the most complex learners. In the case of tree ensembles, we suggest a complexity measure $C(\cdot)$ whereby trees with a smaller number of nodes (size) are considered less complex and, among trees with the same number of nodes, more shallow trees (depth) are considered less complex, that is

$$C(T(x;\Omega_m)) = \#nodes + depth/N, \tag{5}$$

where $\#nodes = \ 2J-1$ and $depth \le J$ for binary decision trees, both depending on Ω_m. N is the number of data points and J the number of leaf nodes in the tree. It is clear that $1 \ge depth/N$ and, consequently, that *depth* merely acts as a tie-breaker for trees of equal size. We further discuss tree complexity in Section 4.1. Up to this step, only model complexity was used to quantify rotations; this corresponds to the right-most section of formula (4).

The sorting procedure in line 7 of Algorithm 1 arranges the rotations into ascending order of complexity C. At this point, there are several ways of using this information. In section 3.1 we apply a parametric, non-increasing family of curves with a tuning parameter h and use the out-of-bag (OOB) errors of the micro-forests to determine the optimal parameter in a grid search. However, as we will show in Section 5, it is also possible to use the ranking on its own, without combining it with predictive performance. The key point here is that whatever procedure we use, it will determine the number of base learners that need to be created for each rotation. This is accomplished in line 9 of Algorithm 1.

Should additional trees (beyond the U already available trees on each rotation) be needed, these are generated and added to the rotation in line 10. Typically, these need to be added to the most favorable rotations.

Finally, the equal-weighted ensemble is constructed from the trees on the different rotations. It is important to note that while the individual trees are equal-weighted in the ensemble, more trees are used from favorable rotations and hence the rotations are not equal-weighted. Also note that $\sum_{i=1}^{R}$ rotations$[i]$.numtrees $= M$.

3.1 Weighting of rotations

Given an *ordered* sequence of R rotations ($r = 1$ for the most favorable rotation and $r = R$ for the least attractive rotation) and a specified total number of base learners M in the ensemble, we need to determine how many base learners to train on each rotation. This corresponds to line 9 in Algorithm 1 above. We now discuss the details of this procedure.

Any sensible (percentage) weighting scheme will have the following three properties:

1. $w(r) \ge 0, \forall r$
2. $w(r) \ge w(r+1)$

Algorithm 1 Regularized Rotation Ensemble (Pseudocode)

```
procedure REG_ROT($M, R, U$)    ▷ $M$: ensemble size, $R$: #rotations, $U$: trees per $\mu$-forest
    rotations ← obtain_rotations($R$)
    for $i \leftarrow 1$ to $R$ do
        rotations[$i$].forest ← create_unconstrained_$\mu$-forest($U$)
        rotations[$i$].complexity ← compute_complexity(rotations[$i$].forest)
    end for
    sort(rotations, complexity)
    for $i \leftarrow 1$ to $R$ do
        rotations[$i$].numtrees ← compute_numtrees($M$, rotations)
        rotations[$i$].forest ← add_trees(rotations[$i$].forest, max(rotations[$i$].numtrees - $U$, 0))
    end for
    for $i \leftarrow 1$ to $R$ do
        for $j \leftarrow 1$ to rotations[$i$].numtrees do
            ensemble.forest ← extend_forest(ensemble.forest, rotations[$i$].forest[$j$])
        end for
    end for
    return ensemble
end procedure
```

3. $\sum_{r=1}^{R} w(r) = 1$

We consider two weighting schemes that meet these criteria:

- Select the first h rotations from the ordered list and generate a fraction of exactly $w(r) = 1/h$ of the required M base learners on each of these rotations;
- Use an exponential family of curves with decay parameter h to determine the percentage of base learners that should be trained on each rotation.

The first scheme corresponds to selecting the h rotations that are expected to produce the lowest complexity base learners and equal-weight the trees on these rotations. The second scheme includes the possibility of including trees on more different rotations but at much smaller weights. In both cases, h acts as a tuning parameter that can be inferred from the data via a simple grid search, the details of which will be described at the end of this section.

In the first case, the weighting follows the formula

$$w_{cut}(R, h; r) = I(r \leq h)/h, \tag{6}$$

where h is an integer tuning parameter in $[1, R]$, representing a cut-off value and $I(\cdot)$ is the indicator function. Note that the sum of the weights is 1, as expected. For the second case, we use the following family of exponential curves:

$$w_{exp}(R, h; r) = \frac{2^{-r/h}(2^{1/h} - 1)}{(1 - 2^{-R/h})}, \tag{7}$$

where R is the total number of rotations and h is a positive, real tuning parameter. In both cases, r is the sorted (integer) rotation number, as described above. In both cases, small values of h result in large weights for the top rotations and small (or zero) weights for less favorable rotations. By contrast, large h eventually lead to the equal-weighting of rotations.

Figure 1 compares the two weighting schemes. A simple method for obtaining a good tuning parameter h is to use the OOB error estimates of the micro-forests on each rotation and compute the sum product of these errors with and the weight vectors using different values of h – effectively a grid search. Since the rotations are in complexity-sorted order and because the weighting schemes are non-increasing, the resulting weighting will differ significantly from a weighting based solely on OOB predictive accuracy. In Section 5 we show that weightings based solely on OOB predictive accuracy produce base learners that are more complex on average, without a corresponding

