Canonical Correlation Forests

Tom Rainforth
Department of Engineering Science
University of Oxford
twgr@robots.ox.ac.uk

Frank Wood
Department of Engineering Science
University of Oxford
fwood@robots.ox.ac.uk

Abstract

We introduce canonical correlation forests (CCFs), a new decision tree ensemble method for classification. Individual canonical correlation trees are binary decision trees with hyperplane splits based on canonical correlation components. Unlike axis-aligned alternatives, the decision surfaces of CCFs are not restricted to the coordinate system of the input features and therefore more naturally represent data with correlation between the features. Additionally we introduce a novel alternative to bagging, the projection bootstrap, which maintains use of the full dataset in selecting split points. CCFs do not require parameter tuning and our experiments show that they significantly out-perform axis-aligned random forests and other state-of-the-art tree ensemble methods.

1 Introduction

Decision tree ensemble methods such as random forests [1], extremely randomized trees [2], rotation forests [3] and boosted decision trees [4] are widely employed methods for classification and regression due to their scalability, fast out of sample prediction and tendency to require little parameter tuning. In many cases they are capable of giving predictive performance close to, or even equalling, state of the art when used in an out-of-box fashion. In this paper we introduce canonical correlation forests (CCFs), a new tree ensemble method for classification where the individual canonical correlation trees (CCTs) use hyperplane splits based on the feature projections from a canonical correlation analysis (CCA) [5] between the input features and class labels. Unlike many previous oblique (i.e. non axis-aligned) decision tree methods, CCFs are equally suited to both binary and multi-class classification. CCA is carried out in a numerically stable and efficient way, avoiding the numerical issues encountered by standard implementations of linear discriminant analysis (LDA) in oblique decision trees. We also introduce the projection bootstrap, a novel alternative to the bagging scheme used by many tree ensemble methods for decorrelating the prediction of individual trees. Open source code for implementation of the CCF method is available online[1].

A decision tree is a predictive model that imposes sequential divisions of an input space to form a set of partitions known as leafs, each containing a local classification or regression model. Out of sample prediction is performed by using the partitioning structure to assign a data point to a particular leaf and then using the corresponding local predictive model. Typically the leaf models are taken to be independent of each other and the class labels are assumed to be independent of input features given the leaf assignments.

Classical decision tree learning algorithms work in a greedy top down fashion, exhaustively searching the possible space of axis-aligned unique split points and choosing the best based on a splitting criterion, such as the Gini gain or information gain used in CART [6] and C4.5 [7] respectively. This process continues until no further split is advantageous or some user-set limit is reached (i.e. a minimum number of points for an internal leaf). For classification with continuous features this

[1] *www.robots.ox.ac.uk/∼twgr/*

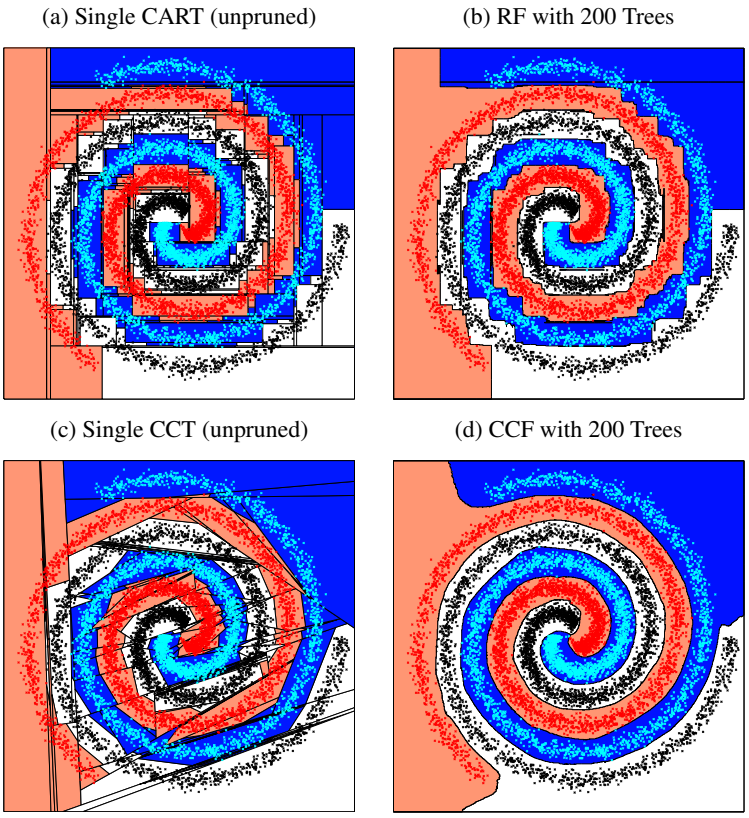


Figure 1: Decision surfaces for artificial data spiral dataset. (a) shows the hierarchical partitions and surface for a single axis aligned tree while (b) shows the effect over averaging over a number of, individually randomized, axis aligned trees. (c) shows a single oblique decision tree and (d) demonstrates that averaging over oblique decision trees leads to "smoother" decision surfaces which better represents the data than the axis aligned equivalent.

typically only occurs once each leaf is "pure," containing only data points of a single class. When used as individual classifiers, trees are usually "pruned" after being grown to prevent overfitting, though as found by Ho [8] this is not usually advantageous in forests.

It was established by Ho [8] that combining individual trees to form a decision forest can simultaneously improve predictive performance and provide regularization against over-fitting. In a forest, each tree is separately trained, with predictions based on a voting system across the ensemble. As classical decision tree algorithms are deterministic procedures, such combination requires the introduction of probabilistic elements into the generative process to prevent identical trees. The random subspace method used by Ho (refined in a later paper [9]) involves only searching splits along a randomly selected subset of the features at each node. Breiman [10] proposed the alternative scheme of bagging in which each each predictor is trained on a bootstrap sample of the original dataset. Breiman later combined these schemes to great effect in his random forest (RF) algorithm [1]. RFs have since become very popular across many fields, largely due to their ability to generate good performance when used in an out of box fashion.

As shown in figure 1, the decision surfaces of finite ensembles of axis-aligned decision trees are restricted to be piecewise axis aligned, even when there is little evidence for this in the data. Naturally this restriction is often detrimental, particularly when the quantity of training data is small or when prediction is required outside the convex hull of the training data. Breiman [1] demonstrated that splitting using random linear combinations of features can alleviate this issue and give a slight performance improvement, but he did not investigate alternative feature combination methods.

Prior to the development of random forests, significant research was made into oblique decision trees. Their core idea was to find splits based on the best combination of the available features,

either attempting to directly optimize for the hyperplane representing the best partition, such as in OC1 [11], or carrying out a linear discriminant analysis (LDA) to find a projection which optimizes some discriminant criterion and then searching over possible splits in this projected space, such as functional trees [12] and QUEST [13]. Although these generally produced better results than single axis aligned trees, there were a number of common issues such as a failure to effectively deal with multiple classes and numerical instability in the LDA procedure. Most also carried out a simplified version of LDA, making the assumption that the classes are normally distributed with the same covariance, as opposed to Fisher's original linear discriminant analysis (FLDA) [14] which simply finds the hyperplane that maximises the ratio of the between class variance and the within class variance without these assumptions.

