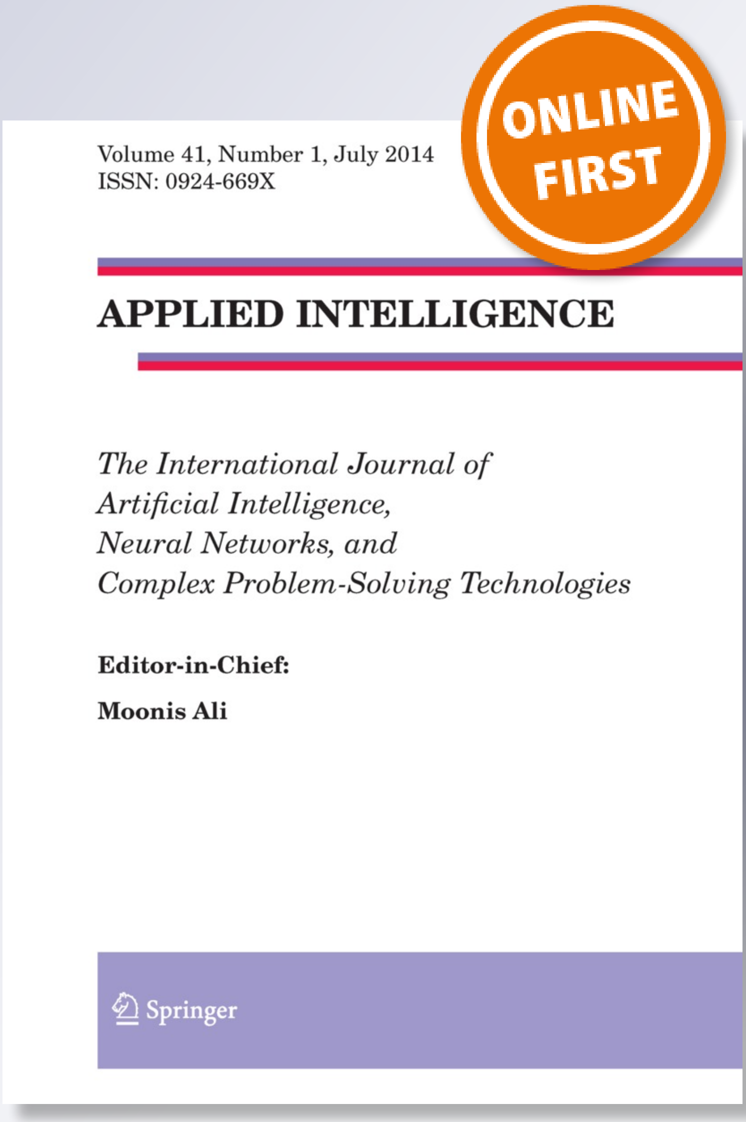
Decision tree ensembles based on kernel features

Amir Ahmad

Applied Intelligence
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies

ISSN 0924-669X

Appl Intell
DOI 10.1007/s10489-014-0575-4


Volume 41, Number 1, July 2014
ISSN: 0924-669X
ONLINE FIRST
APPLIED INTELLIGENCE
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies
Editor-in-Chief:
Moonis Ali
Springer

Appl Intell
DOI 10.1007/s10489-014-0575-4

Decision tree ensembles based on kernel features

Amir Ahmad

Abstract A classifier ensemble is a set of classifiers whose individual decisions are combined to classify new examples. Classifiers, which can represent complex decision boundaries are accurate. Kernel functions can also represent complex decision boundaries. In this paper, we study the usefulness of kernel features for decision tree ensembles as they can improve the representational power of individual classifiers. We first propose decision tree ensembles based on kernel features and found that the performance of these ensembles is strongly dependent on the kernel parameters; the selected kernel and the dimension of the kernel feature space. To overcome this problem, we present another approach to create ensembles that combines the existing ensemble methods with the kernel machine philosophy. In this approach, kernel features are created and concatenated with the original features. The classifiers of an ensemble are trained on these extended feature spaces. Experimental results suggest that the approach is quite robust to the selection of parameters. Experiments also show that different ensemble methods (Random Subspace, Bagging, Adaboost.M1 and Random Forests) can be improved by using this approach.

Keywords Classifier ensembles · Decision trees · Kernel features · Random subspaces · Bagging · AdaBoost.M1 · Random forests.

A. Ahmad (✉)
Faculty of Computing and Information Technology,
King Abdulaziz University, Rabigh, Saudi Arabia
e-mail: amirahmad01@gmail.com

1 Introduction

Classifier ensembles [12, 18, 32] and support vector machines [10, 34] are two important research fields in machine learning because of their high classification accuracy. Classifier ensembles are a combination of multiple base models [12, 18, 24, 32]; the final classification depends on the combined outputs of individual models. Classifier ensembles are successful because they combine the strengths of individual classifiers. The effectiveness of an ensemble method depends on the properties (accurate and diverse) of its base classifier [12, 21]. Some classifiers like decision trees and neural networks [12, 21] are unstable as a small change in training data lead to significantly different classifier structures which produce different outputs. Ensembles of these classifiers are very effective and generally perform better than single classifiers. Bagging [7], AdaBoost [14] and Random Forests [8] are popular ensemble methods.

Decision trees are built by using top-down induction methods. In a decision tree, for a continuous feature, each node partitions the available data points into two or more sets. Univariate decision trees are the most popular decision trees because of their low computational complexity [9, 25]. In univariate decision trees like C4.5 [25] the decision at an internal node uses only one feature. Any decision surface that is not perpendicular to a feature axis is approximated by these decision trees. Very large decision trees can approximate these boundaries well. However, to grow a very large decision tree, a sufficiently large dataset is needed. The lack of a large dataset often restricts the representational power of a decision tree [12]. Ensembles of diverse decision trees solve the representational problem associated with a single univariate decision tree as combined results of decision trees produce a good approximation of a non-orthogonal concept.

Published online: 10 August 2014

Hence, decision tree ensembles can learn complex decision boundaries.

Support vector machines [10, 34] (SVM) are successful because they use the expressive power of kernel functions. SVM use "kernel-trick" which allows kernelized algorithms to operate in high dimensions without incurring a corresponding cost. A kernel function transforms a nonlinear classification problem into a linear classification problem in the new high dimensional space. With appropriate kernel functions, SVM can learn complex decision boundaries. Classifier ensemble methods are quite robust to the choice of the parameters [22, 38] whereas the performance of SVM and other kernel based methods like Kernel-PCA [29] are sensitive to the choice of the kernel function and other parameters [13, 34, 38]. Various studies have been carried out to compare the performance of various ensemble methods and SVM [11, 30, 31]. Caruana and Niculescu-Mizil [11] conclude that boosting trees and Random Forests generally perform better than SVM. However, Statnikov et al. [30] shows that SVM performs better than Random Forests for microarray-based cancer classification. As both the machine learning approaches (ensemble methods and SVM) are useful, attempts have been made to combine the two approaches. One such hybrid approach is SVM ensembles [20, 23, 33, 35, 36]. However, SVM ensembles are not very popular. SVM classifiers are stable [33], hence they don't create diverse classifiers. As the base classifier (SVM classifiers) is sensitive to the selected parameters, SVM ensembles are also sensitive to the selection of parameters. Contrary to decision tree ensembles, generally SVM ensembles don't show large improvements over single SVM classifiers [36].