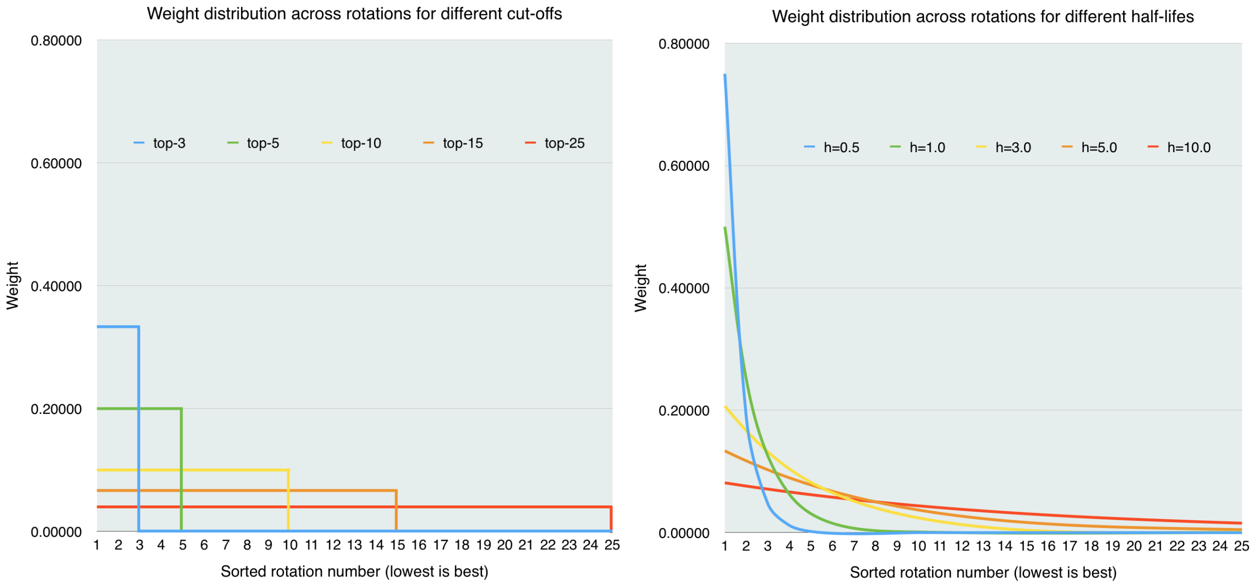

Figure 1: Two parametric families of weight functions: top-h (left) and exponential (right).

out-of-sample performance gain. It should also be noted that in line 9 of Algorithm 1, the weights are multiplied with the ensemble size N and need to be rounded to integer values, since we cannot grow partial trees. In this process, it is possible due to rounding that the sum of the computed number of trees does not add up to N anymore. If this is the case, we automatically add any missing trees to the top rotation or analogously subtract additional trees starting from the worst rotation.

4 Discussion

The goal of this section is to provide an intuitive understanding of which rotations are useful in the context of axis-aligned learners. The discussion applies to higher-dimensional problems but is illustrated in two dimensions.

One visual indication betraying axis-aligned learners, such as decision tree ensembles, is their rugged ("stair-shaped") decision boundary. When a segment of the true decision boundary is not axis-aligned, such learners are forced to approximate the local boundary using a number of smaller steps. The greatest number of such steps is required when the true boundary occurs at a 45-degree angle to one of the axes.

A natural strategy to overcome this predicament is to rotate the space by 45-degrees, such that the decision boundary becomes axis-aligned. After the rotation, only a single hyperplane is necessary to represent the very segment that required many steps prior to rotation. Unfortunately, while rotating the feature space might improve classification locally, it may actually have a negative effect overall, as other segments of the decision boundary might have been well-aligned with the axes prior to rotation but are now poorly-aligned after rotation. For this reason, rotations need to be examined globally and jointly.

To better illustrate the argument, we artificially construct the 2-dimensional, two-class classification problems depicted in Figure 2. For simplicity of the argument, no class overlap is created but the conclusion will be unaffected. The problem on the left-hand side (a) corresponds to the situation where the decision boundary is at a 45-degree angle to both axes. For this problem, we expect a 45-degree (or equivalent) rotation to be optimal.

For the middle problem (b), the decision boundary is flat and axis-aligned but there is a small segment that protrudes at a 45-degree angle to the axes. A zero-degree ro-

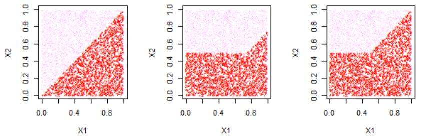

Figure 2: Three illustrative classification problems with known decision boundaries: (a) $y = x$, (b) $y = 0.5 + \max(x - 0.75)$, (c) $y = 0.5 + \max(x - 0.5)$

tation (or equivalent) seems ideal for the longer segment but a 45-degree rotation appears preferable for the smaller segment. Note that since we are running an ensemble of trees, it would be perfectly acceptable to combine one forest trained without rotation with another (perhaps smaller or down-weighted) forest on the rotated space. The question is: which approach produces a better-performing ensemble?

In the final classification problem on the right-hand side (c), the portion of the decision boundary at a 45-degree angle is slightly longer than the axis-aligned section. Here, rotation is likely preferred again. But is it better to rotate by 45-degrees to aid classification near the longer segment or perhaps just by 20-degrees, in such a way that the maximum slope of the decision boundary is reduced at the expense of constructing a problem that is completely unaligned to any axis? In order to answer these questions, we need to define a metric to quantify the value-add provided by a given rotation.

4.1 Tree Complexity

The number of steps required – and hence the average number of nodes required to form a decision tree – generally increases as the boundary becomes less aligned with the axis. This is because the tree construction is done recursively and a new level of the tree is built whenever the local granularity of the tree is insufficient to fully capture the details of the local decision boundary.

For this reason, we propose to use the expected median size of a decision tree as our metric of utility for a given rotation. Rotations that result in smaller, shallower trees on average are considered better rotations. Not only does the metric in Formula (5) assist in creating streamlined trees with fewer spurious splits, it also reduces the computational burden of actually generating and running the full forest. In addition, once we apply Metric (5) to all generated rotations, we are in a position to obtain a ranking of the relative usefulness of each rotation.