Lemmond et al [15] and Menze et al [16] both introduce the idea of creating forests of oblique decision trees using splits based on LDA projections. Neither method is applicable to multi-class classification and neither carries out the LDA in a manner that is both numerically stable and computationally efficient. The latter paper also introduce the idea of carrying out LDA as a ridge regression, where there is regularization towards the principle component directions. However, their results suggest no advantage is gained by this regularization.

Rotation forests [3] instead apply probabilistic rotations based around principle component analysis to the original coordinate system, so that different trees are trained in a different coordinate system. Although rotation forests give significant improvements over RF and AdaBoost [17], they do not use feature sub spacing and so all features are considered for splitting at every node, leading to a very computationally expensive algorithm relative to RF for more than a modest number of features.

2 Canonical Correlation Forests

2.1 Forest Definition and Notation

Although the model we introduce can easily be extended to regression problems, our focus will be on classification. Our aim will be to predict classes labels $y_n \in \{1, \ldots, K\}$ given a vector of input features $x_n \in \mathbb{R}^D$ for each data point $n \in \{1, \ldots, N\}$. We will denote the set of labels $Y = \{y_n\}_{n=1}^N$ and the set of feature vectors $X = \{x_n\}_{n=1}^N$. Let $T = \{t_i\}_{i=1\ldots L}$ denote a forest comprised of binary trees t_i, where L is a user set parameter dictating the ensemble size. The model operates in a train / test fashion in which T is learnt using training data and out of sample predictions are made independently of the training data conditioned on T. Each individual tree $t = \{\Psi, \Theta\}$ is defined by a set of discriminant nodes $\Psi = \{\psi_j\}_{j \in \mathcal{J} \setminus \partial\mathcal{J}}$ and a set of leaf nodes $\Theta = \{\theta_j\}_{j \in \partial\mathcal{J}}$ where $\mathcal{J} \subset \mathbb{Z}^{\geq 0}$ is a set of node indices and $\partial\mathcal{J} \subseteq \mathcal{J}$ is the subset of leaf node indices. Each discriminant node is defined by the tuple $\psi_j = \{\chi_{j,1}, \chi_{j,2}, \phi_j, s_j\}$ where $\{\chi_{j,1}, \chi_{j,2}\} \subseteq \mathcal{J} \setminus j$ are the two child node ids, $\phi_j \in \mathbb{R}^D$ is a weight vector used to project the input features and s_j is the point at which the splitting occurs in the projected space $X^T\phi_j$. Note that for orthogonal trees only a single element of ϕ_j will be non-zero, whereas oblique trees will have multiple non-zero elements. Let $B(j,t)$ denote the partition of the input space associated with node j such that $B(0,t) = \mathbb{R}^D$ and $B(j,t) = B(\chi_{j,1}, t) \cup B(\chi_{j,2}, t)$. The partitioning procedure is then defined such that

$$\begin{aligned} B(\chi_{j,1}, t) &= B(j,t) \cap \left\{ z \in \mathbb{R}^D : z^T\phi_j \leq s_j \right\} \\ B(\chi_{j,2}, t) &= B(j,t) \cap \left\{ z \in \mathbb{R}^D : z^T\phi_j > s_j \right\}. \end{aligned} \tag{1}$$

Thus Ψ defines a hierarchical partitioning procedure that deterministically assigns data points to leaf nodes, with prediction then based on the corresponding local leaf model. Although more complicated leaf models are possible (e.g. logistic regression models [12]), in this paper we only consider the case where the leaf models are deterministic assignment to a particular class, thus $\theta_j \in \{1, \ldots, K\}\ \forall j \in \partial\mathcal{J}$.

Slightly abusing notation, let $t_i(x_n) \in \mathbb{I}^{1 \times K}$ denote a 1-of-K encoding for the prediction of data point n by tree i. Under the equal weighted voting scheme used, the vector

$$v_n = \frac{1}{L} \sum_{i=1}^{L} t_i(x_n) \tag{2}$$

represents the predictive probabilities across classes assigned by the forest.

2.2 Canonical Correlation Analysis

For $W \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{n\times k}$, CCA [5] produces $\nu_{\max} = \min(\text{rank}(W), \text{rank}(V))$ pairs of coefficients $\{A_\nu, B_\nu\}_{\nu\in\{1,\ldots,\nu_{max}\}}$ such that

$$\{A_1, B_1\} = \underset{a,b}{\text{argmax}} \left(\text{corr}\left(Wa, Vb\right)\right) \text{ where } a \in \mathbb{R}^{d\times 1},\ \|a\|_2 = 1,\ b \in \mathbb{R}^{k\times 1},\ \|b\|_2 = 1 \quad (3)$$

and subsequent coefficients are defined by the same optimization with the additional constraint that the new canonical correlation components, WA_ν and VB_ν where $\nu \in \{2, \ldots, \nu_{\max}\}$, must be uncorrelated with the previous components. Therefore CCA deterministically gives the sets of projections that maximises the correlation between two matrices. Note that CCA is a co-ordinate free process that is unaffected by rotation, translation or global scaling of the input vectors W and V. A more complete overview of CCA is provided in the supplementary material.

2.3 Canonical Correlation Forest Training Algorithm

CCFs are formed of ensembles CCTs. A CCT is an oblique decision tree trained by using CCA to find feature projections that give the maximum correlation between the features and a coordinate free representation of the class labels and then select the best split in this projected space. Formally given array X and a 1-of-K encoding of the class labels $Y \in \mathbb{R}^{N\times 1} \rightarrow \mathcal{Y} \in \mathbb{I}^{N\times K}$, where $\mathcal{Y}_{nk} = 1$ indicates point n belongs to class k, we calculate

$$[\Phi, \cdot] = \text{CCA}\left(X, \mathcal{Y}\right) \quad (4)$$

where Φ are the canonical coefficients corresponding to X. For node j, the split projection vector ϕ_j is taken as the column of Φ for which the best split occurred during training and s_j as the corresponding best split point in $X\phi_j$. The process of finding the best split projection vector and split point combination corresponds to the function FINDBESTSPLIT$(\cdot)$ in Algorithm 2 which also returns a gain representing that value of split criterion for the selected split relative to if the node were a leaf. Note that the CCA is only required during the training phase with the splitting rule (1) used directly for out of sample prediction. Algorithm 1 gives a step by step process for the generation of a CCF. We use $\leftarrow$ to denote assignment and MATLAB notation with subscript parentheses for indexing