Therefore, there is a need for an ensemble method that has the robustness of popular ensemble methods (i.e. it should not be very sensitive to the selection of parameters) and at the same time (like SVM) can also utilize the expressive power of a kernel function, if an appropriate kernel function is selected. In other words, the ensemble method should perform very well with appropriate kernel functions, however, the performance should not degrade sharply with inappropriate kernel functions. In this paper, we propose different approaches to create ensembles of decision trees such that each member of an ensemble can use the expressive power of the kernel functions. This approach uses the kernel features created by a mapping proposed by Balcan et al. [5]. This mapping generates a set of features such that if a dataset is linearly separable with a margin under the kernel function, then it is approximately separable in this new kernel feature space. Like other kernel methods [29, 34], the quality of kernel features is dependent on the selected kernel. It is difficult to calculate the required dimension of the kernel feature space (the number of kernel features) that preserves all the information of the dataset because it is data dependent. Balcan et al. [5] propose experiments with 10 and 20 features, however, no justification is given for the selection of these numbers. Rwebangira [28] observe by using different types of classifiers that if kernel features are concatenated with the original features, then the classifiers trained on these new datasets perform better than the classifiers trained on the original data or the classifiers trained on the data created by kernel features. These experiments suggest that classifiers can use the strengths of both kinds of features and results in the improved performance of classifiers [28]. Though, the effect of the kernel function and the dimension of the kernel feature space is not discussed in [28], we expect that when a classifier is trained on an extended feature space (original features + kernel features), the performance of classifier becomes robust to the selection of kernel parameters (the kernel function and the dimension of kernel feature space). The probable reason for this is that even if the kernel features are not informative (because of the inappropriate kernel) or the dimension of the kernel feature space is less than the desired dimension (so that all the information about the dataset is not preserved in the kernel feature space), the performance of classifier does not degrade because the classifier can use the original features.

We propose that this strategy of using an extended feature space (original features + kernel features) for training a classifier can improve existing ensemble methods because it can generate accurate and diverse decision trees (discussed in detail in Section 3).

1.1 Contributions

The main motivation for proposing the new ensemble methods is that we want to combine the robustness of the existing ensemble methods with the excellent expressive power of kernel functions. Following are the two main contributions of this paper:

(i) The mapping, proposed by Balcan et al. [5] to create kernel features, has a random element (it uses random data points from the training dataset). We propose that this property can be exploited to generate classifier ensembles. We propose a novel approach to create classifier ensembles by using kernel features [5] only and study its properties.

(ii) To circumvent the detrimental effect of choice of kernel parameters (the kernel function and the dimension of kernel feature space), we propose that each classifier of an ensemble can be trained on an extended feature space [28] that is created by concatenating kernel features with the original features. We show that existing ensemble methods can be improved with this strategy.

In Section 2, we present literature review, firstly, by giving a brief description on general classifier ensemble techniques, followed by a survey of ensemble methods with extended feature spaces. In Section 3, we describe our proposed method to create decision tree ensembles with kernel features only. The second proposed method that combines the existing ensemble methods with kernel features is also presented in Section 3. Section 4 presents results and discussion. Section 5 concludes the paper and describes directions for future work.

2 Related work

In this section, we first discuss the related research work on general ensemble methods, followed by a literature review of creating ensembles with extended feature spaces.

Several methods have been proposed to build decision tree ensembles. In these methods, randomization is introduced to build diverse decision trees. Bagging [7] and AdaBoost [14] introduce randomization by manipulating the training data supplied to each classifier. Bagging [7] generates different bootstrap training datasets from the original training dataset and uses each of them to train one of the classifiers in the ensemble. AdaBoost [14] generates a sequence of classifiers with different weight distribution over the training set. In each iteration, the learning algorithm is invoked to minimize the weighted error, and it returns a hypothesis. The weighted error of this hypothesis is computed and applied to update the weight on the training examples. The final classifier is constructed by a weighted vote of the individual classifiers. Each classifier is weighted according to its accuracy on the weighted training set that it has trained on. MultiBoost [37] combines the principle of Bagging with AdaBoost. Ho [19] proposes Random Subspaces (RS) that select random subsets of input features for training decision trees of an ensemble.

Random Forests are very popular decision tree ensembles [8]. They combine Bagging with RS. For each decision tree, a dataset is created by the bagging procedure. During the tree growing phase, at each node, k attributes are selected randomly and the node is split by the best attribute from these k attributes. For Random Forests [8], Breiman shows that the prediction error of a random forest, e, satisfies the inequality

$$e \leqslant \overline{\rho}(\frac{1-s^2}{s^2}) \tag{1}$$

where ρ is the mean correlation (it is related to diversity) between any two members of the forest (ensemble) and s, the mean strength (accuracy) of a typical member of the forest (ensemble). This inequality suggests that good random forests should have low ρ and high s. In Random Forests, Breiman adds an extra randomness step (RS) to Bagging to create more diverse decision trees. Diverse decision trees (small ρ) improve the classification accuracy. Breiman shows that Random Forests are quite competitive to AdaBoost. Random Forests can handle mislabeled data points better than AdaBoost. Due to their robustness, they are widely used. Rotation Forests [27] and Extremely Randomized Trees [16] are other popular ensemble methods.

2.1 Ensembles with extended feature spaces

Various researchers have proposed different techniques to develop new extended feature spaces for classifier ensembles. These new extended feature spaces are generated by concatenating original features with the newly generated features. RS [19] performs well when there is a certain redundancy in features. For datasets, where there is no redundancy, redundancy needs to be introduced artificially by concatenating new features that are cross-products of the original features to the original features and treating this as the dataset.

Fatih and Ersoy [2] extend the feature space by combining new features and original features and show that extended space versions of ensemble algorithms perform better than the original ensemble algorithms. They use many operators on random combinations of the original features to generate new features. They find experimentally that new features generated with a difference operator applied to random pairs of original features give best results. However, there is no theoretical justification for these types of new features. Ahmad and Brown [1] present an ensemble method that creates ensembles of linear multivariate decision trees [15]. They develop a Random Discretization method that creates random discretized features from original continuous features. Random projections are used to create new features that are linear combinations of original features. A new dataset is created by augmenting discretized features (created by using Random Discretization) with features created by using random projections. Each decision tree of an ensemble is trained on one dataset from the pool of those datasets by using a univariate decision tree algorithm. As these multivariate decision trees (because of features created by random projections) have more representational power than univariate decision trees, accurate decision trees in the ensemble are created. Diverse training datasets ensure diverse decision trees in an ensemble. This ensemble method matches or outperforms other popular ensemble methods.

Similar to these techniques, we propose that ensembles can be created by using extended feature spaces. However, in the proposed approach, we are using kernel features for creating extended feature spaces that have not been used before for ensemble methods.

3 Ensemble methods with kernel features

In this section, we first discuss the kernel features [5], and then we present decision tree ensemble methods created by using kernel features.

3.1 Kernel features

Kernel functions are widely used in many machine learning applications [10, 34]. A kernel function can be viewed as allowing one to implicitly map data into a high-dimensional feature space without a high computational cost. A kernel is a matrix containing similarity measures for a dataset [10, 34]. There are many possible nonlinear similarity measures, but in order to be mathematically tractable the kernel function has to satisfy Mercer conditions (continuous, symmetric, positive semi-definitive) [10, 34].

Many linear classification models can be transformed into nonlinear methods by kernelizing them [10, 34]. The kernel transformation applied in these kernelized methods acts as a data transformation in a preprocessing stage. The advantage of such a kernel method is that the nonlinear aspects of the problem are captured entirely in the kernel. In other words, a nonlinear problem is converted into a linear problem by this kernel approach, and then the linear model is applied on them. The use of a kernel function greatly increases the representational power of linear methods by nonlinearly transforming the data [10, 34]. However, not all classifiers are easily kernelizable [5]. To overcome this problem, Balcan et al. [5] proposed an alternative to "kernelizing" a learning algorithm: rather than modifying the algorithm to use kernel functions, one can instead construct a mapping into a low-dimensional space using the kernel function and the data distribution, and then run an unkernelized algorithm over examples drawn from the mapped distribution. The advantage of this method is that the classifiers that are not easily kernelizable can use the expressive power of kernel functions.