In order to compute a reliable and consistent (across rotations) estimate of the proposed metric, we generate a micro-forest for each rotation. It is necessary to create multiple trees to counteract the randomness that is injected in the tree-induction process. For each micro-forest we then compute the median number of nodes used. We use the median in order to actively ignore trees that are artificially inflated by poor (random) variable selections. These operations are computationally efficient when compared to generating a full-blown tree ensemble for each rotation and can generate a stable estimate of the true median. Based on our experiments, the complexity rankings computed on the basis a 10-20-tree forest is very similar to the complexity ranking computed on the basis of a full forest. Hence, the metric is highly predictive and useful.

4.2 Illustration

To demonstrate the usefulness of the proposed metric, we have generated 100 random rotations for each of the two dimensional classification problems listed in Figure 2 above. Figures 3, 4 and 5 illustrate cases (a), (b) and (c), respectively, ranked by tree complexity. Note that the sorting is entirely based on the tree complexity and, importantly, does not make use of the predictive performance of these trees. Despite this, it is interesting to see that the sort reflects our intuition: in Figures 3 and 4, those rotations for which one of the feature boundaries is aligned with one of the axis achieve the best scores, while diagonal boundaries achieve the worst scores. This allows us to find useful rotations without resorting to structured rotations (such as PCA) commonly used in other approaches.

However, if all of the top rotations were chosen solely on the basis of the largest segment of the decision boundary that is axis-aligned, important secondary segments might get neglected, ultimately leading to a worse overall prediction. Figure 5 demonstrates that this is not the case. In this case, the best five rotations again aligned the longer segment to one of the axis, as expected. However, the 6th rotation aligned the shorter segment to the y-axis. This illustrates the point that it may be useful to include multi-

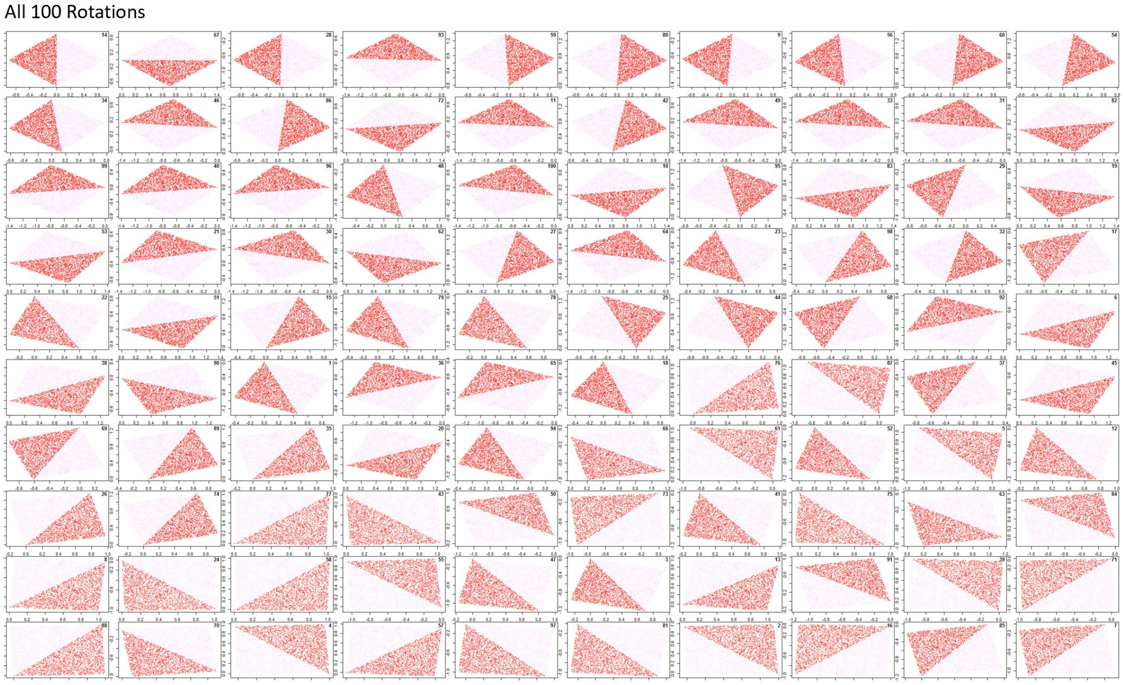

Figure 3: Rotations of classification problem (a) in Figure 2, sorted by expected tree height, as described in the text. Top left is the best rotation, bottom right represents the worst rotation. The small number on the top right of each image is the unique rotation number.

ple rotations in an ensemble, since different rotations can specialize on specific sub-features or decision boundary segments. These results are intuitive and demonstrate the usefulness of the tree-based ranking. What is also striking is that the best rotations do not at all resemble a PCA rotation. This is because the rotation is optimized for alignment of the decision boundary with the tree rather than for the variance of the covariates. This is what sets random rotations apart from rotation forests.

Up to this point, we examined some very simple two-dimensional toy problems with high signal-to-noise ratios (SNRs). In each case, both dimensions were highly informative and contained minimal noise. This setup is ideal for illustrating the method but is not representative of most real-world challenges. Therefore, an important question is how the method performs when we increase the dimensionality or decrease the SNR. To answer this question, we start again with the triangular base shape (a) but incrementally add uniform noise dimensions to the problem before applying the proposed method. In this setting, it is more difficult to visualize the results but we can still demonstrate alignment of the decision boundary with one of the axes by projecting the rotated problem onto the two-dimensional planes formed by the axes – the coordinate surfaces – before plotting. It is important to note that these are *different projections of the same rotation*, rather than different rotations.