Algorithm 1 Canonical correlation forest training algorithm

Inputs: $X^r \in \mathbb{R}^{N\times D^r},\ X^c \in \mathcal{S}^{N\times D^c},\ \mathcal{Y} \in \mathbb{I}^{N\times K}, L \in \mathbb{Z}^+,\ \lambda \in \mathbb{Z}^+$ ▷ $\mathcal{S}$ represents non-ordinal space
Convert X^c to 1-of-K encoding $X^b \in \mathbb{I}^{N\times D^b}$, $\quad X = \left\{X^r, X^b\right\}$
$\mu_{(d)} = \sum_{n=1}^{N} X_{(n,d)}/N, \quad \sigma_{(d)} = \sqrt{\sum_{n=1}^{N} {X_{(n,d)}}^2 - {\mu_{(d)}}^2}/\sqrt{(N-1)} \quad \forall d$
$X_{(:,d)} \leftarrow \left(X_{(:,d)} - \mu_{(d)}\right)/\sigma_{(d)} \quad \forall d,$ set missing values in X to 0
if $\lambda < (D^r + D^c)$ **then** $b = \text{true}$ **else** $b = \text{false}$ **end if**
for $i = 1\colon L$ **do**
    if b **then** $\{X', \mathcal{Y}'\} \leftarrow \{X, \mathcal{Y}\}$ **else** $\{X', \mathcal{Y}'\} \leftarrow$ sample with replacement N rows from $\{X, \mathcal{Y}\}$ **end if**
    $[\cdot, \Psi, \Theta] = \text{GROWTREE}(X', \mathcal{Y}', \{1, \ldots, D^r + D^c\}, \lambda, b)$
    $t_i = \{\Psi, \Theta\}$
end for
return $T = \{t_i\}_{i=1\ldots L}$

such that a colon indicates all the values along a dimension, a vector index indicates assignment to the corresponding subarray and the notation $:\backslash i$ indicates all the indices except those in set i.

Although the CCF training algorithm still employs feature subspacing in the same way as RF, it does not use bagging[2]. Instead we introduce the projection bootstrap which calculates Φ using a local bootstrap sample of the data points $\{X', \mathcal{Y}'\}$, but then searches over possible splits in the projected space $X\Phi$ using the original dataset $\{X, \mathcal{Y}\}$ such that no information is discarded in the choice of $\{\phi_j, s_j\}$ given Φ. Prior to running the CCA, the bootstrap sample is tested to ensure it contains more than one class and more than two unique data points. If there is only a single class or one unique point, one can either just assign the node to be a leaf or replace the bootstrap sample with the original data when this does not have the same degeneracy. We found that either can be

[2] An exception to this is that bagging is used instead of the projection bootstrap when the number of features to be sampled, λ, is equal to the total number of present features. This is done to avoid overfitting.

Algorithm 2 GROWTREE

1: Inputs $X^j \in \mathbb{R}^{N^j \times D^r + D^b}$, $\mathcal{Y}^j \in \mathbb{I}^{N^j \times K}$, $\mathcal{D}^j \subseteq \{1, \ldots, D^r + D^c\}$, $\lambda \in \mathbb{Z}^+, b \in \{\text{true}, \text{false}\}$
2: Set current node index j to an unique node identifier (0 for root node)
3: Sample $\delta \subseteq \mathcal{D}^j$ by taking $\min\left(\lambda, \left|\mathcal{D}^j\right|\right)$ samples without replacement from $\mathcal{D}^j$
4: While δ contains features without variation, eliminate these from $\mathcal{D}^j$ and δ and resample
5: $\gamma = \delta$ mapped to the column indices of X^j in accordance with the 1-of-K encoding of X^c
6: **if** b **then** $\{X', \mathcal{Y}'\} \leftarrow$ sample with replacement N^j rows from $\{X^j_{(:,\gamma)}, \mathcal{Y}^j\}$
7: **else** $\{X', \mathcal{Y}'\} \leftarrow \{X, \mathcal{Y}\}$ **end if**
8: **if** all rows in X' or $\mathcal{Y}'$ are identical **then**
9: **if** all rows in $X^j_{(:,\gamma)}$ or $\mathcal{Y}^j$ are identical **then return** $\left[j, \emptyset, \text{LABEL}(\mathcal{Y}^j)\right]$ **end if**
10: $\{X', \mathcal{Y}'\} \leftarrow \{X^j_{(:,\gamma)}, \mathcal{Y}^j\}$ ▷ LABEL described in Section 2.3
11: **end if**
12: **if** X' contains only two unique rows **then**
13: $\mathcal{X}' = \text{UNIQUEROWS}(X')$
14: $\phi_{j\,(\gamma)} \leftarrow \mathcal{X}'_{(2,:)} - \mathcal{X}'_{(1,:)}, \quad \phi_{j\,(:\setminus\gamma)} \leftarrow 0$
15: **else**
16: $[A, \cdot] = \text{CCA}(X', \mathcal{Y}')$ ▷ as per Section 2.5
17: $R_{(\gamma,:)} \leftarrow A, \quad R_{(:\setminus\gamma,:)} \leftarrow 0$
18: $U = X^j R$
19: $[\xi, s_j, \text{gain}] = \text{FINDBESTSPLIT}(U)$ ▷ as described in Section 2.3
20: **if** gain ≤ 0 **then return** $\left[j, \emptyset, \text{LABEL}(\mathcal{Y}^j)\right]$ **end if**
21: $\phi_j = R_{(:,\xi)}$
22: **end if**
23: $\tau_l = \left\{n \in \{1, \ldots, N^j\} : X^j_{(n,:)}\phi_j \leq s_j\right\}, \quad \tau_r = \{1, \ldots, N^j\} \setminus \tau_l$
24: $\left[\chi_{(j,1)}, \Psi_l, \Theta_l\right] = \text{GROWTREE}(X^j_{(\tau_l,:)}, \mathcal{Y}^j_{(\tau_l,:)}, \mathcal{D}^j, \lambda, b)$
25: $\left[\chi_{(j,2)}, \Psi_r, \Theta_r\right] = \text{GROWTREE}(X^j_{(\tau_r,:)}, \mathcal{Y}^j_{(\tau_r,:)}, \mathcal{D}^j, \lambda, b)$
26: $\psi_j = \left\{\chi_{(j,1)}, \chi_{(j,2)}, \phi_j, s_j\right\}$
27: **return** $[j, \{\psi_j \cup \Psi_l \cup \Psi_r\}, \{\Theta_l \cup \Theta_r\}]$

preferable depending on the dataset but take the latter as the default behaviour as this performed better on average. If there are two unique points the discriminant projection is set to be the vector between the two points, as shown in line 14 of Algorithm 2, instead of carrying out a CCA and the process continues as normal. The process of assigning a label to leaf, i.e. the LABEL($\mathcal{Y}$) function in Algorithm 2, is simply the most populous class at the label. In the event of a tie, the function assigns the most populous of the tied classes at the parent node, recursing up the tree if required.

2.4 Data Preprocessing

The format of our forest definition given in Section 2.1 requires the data to be in numerical form. Ordered categorical features can be simply converted to numerical features based on the order. For unordered categorical features $x^c \in \mathcal{S}$, where $\mathcal{S}$ represents the space of arbitrary qualitative attributes, we use a 1-of-K encoding. To ensure equal probability of selecting categorical and numerical features, the expanded binary array of each categorical feature is still treated as a single feature for the purposes of feature subspacing.

As described in lines 3-4 of Algorithm 1, we suggest converting data points to their corresponding z-scores as a preprocessing step. Although this will make no direct change to the canonical correlation components (as CCA is unaffected by affine transformations of the features), it does effect the rank reduction used in ensuring the numerically stability of CCA as discussed in Section 2.5. Missing data is dealt with by setting its value to the training data mean (note the mean and standard deviations are calculated disregarding missing values).