Arriaga and Vempala [3] suggest that for a kernel function $K(x_i, x_j) = \phi(x_i)\phi(x_j)$, if a data set D is such that the target function has margin γ in the ϕ-space, then a random linear projection of the ϕ-space down to a space of dimension $d = O\left(\frac{1}{\gamma^2} log \frac{1}{\delta\epsilon}\right)$ will, with probability at least 1 - δ, have a linear separator with error rate at most ϵ. In other words, for any kernel function K and margin γ, K can be considered as mapping the input space into an $\widetilde{O}(\frac{1}{\gamma^2})$-dimensional space. Using this argument, Balcan et al. [5] suggest a mapping that uses a given kernel function and random unlabeled data points. This mapping generates a set of features such that if a dataset is linearly separable with a margin under the kernel, then it is approximately separable in this new kernel feature space. Balcan et al. [5] propose that given a pairwise measure of similarity $K(x_i, x_j)$ between data objects x_i and x_j, one can construct features in a simple way by *randomly selecting* a set $x_1, x_2, ..., x_d$ of data points and then using $K(x, x_i)$ (similarity with the point x_i (i = 1 to d)) as the i^{th} feature of example x, where d depends on the margin. Balcan et al. shows that SVM with a nonlinear kernel performs similar to SVM with a linear kernel on a kernel features dataset created by using the proposed mapping [5]. This demonstrates that a nonlinear decision boundary in the original feature space is transformed to a linear decision boundary. We propose that these kernel features can be used to create ensembles or can be utilized to improve existing ensemble methods.

3.2 An ensemble method with kernel features only

As discussed in Section 3.1, the mapping [5] that generates kernel features uses random data points from the training dataset. Therefore, it generates different kernel features in different runs as different random points are selected. Datasets are created by using kernel features generated in different runs. In different runs, different datasets are created because of different kernel features. These diverse datasets can be used to create ensembles. Each decision tree in an ensemble learns on one dataset from the pool of different datasets created by this method. As these decision trees are trained on diverse datasets, we expect diverse trees. If an appropriate kernel is selected, the mapping generates informative features. Decision trees trained on these datasets with these informative kernel features will be accurate. In other words, by using the proposed technique accurate and diverse trees are generated which create an accurate ensemble. The algorithm is presented in Fig. 1. The performance of these ensembles is strongly dependent on the choice of the kernel function and the dimension of the kernel feature space (discussed in detail in Section 4). To overcome these problems, we propose that the kernel features are combined with existing ensemble methods.

3.3 Combining existing ensemble methods with kernel features

In this section, we discuss our proposed approach that combines kernel features with the existing ensemble methods. We propose that existing ensemble methods can be benefited by utilizing kernel features. In the proposed approach, we concatenate two feature spaces, the original feature

```
Input- Dataset $T$ with $m$ continuous features and $k$ classes $(c_1, c_2, .., c_k)$, $L$ the size of
the ensemble and a given kernel $K$.
Training Phase
for i=1...$L$ do
  Data Generation
  Create $d$ kernel features by using the mapping $KF_i$ (by selecting $d$ points randomly)
  proposed by Balcan et al. [5] and discussed in the Section 3. For a point $x$ the $d$ kernel
  features will be $K(x, x_1)$, $K(x, x_2)$,..,$K(x, x_d)$. Create a dataset $T_i$ with these features.
  Learning Phase
  Learn $D_i$ decision tree on $T_i$.
end for
Classification Phase
For a given data point $\mathbf{x}$
for i=1...$L$ do
  Convert $\mathbf{x}$ into a $d$ dimensional data point $\mathbf{x_i}$ by using the mapping $KF_i$.
  Let $p_{i,j}(x)$ be the probability for $\mathbf{x_i}$ by the decision tree $D_i$ to the hypothesis that $\mathbf{x}$
  comes from class $c_j$. Calculate $p_{i,j}(x)$ for all classes (j = 1..k).
end for
Calculate the confidence $C(j)$ for each class $c_j$ (j = 1..$k$) by the average contribution
method,
$C(j) = \frac{1}{L}\sum_{i=1}^{L} p_{i,j}(x)$.
The class with the largest confidence will be the class of $\mathbf{x}$.
```

Fig. 1 The algorithm for ensembles with kernel features only

space and the kernel feature space, to get extended feature spaces. Each decision tree of an ensemble is trained on an extended feature space. This strategy will allow existing ensemble methods to use the expressive power of kernels as each member of an ensemble is using kernel features along with original features. Our philosophy of using extended feature space is similar to the strategy proposed by Rwebangira [28]. However, Rwebangira [28] suggests an extended feature space for single classifier and we are using this strategy for creating ensembles. We exploit the randomness present in the mapping (used to create kernel features) to create diverse extended feature spaces. Whereas, Rwebangira [28] uses only one extended feature space. We will discuss the reasons why these ensembles are likely to be accurate. Different ensemble methods have been proposed (discussed in Section 1). Some of them are based on different mechanisms, like Bagging [7], AdaBoost.M1 [14] and RS [19] etc., whereas some of the ensemble methods combine methods that have different mechanisms, for example Random Forests [8] combines Bagging with Random Subspaces, Multiboosting [37] combines Bagging with AdaBoost and Rotation Forest [27] combines randomization in the feature space division with Bagging. The basic idea behind these “hybrid” ensemble techniques is that as the mechanisms differ for different ensemble methods, their combination may outperform either in isolation. As our proposed approach is also a ”hybrid” ensemble approach, we expect that it will perform similar to or better than its components (existing ensemble methods and ensembles with kernel features only).

Brieman develops an inequality (Eq. 1) for the prediction error of Random Forests [8]. However, this inequality is useful for all ensemble methods in which individual members are created by random mechanisms [38]. We will use this inequality to explain the effectiveness of the proposed ensemble method. As discussed in Section 1, this inequality suggests that a good ensemble method should have accurate (high s) and diverse (low correlation $(\overline{\rho})$) members in an ensemble. Hence, the mechanism we select to generate ensembles should produce accurate classifiers with the conditions that the correlation, between individual classifiers within ensembles, is minimum. We will argue that the proposed method generates accurate (high s) and diverse (low $(\overline{\rho})$)) decision trees.

(i) **Accurate classifiers -** As discussed in Section 1, univariate decision trees [9, 25] can learn decision boundaries orthogonal to the feature space. However, any decision surface that is not orthogonal to feature axes is approximated by these decision trees. In the proposed ensemble method, each univariate decision tree of an ensemble is trained on an extended feature space (original features + kernel features). Though a univariate decision tree is trained, orthogonal decision surface (due to the original features) and non-orthogonal decision surface (defined by the kernel features) are obtained. It is expected that these trees have more representational power as compared to decision trees trained on original features and decision trees trained only on kernel features as they have more decision surfaces. Hence, these trees are expected to be accurate [1].

If an appropriate kernel function is selected, in the kernel feature space a nonlinear decision boundary becomes a linear decision boundary with high probability [5]. In other words, decision trees are learning easier decision boundary (linear decision boundary) in the kernel features space than decision tree that are learning nonlinear decision boundary. Hence, decision trees trained on the extended feature space are likely to be more accurate than decision trees trained on the original features.

Decision trees do the feature selection at each node [25]. This property is useful for datasets whose original features are augmented with kernel features, as kernel features will only be selected at the higher levels of the trees when these features are better than or similar to original features for classification [9, 25]. In case, an appropriate kernel is not selected and non-informative kernel features are generated, and a univariate decision tree is trained on extended feature space, there is a strong probability, that the original features are selected at higher levels as they are more informative, whereas the kernel features will not be selected or will be selected at lower levels as they are less informative. In other words, these trees will be similar to trees trained on the original feature space. Hence, while these trees can improve their accuracy by using informative kernel features (generated due to appropriate kernel functions), they are not affected much due to non-informative kernel features (generated due to inappropriate kernel functions). This property of these trees makes them quite robust to the selection of inappropriate kernel functions.

The dimension of the new kernel feature space, such that almost all the information of the dataset is preserved in the kernel feature space, is an important parameter [5]. This parameter is dataset specific, and there is no method to calculate it [5]. If a decision tree is trained with kernel features only and the dimension of kernel space is less than the desired dimension, the accuracy of the decision tree suffers as there is information loss because of the inadequate dimension of the kernel feature space. However, when decision trees are trained on extended feature spaces, the kernel features act as the extra information about the dataset. Hence, even if the kernel feature space does not have all the information about the dataset, it helps in creating accurate decision trees.