Figure 6 demonstrates that the proposed approach is still successful in higher dimensions and with lower signal-to-noise ratios. In these figures, each row represents an exhaustive list of projections onto the coordinate surfaces for a single rotation in p dimensions. The first two dimensions are always the signal dimensions, while the remaining $p-2$ dimensions are random noise dimensions. For example, in the second row of Figure 6 we started with the two original signal dimensions plus one random noise dimension ($p=3$). We then generated 100 rotations and selected the one rotation that was ranked best according to the metric described in Section 3. The row shows the three two-dimensional projections of this best ranked rotation onto the (x,y), (x,z) and (z,y) planes, respectively. It is very apparent that the best rotation aligns the decision boundary with the third axis (the z-coordinate) in this case.

Even when the number of noise dimensions exceeds the number of signal dimensions, as is the case for $p=5$, the alignment of the decision boundary with one of the axes is still very consistent for the best rotation.

In contrast, Figure 7 shows that the worst ranked rotations are not aligned with any axis, regardless of the dimensionality of the problem and that there is a considerable overlap in the two classes at the decision boundary, making it extremely difficult to produce a successful classifier. These examples very clearly show the value of finding high-quality rotations.

5 Performance

In order to test our hypothesis that it is possible to rotate to simplicity without a corresponding performance penalty, we implemented the following weighting schemes:

(a) RRE: Random rotation ensemble, same number of trees on each rotation: M/R.

(b) CUT: Same number of trees on the top-h rotations in terms of complexity (h is chosen using grid search on OOB performance per section 3.1).

(c) EXP: Exponential weighting with half-life h in terms of complexity (h is chosen using grid search on OOB performance per section 3.1).

(d) BST: All N trees on the lowest complexity (best) rotation (equivalent to CUT with $h=1$.

(e) NEW: Same number of trees on all rotations that are ranked higher than or equal to the identity rotation.

(f) LIN: linearly decreasing number of trees: k on lowest complexity rotation, $k-1$ on second lowest, $\ldots 1$ on highest complexity.

(g) OOB: linearly decreasing number of trees: k on lowest OOB error rotation, $k-1$ on second lowest, $\ldots 1$ on highest OOB error.

(h) JNT: linearly decreasing number of trees: k on lowest joint ranking of complexity and OOB error, $\ldots 1$ on highest joint ranking rotation: rank(rank(OOB error) + rank(complexity)).

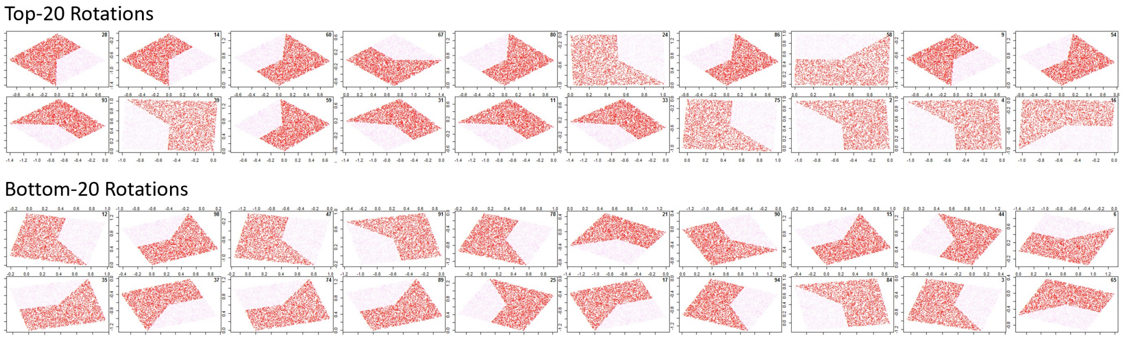

Figure 4: Rotations of classification problem (b) in Figure 2, sorted by expected tree height, as described in the text. The top panel shows the twenty best ranked rotations from top left to bottom right, the bottom panel represents the twenty worst ranked rotations from bottom right to top left. The small number on the top right of each image is the unique rotation number.

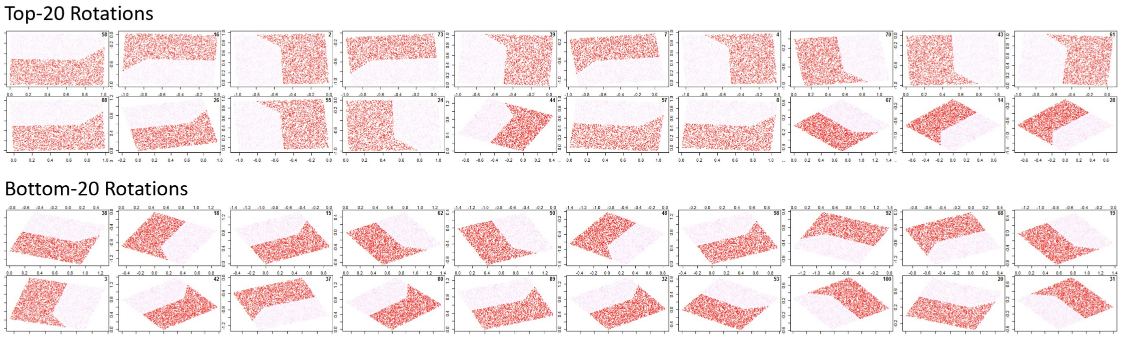

Figure 5: Rotations of classification problem (c) in Figure 2, sorted by expected tree height, as described in the text. the top panel shows the twenty best ranked rotations from top left to bottom right, the bottom panel represents the worst ranked rotations from bottom right to top left. The small number on the top right of each image is the unique rotation number.