2.5 Numerically Stable CCA

The closed form solution for CCA based on eigenvectors, given for example by Borga [18], can be numerically unstable as it requires inversion of the potentially degenerate covariance matrices. For example, if their are more features than data points the covariance matrix is certain to be degenerate.

Given the sequential partitioning of data in a decision tree, such degeneracy will become common as the tree depth increases, even if the covariances for the complete dataset are not degenerate. However, Björck and Golub [19] demonstrated that the solution for CCA can also be found in a numerically stable way. For both inputs, a QR decomposition with pivoting is carried out such that for input W, $QR = WP$ where Q is a unitary matrix, R is an upper triangular matrix and P is a pivot matrix such that the diagonal elements of the R matrices are of decreasing magnitude. If

$$\zeta = \max\left\{i : \left|R_{(i,i)}\right| > \varepsilon \left|R_{(1,1)}\right|\right\} \tag{5}$$

is the number of non-zero main diagonal terms of zero within some tolerance, then the first ζ columns of Q will describe an orthonormal basis for the span of W. Therefore by applying the reductions $Q' \leftarrow Q\left(:, 1:\zeta\right)$ and $R' \leftarrow R\left(1:\zeta, 1:\zeta\right)$, then $Q'R'$ will be a pivoted reduction of W that is full rank and R' will be invertible. The algorithm then proceeds with the reduced matrices Q' and R' to carry out the CCA in a numerical stable manner, full details are provided in the supplementary material. Although for analytical application then the rank tolerance parameter ε should be taken as 0^+, we recommend taking a finite value (we use $\varepsilon = 10^{-4}$) to guard against numerical error and because this can act as a regularization term against individual splits overfitting the inputs. Note that packaged applications of this algorithm are available, for example CANONCORR in MATLAB, but in general these do not allow ε to be set manually.

3 Experiments

To investigate the predictive performance of CCFs, we ran comparison tests against the common state-of-the-art algorithms random forest (RF) and rotation forest over a broad variety of datasets. The results show that CCFs significantly outperformed both methods, creating a new benchmark in classification accuracy for out-of-box decision tree ensembles. In addition to comparing to these state-of-the-art methods, we also compared to a reduced version of our algorithm where we use tree bagging, as per RF, as an alternative to the projection bootstrap. We refer to this method as CCF-Bag. It should be noted that the rotation forest algorithm is a considerably more computationally expensive algorithm than the other methods as discussed in Section 4.1. For each method the ensemble was composed of $L = 200$ trees, noting that as RF converge with increasing L [1], the selection of L need only be based on computational budget. Each tree used the information gain split criterion of C4.5 [7] as the basis for choosing the best split, as is the default for WEKA's [20] implementaions of RF and Rotation Forest. Although we also tried the Gini split criterion (the split criterion in MATLABs TREEBAGGER function and Breiman's original [1] implementation) for RFs and CCFs, we found that the information gain based split criterion dominated in both cases and so we omit the results from these tests. The RF, CCF-Bag and CCF algorithms were all implemented in MATLAB and we set the parameter for the number of features to sample at each step to $\lambda = \text{CEIL}(\log_2\left(D^r + D^c\right) + 1)$ (note this is based on the number of features prior to the binary expansion of categorical features), with the exception that we set $\lambda = 2$ when $D^r + D^c = 3$ so that random subspacing and CCA can both be employed. Rotation Forests were implemented in WEKA with the same number of trees and the default options except that we used binary, unpruned, trees and set the minimum number of instances per leaf 1. In addition to keeping the implementation of rotation forest as consistent as possible with the other algorithms, these settings dominated rotation forests of the same size with the default options over a single cross validation. As recommended by Rodriguez et al [3], 1-of-K encoding was used for non-ordianl features for rotation forests (note rotation forests do not then treat them differently to ordinal variables).

For each dataset, 15 different 10-fold cross-validation tests were performed. The majority of the 37 datasets were taken from the UCI machine learning database [21] with the exceptions of the *ORL* face recognition dataset [22], the *Polyadenylation Signal Prediction* (*polya*) dataset [23] and the artificial spiral dataset from figure 1. Summaries of the datasets along with the results are given in Table 1. Note for the *vowel-c* dataset the sex and identifier for the speaker are included whereas these are omitted for the *vowel-n* dataset which is otherwise identical. The *wholesale-c* and *wholesale-r* datasets correspond to predicting the *channel* and *region* attributes respectively.

4 Discussion

Table 2 shows a summary of results over all the datasets, giving the number of datasets for which the performance of one dataset was significantly better than another at the 1% level of a Wilcoxon