(ii) **Diverse Classifiers -** Generation of diverse classifiers is the second condition of effective ensembles. In the proposed approach, the decision trees of existing ensemble methods are trained on extended feature spaces. A modified version of an ensemble method will be using the randomization mechanism of the original method. For example, in the modified version of Bagging, training data points for each decision tree will be generated by using the same method as Bagging, however, the feature space of these data points has been extended in the modified version. The mapping [5] to create kernel features has random elements, hence, it creates different kernel features in different runs. Therefore, different extended feature spaces (original features + kernel features), can be generated with these different kernel features. This extra randomness will reduce the correlation between individual trees and add more diversity in existing ensemble methods.

In this paper, we show the combination of four popular ensemble methods (RS, Bagging, AdaBoost.M1, Random Forests) with kernel features.

3.4 RS+Kernel features

As discussed in Section 1, in RS ensemble method, each member of an ensemble is trained on a different random subspace. We propose that for each run of the RS, d kernel features are created and combined with the random subspace for that run. A classifier is trained on this new dataset. Results of all runs are combined to get the final results. We present the proposed ensemble method *RS + Kernel features* in Fig. 2.

3.5 (Bagging, Adaboost.M1, Random Forests) + Kernel features

Bagging, AdaBoost.M1 and Random Forests are very popular ensemble methods. We study how these are affected by kernel features. We follow the similar methodology as suggested in Multiboosting [37]. In each run, d kernel features are created and concatenated with the original features. As in each run, different kernel features are created, hence different datasets are created by using these kernel features. On each of these datasets, an ensemble is trained. We do the same exercise for all the runs. Results of all runs are combined to get the final results. The algorithm is presented in Fig. 3.

```
**Input-** Original dataset $T$ with $m$ continuous features, $k$ classes $(c_1, c_2, .., c_k)$, the size of
an ensemble $L$ and a kernel $K$.
$L$ the size of the ensemble.
**Training Phase**
**for** i=1...$L$ **do**
  **Data Generation**
  1- Create $d$ kernel features by using the mapping $KF_i$ (by selecting $d$ points randomly)
  proposed by Balcan et al. [5] and discussed in the Section 3. For a point $x$ the $d$ kernel
  features will be $K(x, x_1)$, $K(x, x_2)$,..,$K(x, x_d)$.
  2- Use Random Subspaces $(RS_i)$ to create a dataset $S_i$.
  3- Concatenate $S_i$ and $d$ kernel features to get the dataset $T_i$.
  **Learning Phase**
  Treating dataset $T_i$ as continuous, learn $D_i$ decision tree on it.
**end for**
**Classification Phase**
For a given data point $\mathbf{x}$
**for** i=1...$L$ **do**
  1- Convert $\mathbf{x}$ to $\mathbf{x_i}$ by using Random Subspaces $(RS_i)$ and the kernel mapping $(KF_i)$.
  2- Let $p_{i,j}(x)$ be the probability for $\mathbf{x_i}$ by the decision tree $D_i$ to the hypothesis that
  $\mathbf{x}$ comes from class $c_j$. Calculate $p_{i,j}(x)$ for all classes (j = 1..$k$).
**end for**
Calculate the confidence $C(j)$ for each class $c_j$ (j = 1..$k$) by the average contribution
method, $C(j) = \frac{1}{L}\sum_{i=1}^{L} p_{i,j}(x)$.
Class with the largest confidence will be the class of $\mathbf{x}$.
```

Fig. 2 The algorithm for *RS + Kernel features* ensembles

3.6 Complexity

We propose that the existing ensemble methods can be benefited by using kernel features. However, creating new features adds extra computational cost. The extended feature space has more dimension than the original feature space. Hence, the tree learning phase with the extended feature space may need more computational resources than the tree training phase with the original feature space.

4 Experiments

We first carried out experiments to analyze the performance of the proposed ensemble methods with extended feature spaces on synthetic datasets. Experiments were also carried on other popular datasets. Experiments were carried out by using WEKA software [17]. Following parameters were used in the experiments;

(i) **RS features size -** Ho [19] achieve good results for RS with half of the original features selected randomly in each run. Therefore, we also used the same value of original features for RS [19].
(ii) **Bagging, Adaboost.M1 and Random Forests -** The modules of WEKA for these ensembles were used in the experiments. All the default parameters were used in the experiments.
(iii) **The dimension of kernel feature space -** It is not possible to know in advance the required dimension (as it depends on the margin) of the new kernel feature space, such that all the information of the original data is preserved in the new kernel feature space. Balcan et. al. [5] carried out experiments with 10 and 20 kernel features. Hence, we used 10 kernel features in each run. As discussed in Algorithm 2 (Fig 2), in each run 10 training points were selected randomly and 10 kernel features were generated ($K(x, x_i)$, similarity with the point x_i, is the i^{th} feature of the data point x). We discuss the issue of choosing the number of kernel features in detail in Section 4.3.
(iv) **Classifier -**J48 decision trees with unpruned option were used (WEKA implementation of C4.5[25]) for the experiments.
(v) **The size of the ensembles -** The size of the ensembles was set to 200.
(vi) **(Bagging, Adaboost.M1, Random Forests) + kernel features ensembles -** In these methods, 14 ($200^{0.5} \approx 14$) runs were used. In each run, 14 ($200^{0.5} \approx 14$) trees were trained. In other words,

```
Input- Dataset $T$ with $m$ continuous features and $k$ classes $(c_1, c_2, .., c_k)$, $L$ the size of
the ensemble and a given kernel $K$. $M$ is the size of the individual ensemble.
Training Phase
for i=1...$\lfloor L/M \rfloor$ do
   Data Generation
   1- Create $d$ kernel features by using the mapping $KF_i$ (by selecting $d$ points randomly)
   proposed by Balcan et al. [5] and discussed in the section 2. For a point $x$ the $d$ kernel
   features will be $K(x, x_1)$, $K(x, x_2)$,..,$K(x, x_d)$.
   2- Concatenate $T$ and $d$ kernel features to get the $m + d$ dimensional dataset $T_i$.
   3- Learn $E_i$ an individual ensemble of size $M$ on $T_i$.
end for
Classification Phase
For a given data point $\mathbf{x}$
for i=1...$\lfloor L/M \rfloor$ do
   1- Convert $\mathbf{x}$ into a $m+d$ dimensional data point $\mathbf{x_i}$ by using the mapping $KF_i$ (original
   features + kernel features).
   2- Let $p_{i,j}(x)$ be the probability for $\mathbf{x_i}$ by the ensemble $E_i$ to the hypothesis that $\mathbf{x}$
   comes from class $c_j$. Calculate $p_{i,j}(x)$ for all classes (j = 1..k).
end for
Calculate the confidence $C(j)$ for each class $c_j$ (j = 1..$k$) by the average contribution
method,
$C(j) = \frac{1}{\lfloor L/M \rfloor} \sum_{i=1}^{L} p_{i,j}(x)$.
The class with the largest confidence will be the class of $\mathbf{x}$.
```

Fig. 3 The algorithm for *Bagging, Adaboost.M1, Random Forests) + Kernel features*

in each ensemble 196 trees were created. The size parameters were selected such that almost 200 trees were created in each ensemble.

(g-) **The final result -** Each member decision tree gives a classification probability for each class. These probabilities are added and the class with the highest summation is chosen as the class of the given data point. The framework of proposed ensembles with the parameters used in the experiments is given in Fig. 4.

(vii) **Statistical test -**Results were compared by using t-test with 95% confidence interval.