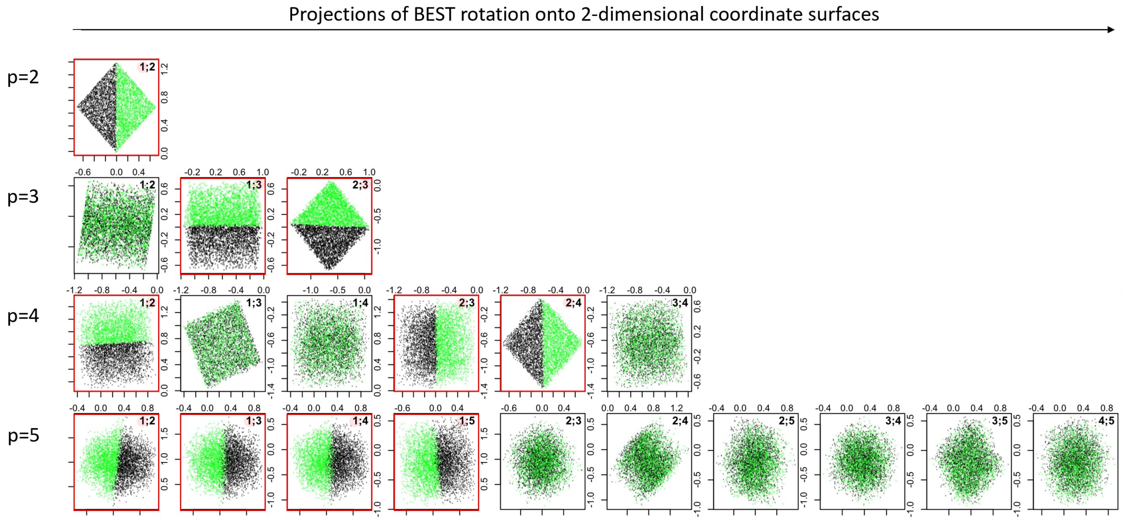

Figure 6: Alignment of the decision boundary of the *best* ranked rotation in p dimensions with each axis. In p dimensions, there are exactly $p(p-1)/2$ coordinate surfaces, meaning two-dimensional planes formed by the p coordinate axes. In this figure, each row depicts the projections of the best rotation in p dimensions onto all available coordinate surfaces. The numbers on the top right of each sub-figure indicate the two axes used to form the specific coordinate surface. For $p = 2$, the signal-to-noise ratio (SNR) is high because only the two signal dimensions were used. For $p > 2$ a total of $p-2$ noise dimensions were added, decreasing the SNR accordingly. Highlighted projections indicate strong alignment with one of the axes.

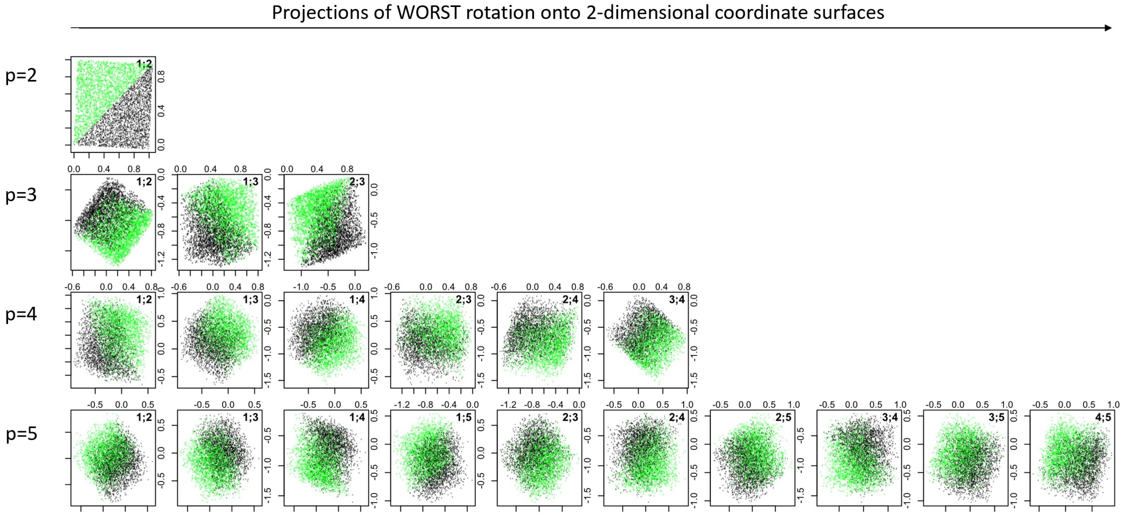

Figure 7: Alignment of the decision boundary of the *worst* ranked rotation in p dimensions with each axis. In p dimensions, there are exactly $p(p-1)/2$ coordinate surfaces, meaning two-dimensional planes formed by the p coordinate axes. In this figure, each row depicts the projections of the worst rotation in p dimensions onto all available coordinate surfaces. The numbers on the top right of each sub-figure indicate the two axes used to form the specific coordinate surface. For $p = 2$, the signal-to-noise ratio (SNR) is high because only the two signal dimensions were used. For $p > 2$ a total of $p-2$ noise dimensions were added, decreasing the SNR accordingly. No alignment is apparent with any of the axes for the worst ranked rotation.

For comparison, we also tested a standard Linear Discriminante Analysis (LDA), as well as three non-linear classifiers: a simple K-Nearest Neighbor classifier (KNN-5), a Support Vector Machine (SVM), and a Gaussian Process classifier (GPR). We have applied the competing methods (GPs and SVMs) in a black-box manner with default parameters available from publicly available software implementations. Hence, their performance is not indicative of the performance that would be achieved if these methods were applied knowledgeably, with state-of-the-art model parameter tuning and consistency checks of the model's assumptions.