Data set	K	N	D^c	D^r	CCF	CCF-Bag		RF		Rotation Forest	
Balance scale	3	625	0	4	8.94 ± 3.74	8.74 ± 3.48		16.19 ± 4.23	•	**7.29 ± 3.43**	○
Banknote	2	1372	0	4	**0.00 ± 0.00**	**0.00 ± 0.00**		0.72 ± 0.75	•	**0.00 ± 0.00**	
Breast tissue	6	106	0	9	**28.42 ± 11.79**	28.91 ± 12.38		30.97 ± 12.99	•	**28.42 ± 12.70**	
Climate crashes	2	360	0	18	**5.83 ± 4.04**	6.44 ± 4.09	•	7.13 ± 4.18	•	5.96 ± 3.89	
Fertility	2	100	0	9	13.27 ± 9.38	13.53 ± 9.70		13.60 ± 9.78		**12.33 ± 8.78**	
Heart-SPECT	2	267	0	22	**17.16 ± 7.13**	18.02 ± 6.71	•	18.62 ± 7.27	•	17.51 ± 7.29	
Heart-SPECTF	2	267	0	44	18.54 ± 7.34	**18.02 ± 6.89**		18.99 ± 6.71		18.77 ± 7.29	
Hill valley	2	1212	0	100	**0.00 ± 0.00**	**0.00 ± 0.00**		38.98 ± 4.34	•	6.26 ± 2.66	•
Hill valley noisy	2	1212	0	100	**5.01 ± 1.85**	5.62 ± 2.02	•	42.02 ± 4.41	•	11.25 ± 2.84	•
ILPD	2	640	0	10	28.03 ± 5.05	**27.97 ± 5.21**		29.66 ± 4.95	•	29.02 ± 5.20	
Ionosphere	2	351	0	33	**4.88 ± 3.63**	5.62 ± 3.78	•	6.44 ± 3.89	•	5.70 ± 3.51	•
Iris	3	150	0	4	2.44 ± 3.89	**2.31 ± 3.78**		5.07 ± 5.39	•	4.18 ± 5.10	•
Landsat satellite	2	6435	0	36	8.24 ± 1.08	8.70 ± 1.05	•	8.16 ± 1.01		**7.81 ± 0.99**	○
Letter	26	20000	0	16	**2.25 ± 0.33**	2.58 ± 0.36	•	3.36 ± 0.38	•	2.48 ± 0.32	•
Libras	15	360	0	90	10.30 ± 4.75	11.35 ± 5.23	•	18.70 ± 5.97	•	**9.72 ± 4.79**	
MAGIC	2	19020	0	10	**11.58 ± 0.72**	11.71 ± 0.72	•	11.85 ± 0.74	•	12.64 ± 0.72	•
Nursery	5	12960	0	8	0.04 ± 0.07	0.09 ± 0.11	•	0.33 ± 0.19	•	**0.04 ± 0.06**	
ORL	40	400	0	10304	**2.18 ± 2.25**	2.62 ± 2.75		2.45 ± 2.52		-	
Optical digits	10	5620	0	64	**1.29 ± 0.45**	1.44 ± 0.46	•	1.65 ± 0.49	•	1.31 ± 0.40	
Parkinsons	2	195	0	22	**6.10 ± 5.43**	7.73 ± 6.01	•	9.03 ± 6.04	•	7.61 ± 5.44	•
Pen digits	10	10992	0	16	**0.40 ± 0.19**	0.46 ± 0.21	•	0.83 ± 0.29	•	0.49 ± 0.23	•
Polya	2	9255	0	169	21.18 ± 1.31	21.28 ± 1.28		21.28 ± 1.36		**20.21 ± 1.32**	○
Seeds	3	210	0	7	**4.79 ± 4.73**	5.43 ± 5.18	•	6.38 ± 5.23	•	4.86 ± 4.53	
Skin seg	2	245057	0	3	**0.03 ± 0.01**	0.03 ± 0.01	•	0.04 ± 0.01	•	0.04 ± 0.01	•
Soybean	19	683	13	22	**5.42 ± 2.94**	5.82 ± 3.14	•	5.56 ± 3.08		5.60 ± 2.92	
Spirals	3	10000	0	2	**0.27 ± 0.16**	**0.27 ± 0.16**		1.22 ± 0.33	•	1.02 ± 0.34	•
Splice	3	3190	60	0	**3.10 ± 0.97**	3.29 ± 1.12	•	3.12 ± 0.93		4.26 ± 1.18	•
Vehicle	4	846	0	18	**17.31 ± 3.93**	17.32 ± 4.08		25.26 ± 4.64	•	20.91 ± 4.39	•
Vowel-c	11	990	2	10	**0.94 ± 0.95**	1.28 ± 1.08	•	2.65 ± 1.72	•	1.00 ± 0.95	
Vowel-n	11	990	0	10	1.99 ± 1.32	2.82 ± 1.64	•	3.25 ± 1.77	•	**1.48 ± 1.23**	○
Waveform (1)	3	5000	0	21	13.58 ± 1.58	**13.48 ± 1.49**		14.96 ± 1.63	•	13.56 ± 1.56	
Waveform (2)	3	5000	0	40	13.36 ± 1.59	**13.31 ± 1.66**		14.69 ± 1.66	•	13.36 ± 1.64	
Wholesale-c	2	440	1	7	8.52 ± 3.80	8.39 ± 3.99		**8.08 ± 4.09**		8.56 ± 4.12	
Wholesale-r	3	440	0	7	30.56 ± 6.18	28.97 ± 6.27	○	28.95 ± 6.19	○	**28.18 ± 6.19**	○
Wisconsin cancer	2	699	0	9	3.29 ± 2.10	3.21 ± 1.96		3.13 ± 1.98		**2.81 ± 1.82**	○
Yeast	10	1484	0	8	38.15 ± 4.04	37.28 ± 3.80	○	37.72 ± 4.07	○	**37.25 ± 4.10**	○
Zoo	7	101	0	16	**3.27 ± 5.73**	3.73 ± 6.19		4.80 ± 6.42	•	5.67 ± 6.70	•

Table 1: Dataset summaries and mean and standard deviations of percentage of test cases misclassified. Method with best accuracy is shown in bold. • and ○ indicate that CCFs were significantly better and worse respectively at the 1% level of a Wilcoxon signed rank test. K = number of classes, N = number of data points, D^c = number of non-ordinal features and D^r = the number of binary or ordinal features. The *ORL* dataset could not be run in reasonable time for rotation forest.

	CCF	CCF-Bag	RF	Rotation Forest
CCF	-	2	2	7
CCF-Bag	**18**	-	1	14
RF	**26**	27	-	25
Rotation Forest	**13**	10	4	-

Table 2: Number of victories column vs row at 1% significance level of Wilcoxon signed rank test

signed rank test. This shows that CCFs performed excellently. The domination of CCFs over CCF-Bag highlights the improvement from the projection bootstrap while the good performance on a large variety of datasets demonstrates the robustness and wide ranging applicability of CCFs.

As shown by Menze et al [16], RFs often struggle on data with highly correlated features. CCA naturally incorporates information about feature correlations, therefore CCFs do not suffer the same issues as demonstrated by their superior performance on the highly correlated *hill valley* datasets. Although rotation forests can account for global correlations, each tree is orthogonal in the transformed space and therefore they cannot adapt to local correlations in the way which CCFs can. This

is demonstrated by the spirals dataset, where rotation forests gave, on average, nearly four times as many misclassifications as CCFs. Further discussion is provided in the supplementary material.

For the *wholesale-r* dataset none of the methods improved on simply predicting the most populous class (this gives a misclassification rate of 28.18%), with CCFs giving significantly worse performance than this (30.56%). Therefore care may be required for datasets with no discernible structure.

4.1 Computational Complexity

Although the exact computational complexity of the method is non-trivial, in general the cost of CCA is not significantly greater than that of the partitioning process, provided λ is set as a logarithmic factor of number of features. In particular both increase linearly with the number of data points. Direct time comparison with RF is impractical due to the differing software packages. However, as we found that the proportion of training time spent on CCA for each dataset was on average around $1/6$ and never more than $1/4$ and the average tree size was similar to RFs, we assert that practically training of CCFs has similar computational cost to RFs. Further, if $K < \lambda$ the number of dimensions searched over for splitting is less than in RFs and therefore CCFs may in fact be faster.

Rotation forest training requires each tree to search the full set of features and therefore is exponentially more expensive in the number of features than RFs and CCFs. This makes their implementation impractical for datasets with more than a modest number of features, as demonstrated by the *ORL* dataset for which we were unable to train a rotation forest, experiencing both memory issues and a train time that was more than all of the other datasets combined.

4.2 Relationship with LDA

As shown by, for example, De la Torre [24], CCA with 1-of-K class encoding is exactly equivalent to Rao's [25] extension of FLDA to the multi-class case. We have presented our work as the former because we believe that considering uncorrelated sets of co-projections which maximise correlation between features and classes labels is an intuitive way to understand the multiple solutions from a multi-class FLDA and because of the relative ease of carrying out CCA in a numerically stable fashion. Since our development of CCFs, we became aware of recent similar work by Zhang and Suganthan [26] whose LDA-RF method is similar to CCF-Bag. Aside from the large improvements we have shown by using projection bootstrapping instead of bagging, we believe our method has a number of other advantages over their method. Firstly their method of dealing with categorical features by carrying out LDA on all possible permutations can lead to combinatorial computational complexity and therefore will be intractable when the number of categorical features is large. We found our choice of λ lead to better results in addition to its preferable computational complexity for large data. Finally, their method of carrying out LDA by finding the generalized eigenvectors of the between-class and within-class scatter matrices can easily become numerically unstable.

