Randomized Support Vector Forest

Xutao Lv[1]
xutao.lv@sri.com
Tony X. Han[2]
hantx@missouri.edu
Zicheng Liu[3]
zliu@microsoft.com
Zhihai He[2]
hezhi@missouri.edu

[1] SRI International
Princeton, NJ, USA
[2] University of Missouri
Columbia, MO, USA
[3] Microsoft Research
Seatle, WA, USA

Abstract

Based on the structural risk minimization principle, the linear SVM aiming at finding the linear decision plane with the maximal margin in the input space has gained increasing popularity due to its generalizability, efficiency and acceptable performance. However, rarely training data are evenly distributed in the input space [10], which leads to a high global VC confidence [9], downgrading the performance of the linear SVM classifier. Partitioning the input space in tandem with local learning may alleviate the unevenly data distribution problem. However, the extra model complexity introduced by partitioning frequently leads to overfitting. To solve this problem, we proposed a new supervised learning algorithm, Randomized Support Vector Forest (RSVF): Many partitions of the input space are constructed with partitioning regions amenable to the corresponding linear SVMs. The randomness of the partitions is injected through random feature selection and bagging. This partition randomness prevents the overfitting introduced by the over-complicated partitioning. We extensively evaluate the performance of the RSVF on several benchmark datasets, originated from various vision applications, including the four UCI datasets, the letter dataset, the KTH and the UCF sports dataset, and the Scene-15 dataset. The proposed RSVF outperforms linear SVM , kernel SVM, Random Forests (RF), and a local learning algorithm, SVM-KNN, on all of the evaluated datasets. The classification speed of the RSVF is comparable to linear SVM.

1 Introduction

Successful supervised learning algorithms must have reached an operating point balancing the model complexity and learning capacity [3, 5, 6, 9, 11, 13, 14, 19, 21, 26, 31]. The linear SVM aiming at finding the linear decision plane with the maximal margin in the input space has gained increasing popularity due to its generalizability and efficiency [17, 26]. But the performance of a linear SVM depends on the linear separability of the data in the input space. That is, linear SVM may perform poorly on data with a distribution not linearly separable, as illustrated in Figure 1. Kernel-SVM may be a remedy but the selection of kernels is task-dependent and tricky: the capacity control parameters, which are implicitly represented as kernel type and support vector numbers, are globally determined by the data distribution in the input space. Furthermore, the slow speed of kernel SVM both in training and classification prevents its application in large scale problems, which constitute a significant portion of current computer vision research.

© 2014. The copyright of this document resides with its authors.

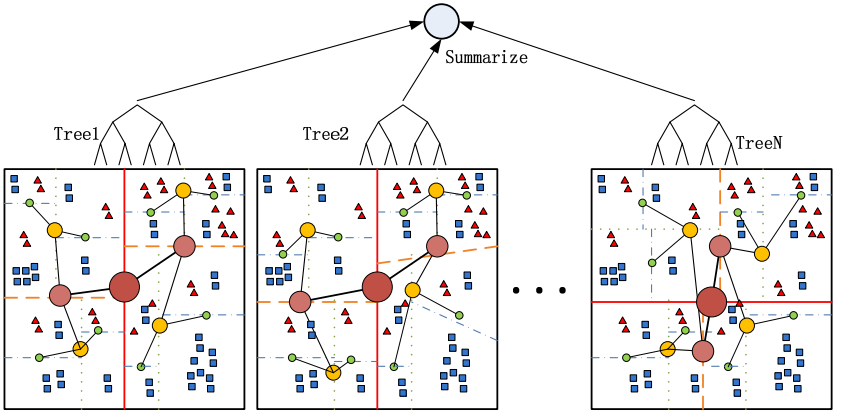


Figure 2: The structure of RSVF. This figure shows a RSVF with N trees. Each tree, with depth 5, is demonstrated in the last row of the figure. The small green dots are the Linear SVM classifiers; the other dots are the binary classifiers. Note, the binary classifiers mentioned in this paper represent decision nodes, which use a threshold to split the data into two child nodes.

Partitioning the input space in tandem with local learning is another way of tuning learning capacity according to data distribution. Bottou and Vapnik [1] proposed a general local learning framework with an implementation which uses the K Nearest Neighbor (KNN) as a data partitioner, and the neural network as a local classifier. Zhang *et al*. [31] proposed an SVM-KNN approach, in which K nearest neighbors in the training set for each testing sample are selected and a local SVM is learned from the K neighbors to classify the testing sample. However, both of the algorithms have the same discrepancy between the data partitioner and the local classifier: KNN as a data partitioner aims at picking up data samples close to each other without considering the local pattern separability in determining the performance of the local classifier; closeness in a user-defined metric does not necessarily lead to the desired pattern separability. For the large scale learning problems, the low efficiency of KNN limits the applicability of the local learning algorithms discussed above. There are other more efficient data partition method such as clustering, hashing, and hierarchical trees. However, the extra model complexity introduced by partitioning frequently leads to overfitting, the main reason that we need to prune a decision tree if it is too large [18].