For KNN, we used the R implementation in the *class* package with k=5 and for LDA the implementation in *MASS*. For the SVM, we used the R implementation in the *e1071* package with default parameters, that is we used type C-classification with a radial basis function (RBF) kernel and a default gamma of 1/N, which was adjusted to reflect the number of data dimensions and added noise dimensions, where applicable. The cost parameter (or C-parameter in SVM parlance) was set to 1.0. For the GPR, we used the R implementation gausspr in the *kernlab* package. Here we too used the problem type classification with a RBF kernel (rbfdot) and took advantage of the built-in automatic sigma estimation (sigest). We did not attempt to manually- or otherwise tune the meta-parameters of these methods, unless a built-in auto-tuning feature was available, just like we did not tune any parameters in the proposed tree-based methods with the exception of the rotation selection that is the subject of this paper. The overarching goal was to compare methods with sensible default parameters across a number of problem sets in order to determine how to best make use of rotations with axis-parallel learners.

The test procedure generated a random subset of 70% of the data for training purposes and all classifiers were tested on the remaining 30% of the data. This process was repeated 100 times and averages are reported.

With the exception of the identity rotation, all rotations were generated uniformly at random from the Haar distribution. As our base case RRE, we implemented a random rotation ensemble, which does not differentiate between rotations. The only other weighting scheme that does not consider tree complexity at all is OOB, which only takes advantage of OOB errors across the different rotations. Our expectation would be for OOB to outperform in terms of predictive accuracy but with high complexity ensembles. We would also expect BST to produce the lowest complexity ensemble but at the cost of lower predictive performance. In terms of methodology, we first generated 100 random rotations, including one identity rotation. These same rotations were then used by all weighting schemes before the entire process was repeated. In each case, we generated an ensemble with exactly $M = 5000$ trees in total. The dimensionality and number of data points for each data set is listed in table 1. The lowest-dimensional problem with 4 predictors is IRIS and the highest-dimensional problem with 34 predictors is IONO. For space reasons, we refer the reader to Dheeru and Taniskidou (2017) for a detailed description of the UCI data sets we used for testing. Before running the classification algorithms, we scaled all numeric predictors to $[0,1]$.

Table 1: Description of UCI Datasets

UCI Name	Dim (p)	Rows (N)
BREAST	10	699
ECOLI	8	336
GLASS	10	214
IONO	34	351
IRIS	4	150
LIVER	7	345
WINE	13	178
WAVE	21	5000

Table 2 shows the names of the data sets, together with the classification error resulting from applying the different weighting schemes to the rotations. Interestingly, algorithm OOB did not perform quite as well as we had anticipated. For three of the data sets, the scheme performed more than one cross-sectional standard deviation above (worse than) the minimum error. In fact, this appears to be a common pattern among these methods, except for CUT and EXP described in Section 3.1, which are competitive on most of these data sets. One interesting exception was the IRIS data set, for which LDA outperformed all variants of the rotation-based ensembles and indeed all non-linear classifiers. This is an example of where the proposed method does not work as well as expected.

Table 2: Classification error on test data (lower is better). Each rotation weighting method RRE, CUT, EXP, BST, NEW, LIN, OOB, JNT, as described in the text, was applied to the data sets listed under UCI Name. Strikethrough represents a performance number that was more than one cross-sectional standard deviations above (worse than) the minimum.

UCI Name	LDA	SVM	GPR	KNN-5	RRE	CUT	EXP	BST	NEW	LIN	OOB	JNT
BREAST	~~0.0472~~	~~0.0386~~	~~0.0407~~	~~0.0410~~	0.0347	0.0348	0.0349	0.0354	0.0348	0.0349	0.0345	0.0347
ECOLI	0.1209	0.1265	~~0.1305~~	~~0.1365~~	0.1217	0.1233	0.1251	~~0.1445~~	~~0.1448~~	0.1208	0.1212	0.1207
GLASS	~~0.3805~~	~~0.3157~~	~~0.3213~~	~~0.3587~~	~~0.3001~~	0.2729	0.2400	0.2395	0.2378	~~0.2977~~	~~0.2994~~	~~0.3001~~
IONO	~~0.1381~~	0.0609	~~0.1317~~	~~0.1587~~	0.0514	0.0535	0.0544	0.0704	0.0706	0.0519	0.0510	0.0512
IRIS	0.0240	~~0.0407~~	~~0.0502~~	~~0.0416~~	~~0.0467~~	~~0.0451~~	~~0.0453~~	~~0.0456~~	~~0.0460~~	~~0.0467~~	~~0.0462~~	~~0.0464~~
LIVER	~~0.3318~~	0.3064	0.3162	~~0.3981~~	0.3116	0.3050	0.3135	0.3191	0.3089	0.3097	0.3079	0.3099
WAVE	0.1423	0.1383	0.1320	~~0.1815~~	0.1352	0.1380	0.1422	0.1435	0.1432	0.1352	0.1350	0.1351
WINE	0.0161	0.0206	0.0180	~~0.0433~~	~~0.0250~~	0.0176	0.0165	0.0172	0.0150	0.0222	~~0.0224~~	~~0.0232~~

In Table 3 we can confirm that BST really does produce the most compact ensembles. However, unfortunately performance suffers accordingly. A good compromise is EXP, which shows significant reductions in complexity without suffering from performance problems.

For the IRIS data set, EXP resulted in an ensemble that outperformed RRE despite a 24.4% decrease in complexity. Similarly, a 17.5% decrease in complexity was achieved in the GLASS data set. The smallest improvement of merely 2.5% decrease in complexity occured on the IONO data set, for which RRE actually outperformed EXP, although not in a statistically significant manner.