5 Conclusions and Future Work

We have introduced canonical correlation forests, a new decision tree ensemble learning scheme that creates a new performance benchmark for out-of-box tree ensemble classifiers, despite being significantly less computationally expensive than some of the previously best alternatives. This performance is based on two core innovations: the use of a numerically stable CCA for generating projections along which the trees split and a novel new alternative to bagging, the projection bootstrap, which retains the full dataset for split selection in the projected space.

As it is the aim of the CCF method to operate in a parameter free context, we have made no attempt to adjust any algorithm operation between datasets and have made little investigation as to whether our choices of default parametrisation are in fact the best. The method is fully compatible with alternative meta schemes such as boosting or training each tree on a rotated coordinate system, such as done for rotation forests. Zhang and Suganthan's [26] RF-ensemble extension could also easily be applied to CCFs and might offer further improvement. Weighted voting schemes for trees such as Bayesian model combination [27] and similarity based schemes [28] would directly apply to CCFs and may also offer improvement at the cost of additional complexity, computational time and in some cases the need for additional parameter adjustment. All the concepts introduced should also directly apply to random forest regression models, providing a natural extension.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[3] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[4] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[5] Harold Hotelling. Relations between two sets of variates. *Biometrika*, pages 321–377, 1936.

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[7] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[8] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[9] Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[10] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[11] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *arXiv preprint cs/9408103*, 1994.

[12] João Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

[13] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica sinica*, 7(4):815–840, 1997.

[14] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[15] Tracy D Lemmond, Andrew O Hatch, Barry Y Chen, David Knapp, Lawrence Hiller, Marshall Mugge, and William G Hanley. Discriminant random forests. In *DMIN*, pages 55–61, 2008.

[16] Bjoern H Menze, B Michael Kelm, Daniel N Splitthoff, Ullrich Koethe, and Fred A Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[17] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[18] Magnus Borga. Canonical correlation: a tutorial. *On line tutorial http://people.imt.liu.se/magnus/cca*, 4, 2001.

[19] Åke Björck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

[20] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[21] M. Lichman. Uci machine learning repository, 2013.

[22] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.

[23] Jinyan Li and Huiqing Liu. Kent ridge bio-medical data set repository. *Institute for Infocomm Research. http://sdmc. lit. org. sg/GEDatasets/Datasets. html*, 2002.

[24] Fernando De la Torre. A least-squares framework for component analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(6):1041–1055, 2012.

[25] C Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

[26] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Random forests with ensemble of feature spaces. *Pattern Recognition*, 47(10):3429–3437, 2014.

[27] Kristine Monteith, James L Carroll, Kevin Seppi, and Tony Martinez. Turning bayesian model averaging into bayesian model combination. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2657–2663. IEEE, 2011.

[28] Marko Robnik-Šikonja. Improving random forests. In *Machine Learning: ECML 2004*, pages 359–370. Springer, 2004.

A Canonical Correlation Analysis

A.1 Overview

Let us consider applying a canonical correlation analysis (CCA) [5] between the arbitrary arrays $W \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{n\times k}$ and let $a \in \mathbb{R}^{d\times 1}$, $\|a\|_2 = 1$ and $b \in \mathbb{R}^{k\times 1}$, $\|b\|_2 = 1$ be arbitrary vectors on the $(d-1)$-hypersphere and $(k-1)$-hypersphere respectively. Denote the set of solutions for the canonical coefficients as $\{A_\nu, B_\nu\}_{\nu\in\{1,\ldots,\nu_{max}\}}$ where each A_ν and B_ν are in the space of a and b respectively and $\nu_{\max} = \min(\operatorname{rank}(W), \operatorname{rank}(V))$. Note if both ranks are zero then $\nu_{\max} = 1$. The first pair of canonical coefficients are given by

$$\{A_1, B_1\} = \underset{a,b}{\operatorname{argmax}} \left(\operatorname{corr}\left(Wa, Vb\right)\right) \tag{6}$$

and the corresponding canonical correlation components are given by WA_1 and VB_1. The second pair of canonical coefficients, $\{A_1, B_1\}$ is given by the solution to (6) under the additional constraints that new components are uncorrelated with the previous components:

$$\left(WA_1\right)^T WA_2 = 0 \quad \text{and} \quad \left(VB_1\right)^T VB_2 = 0. \tag{7}$$

This process continues up to ν_{max} times with all new components uncorrelated with the previous components, to produce a full set of canonical coefficients. Therefore a CCA will provide a set of independent pairs of vectors A and B which give the pairs of projections that maximum correlation between two matrices. Note that CCA is a co-ordinate free process that is unaffected by rotation, translation or global scaling of the input vectors W and V.

As shown by, for example, Borga [18], the solution of CCA has a closed form. Namely the coefficients satisfy the eigenvector problems

$$\begin{aligned} \Sigma_{WW}^{-1}\Sigma_{WV}\Sigma_{VV}^{-1}\Sigma_{VW}A_\nu &= \rho_\nu^2 A_\nu \\ \Sigma_{VV}^{-1}\Sigma_{VW}\Sigma_{WW}^{-1}\Sigma_{WV}B_\nu &= \rho_\nu^2 B_\nu \end{aligned} \tag{8}$$

where Σ_{WW}, Σ_{VV} and $\Sigma_{WV} = \Sigma_{VW}^T$ are the covariance of W, covariance of V and cross covariance of W and V respectively. The common eigenvalues, which correspond to the squares of the canonical correlations ρ_ν, make the pairing between the coefficients of W and V apparent and the order of the coefficients is found by the corresponding decreasing order of the canonical correlations, $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_{\nu_{\max}}$.

A.2 Numerically Stable Solution

Algorithm 3 Numerically Stable CCA

Inputs: $w \in \mathbb{R}^{N\times D}$, $v \in \mathbb{R}^{N\times K}$, $\varepsilon \in [0^+, 1^-]$
$\mu^w = \frac{1}{N}\sum_{n=1:N} w_{(n,:)}, \quad \mu^v = \frac{1}{N}\sum_{n=1:N} v_{(n,:)},$
$w_{(:,d)} \leftarrow w_{(:,d)} - \mu^w_{(d)} \quad \forall d \in \{1,\ldots,D\}, \quad v_{(:,k)} \leftarrow v_{(:,k)} - \mu^v_{(k)} \quad \forall k \in \{1,\ldots,K\}$
$[q^w, r^w, p^w] = \mathrm{QR}(w), \quad [q^v, r^v, p^v] = \mathrm{QR}(v)$
$\zeta^w = \max\left\{i : \left|r^w_{(i,i)}\right| < \varepsilon \left|r^w_{(1,1)}\right|\right\}, \quad \zeta^v = \max\left\{i : \left|r^v_{(i,i)}\right| < \varepsilon \left|r^v_{(1,1)}\right|\right\}$
$q^w \leftarrow q^w_{(:,1:\zeta^w)}, \quad r^w \leftarrow r^w_{(1:\zeta^w,1:\zeta^w)}, \quad q^v \leftarrow q^v_{(:,1:\zeta^v)}, \quad r^v \leftarrow r^v_{(1:\zeta^v,1:\zeta^v)}$
$\nu_{\max} = \min(\zeta^w, \zeta^v)$
$[u, \Omega, z] = \mathrm{SVD}((q^w)^T q^v)$
$u \leftarrow u_{(:,1:\nu_{\max})}, \quad z \leftarrow z_{(:,1:\nu_{\max})}$
$A = (r^w)^{-1} u, \quad B = (r^v)^{-1} z, \quad \rho = \mathrm{DIAG}(\Omega)$
$A_{(p^w,:)} \leftarrow \left[A^T, \mathbf{0}\right]^T, \quad B_{(p^v,:)} \leftarrow \left[B^T, \mathbf{0}\right]^T$
return A, B, ρ