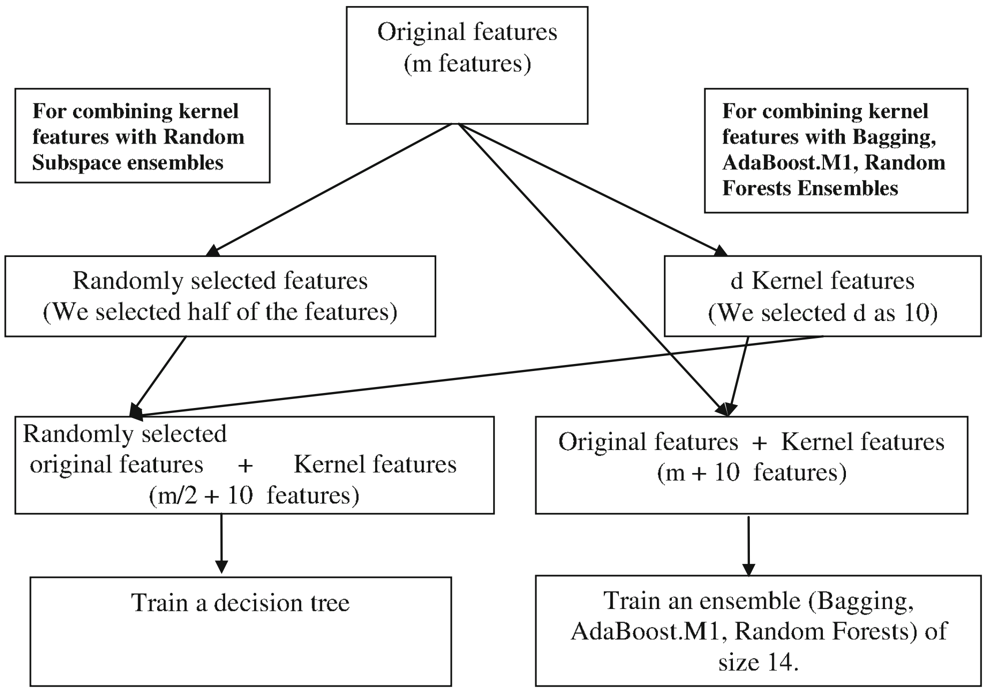

Original features
(m features)
For combining kernel features with Random Subspace ensembles
For combining kernel features with Bagging, AdaBoost.M1, Random Forests Ensembles
Randomly selected features
(We selected half of the features)
d Kernel features
(We selected d as 10)
Randomly selected original features + Kernel features
(m/2 + 10 features)
Original features + Kernel features
(m + 10 features)
Train a decision tree
Train an ensemble (Bagging, AdaBoost.M1, Random Forests) of size 14.

Fig. 4 The framework for training a classifier for the proposed ensemble methods in the experiments.

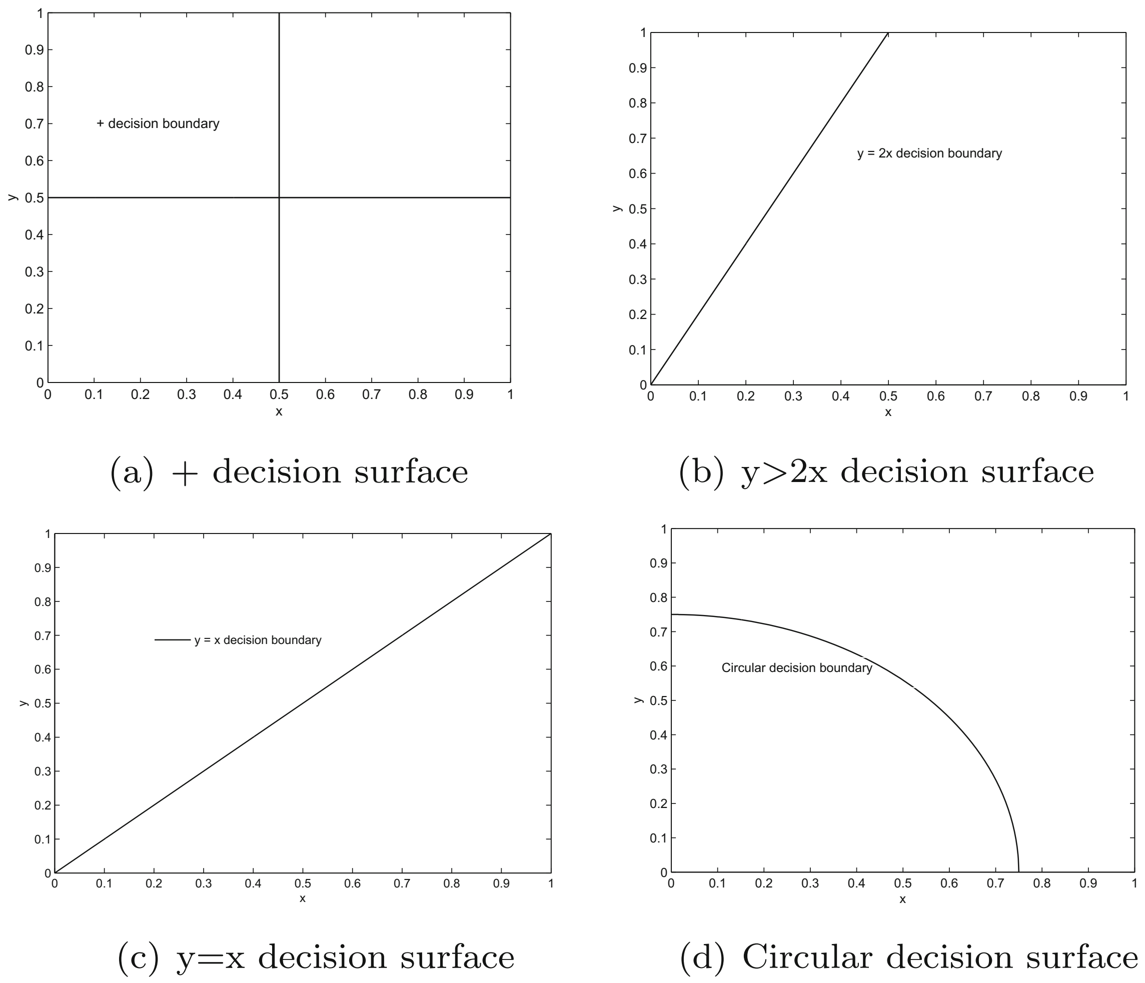


Fig. 5 Decision surfaces of synthetic datasets.

4.1 Experiments with synthetic datasets

We carried out experiments with synthetic datasets with different decision boundaries to understand which kinds of decision boundaries, decision tree ensembles with extended feature spaces are useful. These datasets have linear and nonlinear decision boundaries. We created following four two-dimensional datasets, with two features(x and y; data points were created with $0 <= x <= 1$ and $0 <= y <= 1$.

(i) $+$ **decision boundary -** This dataset has four classes. Points with x<=0.5 and y<=0.5 are placed in class 1. Class 2 has points with x>0.5 and y<=0.5. Class 3 has points with x<=0.5 and y>0.5. Points with x>0.5 and y>0.5 are placed in class 4. This dataset has an orthogonal decision boundary as shown in Fig. 5 a.

(ii) $y > 2x$ **decision boundary -**Data points with y>2x are labeled as one class and all other points are placed in the other class. It is a linear and non-orthogonal decision boundary. The decision boundary is given in Fig. 5 b.

(iii) $y = x$ **decision boundary -** All the points with y>x are placed in one class and all other points are placed in the other class. The dataset has a diagonal decision boundary. It is a linear and non-orthogonal decision boundary, however, it is more dissimilar to the orthogonal decision boundary than $y > 2x$ decision boundary is. The decision boundary is given in Fig. 5c.

(iv) **Circular decision boundary -** Data points with $(x^2 + y^2) > 0.5$ are placed in one class all other points were placed in another class. It was a nonlinear decision boundary. The decision boundary is given in Fig. 5d.