Tables 4 and 5 show the performance of a set of baseline classifiers (SVM, GPR, KNN-5) and the various rotation variants after adding noise dimensions to the data sets IRIS and IONO. It is evident that the performance of the rotation-based classifiers deteriorates relative to other classifiers as the signal-to-noise ratio decreases. This is a known limitation of the method, further described in the following chapter. At the same time, it can be observed that LDA performance is very problem dependent, while KNN and SVM classifiers actually became more competitive in a relative sense with decreasing SNR.

6 Limitations

It was empirically demonstrated in Tomita et al. (2017) that in situations where the signal is contained in a subspace that is small relative to the dimensionality of the feature space, random rotation ensembles tend to underperform ordinary random forests. This is because such a setup renders most rotations unhelpful. By overweighting the most successful rotations, as we propose in this paper, this effect is somewhat mitigated but not entirely eliminated.

Even in the illustrations in Figure 6 it is clear that the quality of the most successful rotations decreases marginally as the number of noise dimensions is increased. The alignment with the axes are not perfect and the noise around the decision boundaries increases visibly. Nonetheless, the rotated features lead to better (axis-aligned) classifiers than the those trained on the unrotated space.

The underlying issue is that rotations in the direction of uninformative noise dimensions do not improve predictions and when the number of noise dimensions is large relative to the signal dimensions, the likelihood of rotating in uninformative directions increases. Note that the same is not necessarily true when the SNR is decreased without increasing the dimensionality of the problem. In this case, random rotations and the ideas in this paper *do not* underperform ordinary random forests in our experience.

One important consideration when introducing rotations into a classifier is that features need to be of comparable scale. We do not explicitly mention this in this paper but a section on recommended scaling mechanisms can be found in Blaser and Fryzlewicz (2016). We do not recommend using any rotation-based ensembles without prior scaling or ranking for practical problems.

Table 3: Tree complexity on test data (lower is better, only relevant for tree-based classifiers). Each rotation weighting method RRE, CUT, EXP, BST, NEW, LIN, OOB, JNT, as described in the text, was applied to the data sets listed under UCI Name. Strikethrough represents a complexity number that was more than one cross-sectional standard deviations above (worse than) the minimum.

UCI Name	RRE	CUT	EXP	BST	NEW	LIN	OOB	JNT
BREAST	~~54.69~~	52.08	51.68	51.11	52.94	53.66	~~54.42~~	~~54.63~~
ECOLI	~~76.82~~	~~71.67~~	69.08	44.52	44.56	~~75.23~~	~~76.68~~	~~76.59~~
GLASS	~~81.68~~	73.69	67.42	66.93	66.95	~~80.84~~	~~81.14~~	~~81.39~~
IONO	~~61.58~~	60.24	60.02	57.58	58.20	~~61.16~~	~~61.51~~	~~61.60~~
IRIS	~~20.98~~	16.52	15.86	15.36	18.67	19.44	~~20.30~~	~~20.98~~
LIVER	~~115.48~~	112.14	111.18	110.84	114.06	~~114.64~~	~~114.94~~	~~115.51~~
WAVE	~~1368.76~~	~~1332.53~~	1310.13	1303.26	1303.29	~~1359.99~~	~~1365.58~~	~~1368.30~~
WINE	~~36.21~~	32.52	31.02	30.88	31.21	~~35.42~~	~~35.42~~	~~36.15~~

Table 4: Classification error on test data (lower is better). Each rotation weighting method and control classifier was applied to the UCI data set IRIS. The first (NOISE) column indicates the number of noise dimensions added to the original data set, from which an upper bound to the signal-to-noise ratio can be estimated as SNR (dB) $\leq$ 10 x log(4/NOISE) by assuming that the original data set is noise free. In the table, SNR represents this upper bound.

NOISE	SNR	LDA	SVM	GPR	KNN-5	RRE	CUT	EXP	BST	NEW	LIN	OOB	JNT
1	6.02	0.0258	0.0578	0.0658	0.0507	0.0516	0.0489	0.0516	0.0560	0.0524	0.0507	0.0524	0.0516
2	3.01	0.0258	0.0533	0.0898	0.0480	0.0569	0.0551	0.0551	0.0587	0.0613	0.0569	0.0604	0.0578
4	0.00	0.0240	0.0827	0.1164	0.0871	0.0853	0.0693	0.0631	0.0640	0.0640	0.0853	0.0889	0.0853
8	-3.01	0.0267	0.1591	0.1680	0.1467	0.1316	0.1093	0.0960	0.0996	0.1013	0.1333	0.1289	0.1289
16	-6.02	0.0533	0.1591	0.1822	0.1200	0.1458	0.1396	0.1511	0.1671	0.1440	0.1440	0.1449	0.1422
32	-9.03	0.0569	0.1618	0.1724	0.1458	0.1591	0.1636	0.1733	0.1724	0.1618	0.1609	0.1644	0.1671
64	-12.04	0.1280	0.2222	0.2516	0.2249	0.2124	0.2116	0.2204	0.2364	0.2151	0.2151	0.2116	0.2133
128	-15.05	0.1618	0.2951	0.3484	0.2596	0.2640	0.2676	0.2684	0.2898	0.2729	0.2658	0.2791	0.2676

7 Computational Considerations

When compared to random rotation ensembles, there is an additional computational cost for regularizing the ensemble. Given the desired total number of trees M, the algorithm requires the generation of micro-forests of size U for each of the R rotations. These micro-forests are essential for estimating the relative efficacy of each rotation. However, depending on the weighting scheme employed, only a subset of the rotations is actually included in the final model.