Algorithm 3 outlines the numerical stable method used to carry out the CCA. In this algorithm then the function $[q, r, p] = \mathrm{QR}(\alpha)$ refers to a QR decomposition with pivoting such that $qr = \alpha(:, p)$ where q is an orthogonal matrix, r is upper triangular matrix and p is a column ordering defined implicitly such that $|r(i,i)| > |r(j,j)| \ \forall i < j$. The $[u, \Omega, z] = \mathrm{SVD}(\alpha)$ function is a SVD such

that $u\Omega z^T = \alpha$ where u and z are unitary matrix and Ω is diagonal matrix of singular values, with the ordering defined such that $\Omega(i,i) > \Omega(j,j)\ \forall i < j$.

The core idea of the algorithm is that reducing q^w, q^v, r^w and r^v such that r^w and r^v is full rank, ensures that r^w and r^v are invertible. Further as the r matrices are upper triangular, the coefficient calculation of line 10 can be simply calculated by back substitution without the need for inversion.

B Additional Experiments and Discussion

B.1 Inverted Cross Validation

To investigate the performance of CCFs on datasets containing few data points relative to the complexity of the underlying structure, we carried out inverted cross validations, training on one fold and testing on the other nine. 15 such tests were carried out, using the same folds as the original cross validation. Table 3 gives the results on each of the individual datasets and Table 4 gives a summary of the significant victories and losses.

Data set	CCF	CCF-Bag		RF		Rotation Forest	
Balance scale	14.86 ± 2.54	**14.08 ± 2.59**	○	21.02 ± 2.74	●	15.27 ± 2.52	●
Banknote	**0.64 ± 0.57**	0.78 ± 0.58	●	3.96 ± 1.66	●	0.89 ± 0.70	●
Breast tissue	**50.51 ± 8.19**	54.76 ± 8.87	●	52.06 ± 7.72	●	51.03 ± 8.10	
Climate crashes	7.23 ± 0.65	7.22 ± 0.54		**7.22 ± 0.51**		7.22 ± 0.79	
Fertility	18.97 ± 8.16	16.76 ± 7.20	○	**14.20 ± 5.36**	○	15.38 ± 5.92	○
Heart-SPECT	20.63 ± 3.08	**20.04 ± 2.83**	○	20.76 ± 3.06		20.09 ± 3.22	○
Heart-SPECTF	21.11 ± 1.87	21.38 ± 2.56		**20.79 ± 1.93**	○	20.93 ± 2.05	○
Hill valley	**0.14 ± 0.39**	0.17 ± 0.45		48.34 ± 1.75	●	7.18 ± 1.77	●
Hill valley noisy	**20.37 ± 4.07**	21.63 ± 4.57	●	49.27 ± 1.48	●	22.22 ± 3.50	●
ILPD	30.65 ± 1.79	30.29 ± 1.62	○	30.68 ± 1.71		**30.14 ± 1.52**	○
Ionosphere	**11.90 ± 4.05**	16.78 ± 4.82	●	13.90 ± 4.44	●	12.95 ± 4.28	●
Iris	5.93 ± 3.83	**5.65 ± 4.13**		7.59 ± 4.31	●	9.74 ± 5.64	●
Landsat satellite	11.58 ± 0.41	12.19 ± 0.42	●	12.07 ± 0.49	●	**11.41 ± 0.45**	○
Letter	**10.14 ± 0.47**	11.22 ± 0.48	●	13.46 ± 0.57	●	11.14 ± 0.48	●
Libras	**49.86 ± 4.83**	52.30 ± 4.66	●	61.57 ± 4.31	●	55.27 ± 4.82	●
MAGIC	**13.46 ± 0.22**	13.55 ± 0.23	●	14.03 ± 0.28	●	14.19 ± 0.29	●
Nursery	3.15 ± 0.34	3.67 ± 0.36	●	3.97 ± 0.43	●	**2.89 ± 0.39**	○
ORL	**54.95 ± 3.80**	61.22 ± 3.74	●	60.35 ± 4.04	●	-	
Optical digits	**3.54 ± 0.38**	3.93 ± 0.38	●	4.79 ± 0.47	●	3.91 ± 0.42	●
Parkinsons	**17.06 ± 4.05**	18.24 ± 4.62	●	18.75 ± 4.02	●	18.55 ± 4.43	●
Pen digits	**1.43 ± 0.22**	1.67 ± 0.24	●	2.97 ± 0.35	●	1.84 ± 0.27	●
Polya	23.80 ± 0.51	23.84 ± 0.50		24.16 ± 0.61	●	**23.08 ± 0.50**	○
Seeds	8.84 ± 3.14	**8.53 ± 3.21**	○	13.34 ± 3.21	●	11.71 ± 3.70	●
Skin seg	**0.06 ± 0.01**	**0.06 ± 0.01**		0.13 ± 0.02	●	0.10 ± 0.01	●
Soybean	**18.96 ± 3.95**	20.23 ± 4.06	●	21.99 ± 4.00	●	21.22 ± 4.73	●
Spirals	**0.68 ± 0.16**	**0.68 ± 0.16**		3.29 ± 0.49	●	3.80 ± 0.57	●
Splice	7.53 ± 1.75	9.06 ± 2.12	●	**4.81 ± 0.68**	○	6.72 ± 1.11	○
Vehicle	**26.02 ± 2.14**	26.78 ± 2.26	●	32.76 ± 2.66	●	27.79 ± 2.31	●
Vowel-c	**40.71 ± 3.17**	41.46 ± 3.44	●	46.42 ± 3.00	●	41.63 ± 2.87	●
Vowel-n	**34.15 ± 2.99**	35.37 ± 2.92	●	41.52 ± 2.94	●	35.82 ± 3.16	●
Waveform (1)	14.76 ± 0.41	**14.66 ± 0.41**	○	16.52 ± 0.53	●	14.91 ± 0.42	●
Waveform (2)	14.83 ± 0.48	**14.70 ± 0.47**	○	16.24 ± 0.49	●	14.83 ± 0.49	
Wholesale-c	11.41 ± 2.14	11.16 ± 2.13	○	10.82 ± 2.00	○	**10.58 ± 1.84**	○
Wholesale-r	35.64 ± 4.09	33.08 ± 3.81	○	33.38 ± 3.66	○	**31.23 ± 3.59**	○
Wisconsin cancer	4.12 ± 1.02	4.10 ± 1.03		4.31 ± 0.98	●	**3.53 ± 0.69**	○
Yeast	46.12 ± 1.86	45.37 ± 1.83	○	45.98 ± 1.91		**45.32 ± 1.82**	○
Zoo	**23.43 ± 10.80**	24.12 ± 11.15	●	25.36 ± 11.26	●	24.10 ± 10.66	

Table 3: Mean and standard deviations of percentage of test cases misclassified for inverted cross validation. Method with best accuracy is shown in bold. ● and ○ indicate that CCFs were significantly better and worse respectively at the 1% level of a Wilcoxon signed rank test.