In each run, 500 data points were created randomly for training and 5000 data points were created randomly for testing. 10 runs were done for each dataset, each run had different training and testing data points.. For datasets with linear decision boundaries, a linear kernel ($K(x, y) = x^T y+1$)

Table 1 Average classification errors (in % with s.d. in brackets) for different ensemble methods for different decision surfaces. '+/-' shows that the performance of an ensemble method with extended feature space is statistically better/worse than that of that ensemble method with the original feature space

Decision boundary	*Bagging +Kernel features*	Bagging	*AdaBoost.M1 + Kernel features*	AdaBoost.M1	*Random Forests features*	Random Forests
+	0(0.0)	0(0.0)	0(0.0)	0(0.0)	0(0.0)	0(0.0)
y>2*x*	3.1(0.3)	2.9(0.2)	2.7(0.2)	2.7(0.2)	2.6(0.2)	2.5(0.2)
y=x	0.5(0.1)	3.1(0.3)(+)	0.6(0.1)	3.4(0.3)(+)	0.7(0.1)	3.3(0.1)(+)
Circular	0.9(0.2)	2.4(0.4)(+)	1.0(0.2)	2.5(0.3)(+)	0.9(0.3)	2.6(0.3)(+)

was selected, whereas for the dataset with nonlinear decision boundary, Gaussian kernel ($K(x, y) = \exp(-||x - y||^2)$) was used. For the above described datasets, we did not perform the experiments for the RS ensemble method as there were only two attributes in each dataset. The average results of 10 runs for each dataset are presented in Table 1, which suggest that for the dataset with orthogonal decision boundary (+ decision boundary) and the dataset with decision boundary is not very different than orthogonal decision boundary (y>2x), the performance of ensemble methods with extended feature spaces was statistically similar to the performance of original ensemble methods. Whereas for diagonal decision boundary (y=x) and nonlinear decision boundary (circular decision boundary) ensemble methods with extended spaces performed statistically better than original ensemble methods.

These results suggest that ensemble methods with extended feature spaces are useful for linear decision boundaries which are very different from orthogonal decision boundaries, and for nonlinear decision boundaries. Kernel functions are used to represent complex decision boundaries, hence it is expected that kernel features improve the representational power of decision tree ensembles. This is the reason for the success of the decision tree ensembles with extended feature spaces for these decision boundaries.

4.2 Comparative study with benchmark datasets

Experiments were carried out on popular datasets to study the following points;

(i) The performance of decision tree ensembles based on kernel features only.
(ii) The effect of kernel features on existing ensemble methods. In other words, several existing ensemble methods with original features are tested against their versions with extended feature spaces.

Table 2 The information about datasets used in the experiments [26]

Dataset	Training data points	Testing data points	No. of features	No. of sets of training points and testing points
Banana	400	4900	2	100
Breast-Cancer	200	77	9	100
Diabetes	468	300	8	100
Flare-Solar	666	400	9	100
German Credit	700	300	20	100
Heart	170	100	13	100
Image	1300	1010	18	20
Ringnorm	400	7000	20	100
Splice	1000	2175	60	20
Thyroid	140	75	5	100
Titanic	150	2051	3	100
Twonorm	400	7000	20	100
Waveform	400	7000	20	100

Table 3 Average classification errors (in % with s.d.) in brackets for different ensemble methods for different datasets (RS ensembles are not created for the Banana dataset as it has only 2 features). '+/-' shows that the performance of ensemble with kernel features only is statistically better/worse than that algorithm for that dataset. The result in bold shows the winning approach

Dataset	Ensemble with kernel features only	Random subspace (Rs)	Bagging	Random forests	AdaBoost.M1
Banana	11.2(0.5)	–	13.2(.6)(+)	13.4(.7)(+)	13.3(0.7)(+)
Breast-Cancer	29.4(4.8)	25.0(4.3)	28.8(4.5)	26.5(4.3)	30.3(4.9)
Diabetes	25.3(1.8)	25.9(2.2)	24.3(2.2)	24.1(2.0)	25.6(1.8)
Flare-Solar	35.8(1.8)	35.3(2.7)	34.6(2.4)	35.7(2.2)	36.1(1.8)
German Credit	28.3(2.4)	25.1(2.3)	23.9(2.1)	22.2(2.2)	23.6(2.1)
Heart	18.1(3.6)	17.2(3.9)	19.5(3.9)	17.9(3.4)	19.8(3.5)
Image	6.3(1.0)	2.0(0.3)(−)	2.2(0.5)(−)	1.6(0.5)(−)	**1.4(0.5)**(−)
Ringnorm	1.5(0.1)	5.4(0.5)(+)	9.2(0.9)(+)	6.9(0.7)(+)	3.5(0.4)(+)
Splice	22.1(1.1)	**2.8(0.3)**(−)	4.5(0.8)(−)	3.6(0.6)(−)	3.5(0.8)(−)
Thyroid	5.1(2.7)	5.8(2.5)	7.9(2.8)	5.3(2.3)	5.2(2.5)
Titanic	**22.6(1.1)**	25.8(3.4)	22.5(1.3)	22.4(1.5)	**22.6(1.2)**
Twonorm	**2.5(0.2)**	5.1(0.4)(+)	7.2(0.5)(+)	4.8(0.3)(+)	3.6(0.2)(+)
Waveform	**10.0(0.5)**	12.0(0.6)(+)	12.9(0.7)(+)	11.5(0.6)(+)	11.1(0.6)(+)
Win/Draw/Loss		3/7/2	4/7/2	4/7/2	4/7/2

We carried out experiments on benchmark datasets presented in [26]. Information about these datasets is presented in Table 2. Each dataset has many sets of training data points and testing data points (the last column of Table 2). All the sets of training data points and testing data points of the datasets are available online (http://www.raetschlab.org/Members/raetsch/benchmark). For each dataset, we carried out experiments on all the sets of training data points and testing data points. For example, Banana dataset had 100 sets of training data points and testing data points. Hence, for Banana dataset, 100 runs were carried out. The average results of different runs on different datasets are presented. We used the same radial basis function kernel functions as suggested by Ratsch et al. in [26].

Table 3 presents the comparative study of decision tree ensembles created by kernel features only against other ensemble methods. Ensembles created by kernel features performed statistically better than other ensemble methods for four datasets (Banana, Ringnorm, Twonorm and Waveform). Whereas for two datasets (Image and Splice), these ensembles performed statistically worse than other ensemble methods.

There are two important parameters in the proposed method; the kernel function and the dimension of the new kernel features space. If an appropriate kernel is not selected, then the kernel features are very informative. Hence, ensembles with kernel features only are not very accurate. This is true for Splice dataset [26]. For Image dataset, the dimension of the kernel space may be the reason for the poor performance of the ensembles with kernel features only (this point will be discussed later).

The other approach that we proposed, was a combination of kernel features with existing ensemble methods. Table 4 presents the results of *RS+kernel features* and the RS ensemble method. For four datasets (Image, Ringnorm,

Table 4 Average classification errors (in % with s.d.) in brackets for *RS* + *Kernel features* and RS methods for different datasets (RS ensembles are not created for the Banana dataset as it has only 2 features). '+/-' shows that the performance of *RS* + *Kernel features* is statistically better/worse than that RS for that dataset. The result in bold shows the winning approach

Dataset	*RS* + *Kernel features*	RS
Banana	11.2(0.5)	–
Breast-Cancer	27.2(4.2)	**25.0(4.3)**
Diabetes	**24.4(1.8)**	25.9(2.2)
Flare-Solar	35.7(2.4)	**35.3(2.7)**
German Credit	**24.1(2.2)**	25.1(2.3)
Heart	17.6(3.5)	**17.2(3.9)**
Image	**1.6(0.3)**	2.0(0.3)(+)
Ringnorm	**1.5(0.1)**	5.4(0.5)(+)
Splice	3.2(0.5)	**2.8(0.3)**(−)
Thyroid	5.1(2.6)	5.8(2.5)
Titanic	**22.5(1.1)**	25.8(3.4)
Twonorm	**2.5(0.2)**	5.1(0.4)(+)
Waveform	**10.2(0.5)**	12.0(0.6)(+)
Win/Draw/Loss		4/7/1

Table 5 Average classification errors (in % with s.d.) in brackets for *Bagging + Kernel features* and Bagging for different datasets. '+/-' shows that the performance of *Bagging + Kernel features* is statistically better/worse than that Bagging for that dataset. The result in bold shows the winning approach