More specifically, in the initial step, $U \times R$ trees are constructed. However, if the weighting scheme only involves the top r rotations, then $(R - r) \times U$ trees subsequently get discarded. This, in turn, implies that $M - r \times U$ additional trees need to be induced within the r selected rotations to end up with M trees in total within the selected rotations. Expressed as a percentage, we know that $(R - r) \times U / (M - r \times U + R \times U)$ percent of the initially constructed trees get subsequently discarded, resulting in computational overhead when compared to random rotation ensembles, where all trees are used.

In order to obtain a bound on this expression, note that $R \times U <= M$. This is because it is not practical to generate more trees in the micro-forests than are needed in total. Hence, in the worst case $(R - r)/(2R - r)$ percent of the initially constructed trees get subsequently discarded, a quantity that is smaller than $1/2$ because r is in $[1, R]$ and R is in $[1, M]$. This expression is maximized when only the best rotation is selected $(r = 1)$ and minimized when all rotations are selected $(r = R)$. Therefore, in terms of computational overhead, the worst case is that nearly twice as

Table 5: Classification error on test data (lower is better). Each rotation weighting method and control classifier was applied to the UCI data set IONO. The first (NOISE) column indicates the number of noise dimensions added to the original data set, from which an upper bound to the signal-to-noise ratio can be estimated as SNR (dB) $\leq$ 10 x log(34/NOISE) by assuming that the original data set is noise free. In the table, SNR represents this upper bound.

NOISE	SNR	LDA	SVM	GPR	KNN-5	RRE	CUT	EXP	BST	NEW	LIN	OOB	JNT
1	15.31	0.1479	0.0675	0.1423	0.1725	0.0600	0.0604	0.0600	0.0721	0.0706	0.0596	0.0592	0.0592
2	12.30	0.1234	0.0551	0.1294	0.1509	0.0521	0.0558	0.0589	0.0702	0.0691	0.0528	0.0521	0.0517
4	9.29	0.1577	0.0725	0.1464	0.1675	0.0675	0.0683	0.0702	0.0868	0.00875	0.0679	0.0698	0.0683
8	6.28	0.1506	0.0668	0.1408	0.1706	0.0668	0.0660	0.0717	0.0808	0.0815	0.0664	0.0668	0.0679
16	3.27	0.1438	0.0634	0.1362	0.1672	0.0657	0.0615	0.0747	0.0789	0.0785	0.0649	0.0664	0.0642
32	0.26	0.1642	0.0792	0.1638	0.1864	0.0864	0.0808	0.0879	0.0906	0.0838	0.0860	0.0857	0.0842
64	-2.75	0.1951	0.1155	0.2257	0.1898	0.1521	0.1408	0.1385	0.1423	0.1442	0.1525	0.1498	0.1528
128	-5.76	0.2921	0.1479	0.2687	0.2083	0.2230	0.2242	0.2200	0.2208	0.2211	0.2238	0.2245	0.2242

many trees need to be constructed when the ensemble is regularized than for standard random rotation ensembles.

In practice, this bound is unrealistically high and the magnitude of the overhead can be influenced by selecting sensible parameters. For example, using $M = 5000$, $R = 50$ and $U = 10$ and utilizing the top $r = 10$ rotations, we achieve a computational overhead of merely $(50 - 10) \times 10/(5000 - 10 \times 10 + 50 \times 10) = 2/27$, or less than 7.41%. In addition, it should be noted that this overhead gets partially offset by the fact that only R rotations need to be generated with our method instead of M for random rotation ensembles. In the current example, that number is 50 instead of 5000.

Besides these rather modest effects, the computational complexity of our method is equivalent to that of random rotation ensembles, regardless of the number of training samples or data dimensions.

8 Software

The authors have released an open source R package by the name of *random.rotation*. The package contains a reference implementation of random rotations, including the weighting- and regularisation methods described in this paper. The package can be downloaded from GitHub without registration. The easiest way to accomplish this is directly within an R command-line shell:

```
library("devtools")
install_github("randomrotation/random.rotation")
```

9 Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and constructive feedback.

References

Theodore Anderson, Ingram Olkin, and Les Underhill. Generation of random orthogonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 8:625–629, 1987.

Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. *Journal of Machine Learning Research*, 17: 1–26, 2016.

Leo Breiman. Random forests random features. *Technical report*, 1999.

Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and regression trees*. Wadsworth, Belmont, CA, 1984.

Timothy Cannings and Richard Samworth. Random-projection ensemble classification. *Journal of the Royal Statistical Society: Series B*, 79:1–38, 2017.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Robert Durrant and Ata Kaban. Random projections as regularizers: learning a linear discriminant ensemble from fewer observations than dimensions. *JMLR: Workshop and Conference Proceedings*, 29:17–32, 2013.

Jerome Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 5: 1189–1232, 2001.

Michael Goodrich, Vincent Mirelli, Mark Orletsky, and Jeffery Salowe. Decision tree construction in fixed dimensions: being global is hard but local greed is good. *Technical Report TR95-1*, 1995.

Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The elements of statistical learning: data mining, inference, and prediction*. Springer, New York, 2009.

Laurent Hyafil and Ronald Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976.

Gareth James, Daniela Witten, and Trevor Hastie. *An introduction to statistical learning*. Springer, New York, 2013.

Juan Rodriguez, Ludmila Kuncheva, and Carlos Alonso. Rotation forest: a new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1619–1630, 2006.

Gabriel Takacs, Vijay Chandrasekhar, and Sam Tsai. Fast computation of rotation-invariant image features by approximate radial gradient transform. *IEEE Transactions on Image Processing*, 22:2970–2982, 2013.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–88, 1996.

Tyler Tomita, Mauro Maggioni, and Joshua Vogelstein. Roflmao: Robust oblique forests with linear matrix operations. *Proceedings of the 2017 SIAM International Conference on Data Mining*, 1:498–506, 2017.