[статья]

Abstract

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 fea-ture 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.

Introduction

> 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

> 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 param-eters. Contrary to decision tree ensembles, generally SVM ensembles don't show large improvements over single SVM classifiers [36].

>In this paper, we propose different approaches to create ensembles of decision trees such that each member of an ensemble can use the expres-sive power of the kernel functions. **This approach uses the kernel features created by a mapping proposed by Balcan et al. [5].**

> 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 tree.

> **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.

Ensemble methods with kernel features

>

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]

> Balcan et al. [5] propose that given a pairwise measure of similarity K(xi, xj) between data objects xi and xj , one can construct features in a simple way by **randomly selecting a set x1, x2,...,xd of data points and then using K(x, xi) (similarity with the point xi (i = 1 to d)) as the i-th feature of example x, where d depends on the margin**.

> **The algorithm for ensembles with kernel features only**

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 **x**
for i=1...L **do**
  Convert **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 **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 **x**.

Fig. 1 The algorithm for ensembles with kernel features only

> Decision trees do the feature selection at each node [25]. 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]. 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.

> **Diverse Classifiers - Generation of diverse classifiers is the second condition of effective ensembles.** 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.

> **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 for RS + Kernel features ensembles

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 **x**
for i=1...L **do**
1- Convert **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 **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 **x**.

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

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 **x**

for i=1...$\lfloor L/M \rfloor$ **do**

1- Convert **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 **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 **x**.

Experiments

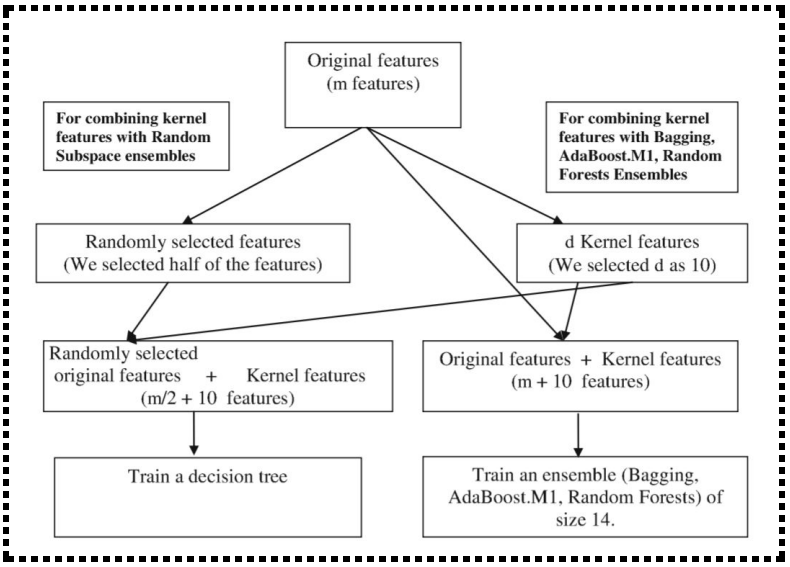
> Experiments were carried out by using **WEKA software [17]**.

>

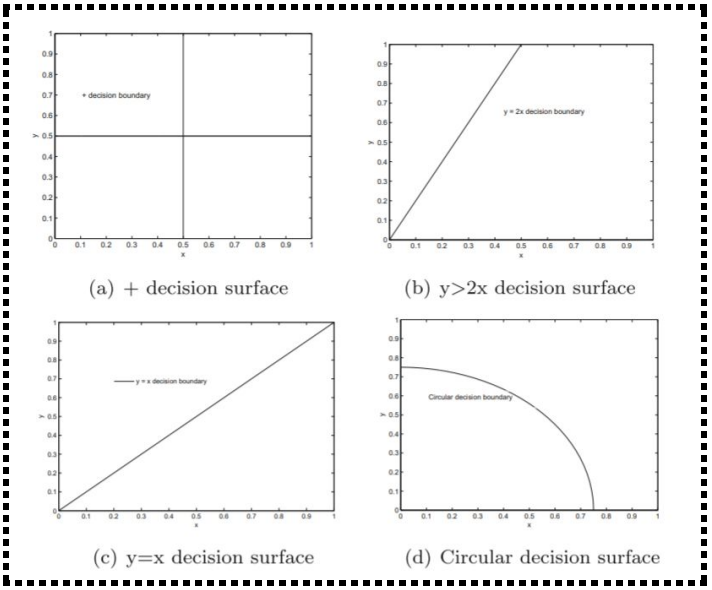
(Bagging, Adaboost.M1, Random Forests) + kernel features ensembles - In these methods, 14 ($200^{0.5} \approx 14$) runs were used. In each run, 14 ($200^{0.5} \approx 14$) trees were trained. In other words, in each ensemble 196 trees were created. The size parameters were selected such that almost 200 trees were created in each ensemble.

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

> **The framework** for training a classifier for the proposed ensemble methods in the experiments.



> Эксперименты на синтезированных данных



> Результаты реальных датасетов

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

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

> 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).

> From these results, we cannot conclude the effect of the size of training datasets on the performance of ensembles as many factors work together

> All ensemble methods improved by using kernel features on the same datasets

> 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.**

> 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.

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

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.

>

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)

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. 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.** This result validates our premise that if we create kernel features more than the expected dimension of datasets, some non-informative features are generated and these features have adverse effect on the performance of ensembles.

Их идеи

1) In the future research, we will study other classifiers with low representational power like linear classifiers.
2) 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.