Dataset	*Bagging + Kernel features*	Bagging
Banana	11.5(0.5)	13.2(.6)(+)
Breast-Cancer	28.1(4.1)	28.8(4.5)
Diabetes	24.2(2.3)	24.3(2.2)
Flare-Solar	35.3(2.2)	**34.6(2.4)**
German Credit	22.9(1.9)	23.9(2.1)
Heart	18.5(3.7)	19.5(3.9)
Image	2.1(0.4)	2.2(0.5)
Ringnorm	1.6(0.1)	9.2(0.9)(+)
Splice	4.7(0.9)	**4.5(0.8)**
Thyroid	6.4(1.8)	7.9(2.8)
Titanic	22.4(1.3)	22.5(1.3)
Twonorm	2.8(0.2)	7.2(0.5)(+)
Waveform	11.1(0.6)	12.9(0.7)(+)
Win/Draw/Loss		4/9/0

Table 6 Average classification errors (in % with s.d.) in brackets for *AdaBoost.M1 + Kernel features* and AdaBoost.M1 for different datasets. '+/-' shows that the performance of *AdaBoost.M1 + Kernel features* is statistically better/worse than that AdaBoost.M1 for that dataset. The result in bold shows the winning approach

Dataset	*AdaBoost.M1 + Kernel features*	AdaBoost.M1
Banana	11.9(0.6)	13.3(0.7)(+)
Breast-Cancer	27.7(4.2)	30.3(4.9)
Diabetes	24.9(1.6)	25.6(1.8)
Flare-Solar	36.0(2.1)	36.1(1.8)
German Credit	22.9(2.0)	23.6(2.1)
Heart	18.6(3.1)	19.8(3.5)
Image	1.3(0.4)	1.4(0.5)
Ringnorm	1.6(0.2)	3.5(0.4)(+)
Splice	3.2(0.7)	3.5(0.8)
Thyroid	4.8(2.1)	5.2(2.5)
Titanic	22.5(1.1)	22.6(1.2)
Twonorm	2.8(0.2)	3.6(0.2)(+)
Waveform	10.5(0.5)	11.1(0.6)(+)
Win/Draw/Loss		4/9/0

Twonorm and Waveform) *RS+kernel features* performed better than the RS method. Interestingly, for Image dataset, *RS + kernel features* performed statistically better than RS. The dimension of kernel space (we selected the dimension as 10) may be less than the required value, however, as they are combined with the original feature, they help in getting better performance. For Splice dataset, *RS + kernel features* performed statistically worse than the RS method, however, its performance (error 3.2%) was better than the ensemble based on kernel features only (error 22.1%).

Table 5 presents the results of *Bagging + Kernel features* and Bagging ensemble methods. For four datasets (Banana, Ringnorm, Twonorm and Waveform) *Bagging + kernel features* performed statistically better than Bagging. For all other datasets, including Image dataset and Splice dataset, *Bagging + Kernel features* performed similar to Bagging. The same behavior is observed for *AdaBoost.M1 + Kernel features* vs AdaBoost.M1 (Table 6) and *Random Forests+ Kernel features* vs Random Forests (Table 7). These results suggest that these ensemble methods are likely to work better than or similar to original ensemble methods (Bagging, Adaboost.M1 and Random Forests). Hence, when kernel features are combined with these ensemble methods, it becomes robust to the choice of the kernel function and the dimension of kernel space. From these results, we cannot conclude the effect of the size of training datasets on the performance of ensembles as many factors work together (e.g., the selected kernel, the dimension of the new feature space). For example, the size of the splice training dataset is the second largest of datasets we studied; however, we observed that generally the performance of ensemble methods degrades slightly with kernel features. It is mainly due to the selection of an inappropriate kernel. This example suggests that it is difficult to find the relationship between the size of training data and the performance of ensembles

Table 7 Average classification errors (in %) with s.d. in brackets of *Random Forests + Kernel features* and Random Forests for different datasets. '+/-' shows that the performance of *Random Forests + Kernel features* is statistically better/worse than that Random Forests for that dataset. The result in bold shows the winning approach

Dataset	*Random Forests + Kernel features*	Random Forests
Banana	12.1(0.6)	13.4(.7)(+)
Breast-Cancer	29.9(4.2)	**26.5(4.3)**
Diabetes	24.7(1.9)	**24.1(2.0)**
Flare-Solar	35.3(2.1)	35.7(2.2)
German Credit	23.1(1.9)	**22.2(2.2)**
Heart	17.8(3.2)	17.9(3.4)
Image	1.5(0.5)	1.6(0.5)
Ringnorm	1.6(0.2)	6.9(0.7)(+)
Splice	3.9(0.5)	**3.6(0.6)**
Thyroid	5.5(2.1)	**5.3(2.3)**
Titanic	22.3(1.4)	22.4(1.5)
Twonorm	2.7(0.2)	4.8(0.3)(+)
Waveform	10.5(0.4)	11.5(0.6)(+)
Win/Draw/Loss		4/9/0

with kernel features. If we consider the datasets in which the results with kernel features significantly improved, except RS, all ensemble methods improved by using kernel features on the same datasets (RS has no result on Banana dataset); Banana, Ringnorm, Twonorm and Waveform. This suggests that the kernel features are useful for decision boundaries which are present in these datasets. We observed that Bagging ensemble method had more improvement on these datasets as compared to other ensemble methods. For example, for Ringnorm dataset, the average classification error for Bagging method with kernel features decreases from 9.2 % to 1.6 %, whereas this improvement in RS was from 5.4 % to 1.5 %, in AdaBoost.M1 from 3.5 % to 1.6 % and for Random Forests from 5.4 % to 1.5 %. There is another observation that for all the datasets, AdaBoost.M1 improved with kernel features, however the improvement was small.

On the basis of our observations, we infer that AdaBoost.M1 is more likely to improve with kernel features, however Bagging is more likely to have larger improvement for datasets, on which all the ensemble methods have significant improvement with kernel features.

4.3 Study of the dimension of the kernel feature space

As discussed earlier that the required dimension of the new kernel feature space is dependent on the margin [5], hence, it is not possible to know in advance the required dimension. If the dimension of the kernel feature space is less than the desired dimension, accurate classification results may not be obtained because all the information of the data is not preserved in the kernel feature space. To study this problem, the similarity of this approach with kernel-PCA method [29] was used. The classification by using a linear classifier with leading kernel PCA features has shown good performance [29]. This suggests that in leading kernel features the classification problem can be assumed to be linearly separable (similarly, in the method proposed by Balcan et al. [5] the problem is linearly separable in the kernel feature space). "What should be the minimum number of the kernel features to recover most of the information required for the classification task" has been an active research area in kernel-PCA field [6]. Braun et al. [6] propose an algorithm for kernel-PCA to estimate the dimension (the number of leading kernel PCA components relevant for accurate classification) and expected error for a learning problem. They calculate the minimum dimensions for the same datasets with which we did our experiments. We present their results in Table 8, which suggest that for Image dataset and Splice dataset, the estimated dimensions are relatively large. As in our experiments, we created only 10 new kernel features, the poor performance of ensembles created by using kernel features only on these datasets may be due to this reason.

In these experiments, we wanted to analyze the effect of the dimension of the kernel feature space on the ensemble accuracies. We varied the dimension of the new feature space, and studied its effect on the performance of ensembles. Four datasets; Image, Splice, Twonorm and Waveform were used in the experiments. Two of these datasets (Image and Splice) have relatively larger estimated dimensions (minimum features to preserve all the information), whereas, the other two datasets (Twonorm and Waveform) have relatively smaller estimated dimensions.