The performance of CCFs relative to rotation forests was similar to the standard cross validation case. CCF-Bag performed relatively more favourably compared with CCFs and rotation forests, but was still comprehensively outperformed by the former. The relative performance of RFs to the other methods improved slightly but it was still the worst performing method by a large margin.

	CCF	CCF-Bag	RF	Rotation Forest
CCF	-	10	5	12
CCF-Bag	**19**	-	8	12
RF	**28**	26	-	26
Rotation Forest	**20**	15	5	-

Table 4: Number of victories column vs row at 1% significance level of Wilcoxon signed rank test for inverted cross fold validation

A further point of note is that on the *fertility* and *wholesale-r* datasets all the methods performed significantly worse than just predicting the most populous class, giving misclassification rates of 12% and 28.2% respectively with CCFs performing particularly poorly. This suggests that some sort of regularization, for example pruning of trees or incorporating information about the overall class ratios, might be beneficial for some datasets.

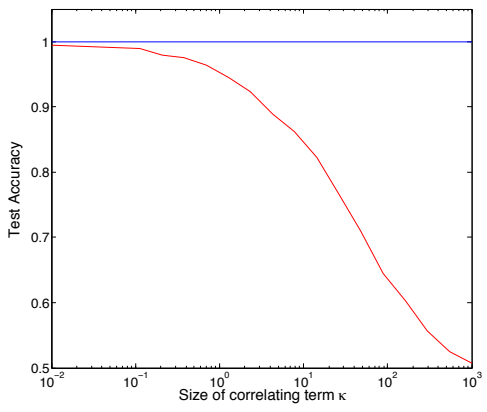
B.2 Effect of Correlation

As shown by Menze et al [16], RFs often struggle on data with highly correlated features. As CCA naturally incorporates information about correlations, we expect CCFs to be more robust. To investigate this formally we used the following method of artificially correlating the data:

1. Process categorical features and convert to z-scores as described in section 2.4.
2. Create a new random feature $X_{(D+1,n)} \sim \mathcal{N}(0, \kappa)\,,\ \forall n = 1, \ldots, N$. Note that κ will control the degree of correlation added.
3. To each of the existing features, randomly either add or subtract the new feature: $X_{(:,d)} \leftarrow X_{(:,d)} + \zeta_d X_{(:,D+1)},\ \forall d = 1, \ldots, D$ where each ζ_d is an i.i.d. draw randomly and independently sampled from $\{-1, 1\}$ with probability $\{1/2, 1/2\}$.
4. Train the selected algorithm using the full feature set $X_{(1:D+1,:)}$

As shown in figure 2, this transformation has no effect on the accuracy of CCFs, whereas the accuracy of RF slowly decreases as the correlation increases until it reaches an accuracy equivalent to random prediction. The use of PCA means rotation forests exhibit similar robustness to global correlations as CCFs. Note that for these tests we take the tolerance parameter of the CCA, ε, to be 10^{-12} on the basis that for large κ the added terms can dwarf the original data and cause canonical components corresponding to original data to be eliminated in the rank reduction.

(a) Banknote



(b) Iris

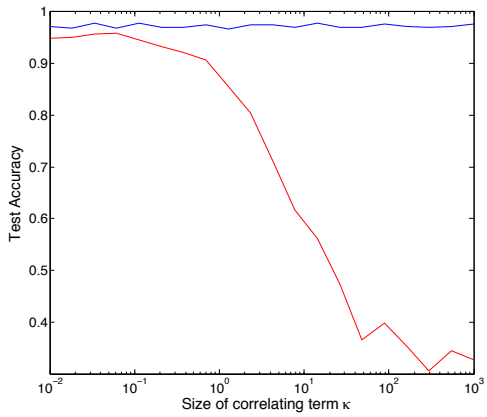


Figure 2: Accuracy of CCFs (blue) and RFs (red) on artificially correlated data. For each method, 5 separate 10-fold cross validations were run and the mean test accuracy across the 50 tests reported.

We postulate that CCFs are better than rotation forests at incorporating class dependent correlations because the rotation step in a rotation forests does not incorporate any class information other than in the random elimination of classes. Further, individual trees in a rotation forest are orthogonal and therefore cannot incorporate spatial variation in correlation. The self similar nature of the growth algorithm for CCTs on the other hand, means that the local correlations of a partition can be incorporated as naturally as the global correlations. To investigate these suggestions formally, we tested performance on compound datasets in which localized and class dependent correlations had been artificially added as described below:

1. Create a replica of a dataset, assigning classes such that if a point has class k in the original dataset, it has class $k + K$ in the replica.
2. Independently apply the artificial correlations discussed earlier in this section to both the original data and the replica.
3. Add a scalar constant, β, to every feature of every data point in the replica dataset to separate the replica points from the originals.
4. Carry out the selected algorithm using the compound dataset formed by the union of the original data and the replica.

We tested the effect of this transformation on the datasets where CCFs had lost significantly to rotation forests in the original tests so that any victories should be a direct consequence of introducing the correlations (though there may be small effects from the increased number of classes). We performed a single crossfold validation, taking $\beta = 2000$ and $\kappa = 100$. The results, given in Table 5, show that CCFs only experienced a small loss of accuracy on all of the datasets, whereas there was a large loss of accuracy in all cases for RFs and for some of the datasets for rotation forests.

Data set	CCF	RF	Rotation Forest
Balance scale	**8.24 ± 2.09**	48.40 ± 4.75 •	20.08 ± 3.51 •
Landsat satellite	**9.09 ± 0.95**	32.82 ± 1.36 •	11.66 ± 1.02 •
Polya	21.75 ± 0.62	32.40 ± 1.32 •	**21.06 ± 0.50** ○
Vowel-n	**3.99 ± 0.69**	86.11 ± 2.12 •	21.36 ± 3.59 •
Wholesale-r	29.89 ± 3.05	37.73 ± 4.73 •	**28.18 ± 4.57**
Wisconsin cancer	**2.71 ± 1.68**	30.36 ± 3.00 •	3.07 ± 1.63
Yeast	**38.52 ± 2.23**	70.10 ± 2.14 •	55.39 ± 2.39 •

Table 5: Mean and standard deviations of percentage of test cases misclassified for compound datasets. Method with best accuracy is shown in bold. • and ○ indicate that CCFs were significantly better and worse respectively at the 1% level of a Wilcoxon signed rank test.