We created ensembles with 10, 50 and 100 (all but one dataset have estimated dimensions less than 100 (Table 8)) kernel features for each decision tree. Decision trees were trained on kernel features only. Experiment settings were similar to the previous experiment. Results suggest (Table 9) that for datasets (Image and Splice) which have larger estimated dimensions, the performance of ensembles improved with the larger number of kernel features for each decision tree. This indicates that for those datasets 10 kernel features were not enough to preserve all the information. Hence, more kernel features were required. Whereas, for datasets with smaller estimated dimensions (Twonorm and Waveform), the performance of ensembles degraded little with decision trees having more than 10 kernel features. We premise that when 50 or 100 kernel features were created for each decision tree, some of these features might be non-informative, that had a negative effect on the performance of the ensembles. Ensembles with decision trees with 100 kernel features showed more classification errors than that of ensembles with decision trees with 50 kernel features. This result validates our premise that if we create kernel features more than the expected dimension of datasets, some

Table 8 The estimated dimensions of different datasets in kernel PCA spaces, taken from [6]

Dataset	Estimated dimensions
Banana	26
Breast-Cancer	2
Diabetes	9
Flare-Solar	10
German Credit	12
Heart	5
Image	368
Ringnorm	37
Splice	89
Thyroid	18
Titanic	6
Twonorm	2
Waveform	23

Table 9 Average classification errors (in % with s.d. in brackets) of ensembles with decision trees trained with 10, 50 and 100 new kernel features. The result in bold shows best performance

Dataset	Ensembles with decision trees trained with 10 kernel features	Ensembles with decision trees trained with 50 kernel features	Ensembles with decision trees trained with 100 kernel features
Image	6.3(1.0)	5.4(0.9)	**5.1(0.8)**
Splice	22.1(1.1)	17.1(1.2)	**16.6(1.1)**
Twonorm	**2.5(0.2)**	2.6(0.1)	2.8(0.2)
Waveform	**10.0(0.5)**	10.1(0.4)	10.4(0.4)

non-informative features are generated and these features have adverse effect on the performance of ensembles.

This experiment verifies that for ensembles with decision trees with kernel features only, the dimension of the kernel feature space is an important parameter and the number of features less than the number of required features may lead to poor results. Results presented in the Table 4 to Table 7 show that one of the proposed ensemble methods, the combination of existing ensemble methods with kernel features, performed well even with default number (10) of kernel features for the datasets (Image and Splice) having large estimated dimensions. This demonstrates that the proposed approach of creating ensembles with extended feature spaces is quite robust to the characteristics of the datasets.

5 Conclusion and future work

Ensemble methods and kernel machines are two important research areas in the machine learning field. Ensemble methods are successful as they use the combined strength of different classifiers. Kernel machines are successful because they use the excellent representational power of kernels. In this paper, we first proposed that kernel features [5] can be used to create decision tree ensembles. We also studied their properties and suggested to combine the existing ensemble methods with kernel features. These kernel features encourage extra diversity in the ensembles. They also help in creating accurate decision trees. Experimental results demonstrate that ensemble methods with extended feature spaces match or outperform ensemble methods with original features. Experiments also suggest that these ensemble methods are more useful for non-orthogonal decision boundaries and nonlinear decision boundaries.

In this paper, we carried out experiments with decision trees. However, the proposed approach is a general framework for creating ensembles and can be used with any classifier. In the future research, we will study other classifiers with low representational power like linear classifiers. Some of the similarity functions cannot be used in SVM as they do not fulfill all the conditions of kernels (continuous, symmetric, positive semi-definitive). The proposed methods can be used with any similarity function as similar mappings have been proposed for similarity functions [4]. We will do the experiments with different similarity functions that are not kernel functions. This will be another area of future research.

References

1. Ahmad A, Brown G. Article in press
2. Amasyali M, Ersoy O. Article in press
3. Arriaga R, Vempala S (2006) An Algorithmic Theory of Learning:Robust Concepts and Random Projection. Mach Learn 63(2):161–182
4. Balcan MF, Blum A (2006) On a Theory of Learning with Similarity Functions. In:Proceedings of the 23rd International Conference on Machine Learning
5. Balcan MF, Blum A, Vempala S (2006) Kernels as Features: On Kernels, Margins, and Low-dimensional Mappings. Mach Learn 65:79–94
6. Braun ML, Buhmann JM, Muller KR (2009) On Relevant Dimensions in Kernel Feature Spaces. J Mach Learn Res 9:1875–1908
7. Breiman L (1996) Bagging Predictors. Mach Learn 24(2):123–140
8. Breiman L (2001) Random Forests. Mach Learn 45(1):5–32
9. Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and Regression Trees. Wadsworth International Group, CA
10. Burges CJC (1998) A Tutorial on Support Vector Machines for Pattern Recognition. Data Min Knowl Disc 2:121–167
11. Caruana R, Niculescu-Mizil A. (2006) An Empirical Comparison of Supervised Learning Algorithms
12. Dietterich TG (2000) Ensemble Methods in Machine Learning
13. Diosan L, Rogozan A, Pecuchet J (2012) Improving Classification Performance of Support Vector Machine by Genetically Optimising Kernel Shape and Hyper-parameters. Appl Intell 36(2):280–294
14. Freund Y, Schapire RE (1996) Experiments with a New Boosting Algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, pages 148–156
15. Gama J, Liu X, Cohen P (1997). In: In Second International Symposium on Advances in Intelligent Data Analysis pages 187–198 (ed) Oblique Linear Tree, Springer-Verlag
16. Geurts P, Ernst D, Wehenkel L (2006) Extremely Randomized Trees. Mach Learn 63(1):3–42
17. Hall M, Frank E, Holmes Geoffrey, Pfahringer B, Reutemann P, Witten IanH (2009) The WEKA Data Mining Software: An Update. SIGKDD Explor 11(1):10–18
18. Hansen LK, Salamon P (1990) Neural Network Ensembles. IEEE Trans on Pattern Anal and Mach Intell 12(10):993–1001

19. Ho TK (1998) The Random Subspace Method for Constructing Decision Forests. IEEE Trans on Pattern Anal and Mach Intell 20(8):832–844
20. Kim H, Pang S, Je H, Kim D, Bang S (2003) Constructing Support Vector Machine Ensemble. Pattern Recog 36(12):2757–2767
21. Kuncheva LI (2004) Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience
22. Mansilla EB, Ho TM (2004) On Classifier Domains of Competence. In Proceedings of the 17th International Conference on Pattern Recognition (ICPR04), pages 136–139
23. Maudes J, Rodrguez JJ, Garca-Osorio C, Pardo C (2011) Random Projections for Linear SVM Ensembles. Appl Intell 34(3):347–359
24. Polikar R (2006) Ensemble Based Systems in Decision Making. IEEE Circuits and Systems Magazine, pages 21–45.Third Quarter
25. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Inc., San Francisco,CA,USA
26. Rätsch G, Onoda T, Müller K-R (2001) Soft margins for AdaBoost. Mach Learn 42(3):287–320
27. Rodriguez JJ, Kuncheva LI, Alonso CJ (2006) Rotation Forest: A New Classifier Ensemble Method. IEEE Trans on Pattern Anal and Mach Intell 28(10):1619–1630
28. Rwebangira MR (2008) Techniques for Exploiting Unlabeled Data. PhD thesis, School of Computer Science. Carnegie Mellon University
29. Scholkopf B, Smola AJ, Muller K (1998) Nonlinear Component Analysis as a kernel Eigenvalue problem. Neural Comput 10:1299—1319
30. Statnikov A, Wang L, Aliferis CF (2008) A Comprehensive Comparison of Random Forests and Support Vector Machines for Microarray-based Cancer Classification. BMC Bioinforma 9(319)
31. Truong Y, Lin X, Beecher C (2004) Learning a Complex Metabolomic Dataset Using Random Forests and Support Vector Machine. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Seattle, Washington, USA, August 22-25, 2004, pages 835–840
32. Tumer K, Ghosh J (1996) Error Correlation and Error Reduction in Ensemble Classifiers. Connect Sci 8(3):385–404
33. Valentini G, Dietterich T (2004) Bias-variance Analysis of Support Vector Machines for the Development of Svm-based Ensemble Methods. J Mach Learn Res:725–775
34. Vapnik V (1998) Statistical Learning Theory. Wiley-Interscience, New York
35. Wang C, You W (2013) Boosting-SVM:Effective Learning With Reduced Data Dimension. Appl Intell 39(3):465–474
36. Wanga S, Mathewb A, Chenc Y, Xia L, Mab L, Lee J (April 2009) Empirical Analysis of Support Vector Machine Ensemble Classifiers. Expert Syst Appl 9(3):6466–6476
37. Webb GI (2000) Multiboosting: A Technique for Combining Boosting and Wagging. Mach Learn 40(2):159–196
38. Zhu M (2008) Kernels and Ensembles: Perspectives on Statistical Learning. The American Stat 62:97–109