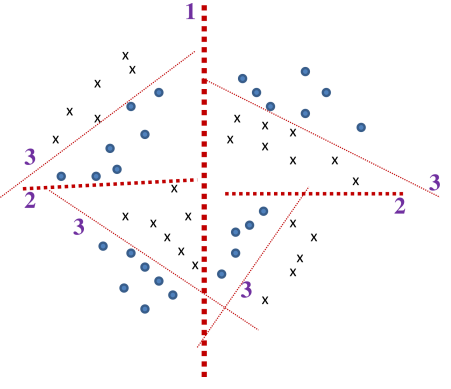


Figure 1: Demonstration of data distribution of the toy example. The cross marks represent the positive samples and the blue circle marks stand for the negative samples.

To solve this problem, we proposed a new supervised learning algorithm, Randomized Support Vector Forest (RSVF): Many partitions of the input space are constructed with par-

It may be distributed unchanged freely in print or electronic forms.

titioning regions amenable to the corresponding linear SVMs. The randomness of the partitions is injected through random feature selection and bagging. This partition randomness prevents the overfitting introduced by the over-complicated partitioning.

As shown in Figure 2, the basic idea is to split the dataset, with certain randomness, into many small linearly separable partitions and train a linear SVM for each partition. The training data are organized into different hierarchical partitions with randomized partition criteria. Among randomly selected partition planes at each node, the one which leads to the maximal separating margin for the child partitions is selected. Unlike previous local learning algorithms that use the nearest neighbor for partitioning, we use randomized forest with minimum error rate criterion to partition the input space, reconciling the discrepancy between the data partitioner and local classifier. The partitioned input space under such rule will be shown to lead to better classification performance than linear SVM, kernel SVM, Random Forests (RF) [3], and SVM-KNN, in Section 3.

As shown in Figure 2, a tree in the proposed RSVF may bear some similarities with a decision tree. But there are two major differences. First, decision tree uses the information gain or Gini importance to guide the node splitting while the RSVF uses the linear separability (i.e., the margin between samples) of each child node, in order to reduce the learning capacity of the local classifier. Second, the decision tree estimates the class label through a majority voting at the nodes, whereas in the proposed RSVF, much more powerful linear SVM is applied.

There have been several previous works based on pooling of customized trees, such as the Discriminant Random Forest (DRF) [15], the Binary SVM Decision Trees (BSDT) [16] and, the decision trees in [29] (RF3). The node in DRF is split using Linear Discriminant Analysis (LDA). For BSDT, the internal nodes of SVM-BDT are trained by first partitioning the data through clustering, labeling two partitions of data with label '+1' and '-1' and the SVM classifiers are trained on these two partitions of data. The RF3 are designed specifically for image classification with very large number of categories. At each node of RF3, binary labels are randomly assigned to each category and the corresponding linear SVM is trained. For the previous works discussed above, the nodes are split through information gain or LDA without considering the linear separability of the leaf nodes.

Our main contribution lies in two folds. First, we reconcile the discrepancy between the data partitioner and the local classifier: for the data partitioner, we use the estimated linear separability of the resulting linear SVM as the data partition criterion while previous local learning algorithms used distances defined in the input space. Second, by injecting the randomness to data partition through random feature selection and bagging, we avoid the overfitting caused by over-complicated data partitioners. The advantages of the proposed RSVF are: (1) Very good performance compared favorably with popular supervised learning algorithms including linear SVM, kernel SVM, RF, and SVM-KNN; (2) Efficiency thanks to its shallow depth and the linear operation at each node; (3) Parallelizability.

2 Randomized Support Vector Forest (RSVF)

2.1 Training RSVF

As illustrated in Figure 2, the RSVF consists of many Support Vector Trees (SVT). Each SVT represents a scheme of data partition and the corresponding local classifier. The final classification result of RSVF is a pooling from all the SVTs. After comparing various pooling methods including the majority voting, and max voting, i.e., taking the prediction from the SVT with the maximal confidence, we use majority voting from all of the trees in

the forest for its simplicity and efficacy. We grow the RSVF through a procedure similar to growing the Classification And Regression Trees (CART) in random forest [23].

The majority voting from all SVTs in the resulted RSVF is used to achieve the final classification.

2.2 Training Support Vector Trees (SVTs)

Our algorithm consists of two major steps. First, the input space or the partite cells are partitioned according to the linear separability of the child nodes. Second, within each partite cell, an linear SVM is learned to classify the feature vectors. As shown in Figure 2, each SVT in the proposed RSVF has two types of nodes, inner nodes and leaf nodes. The inner nodes are partition nodes, serving as data partitioners; and the leaf nodes are linear SVM classifiers, serving as local classifiers. As a data partitioner, an inner node of an SVT should facilitate the task of the corresponding local classifier, i.e., the corresponding linear SVM, by enlarging the linear separability of the local data inside the cell. A straightforward estimate of the linear separability is the classification margin of the local linear SVM. However, the classification margin should be normalized according to the partition cell size and number of local data; this normalization is very sensitive and leads to fragile estimate.

To make the estimate of the linear separability insensitive to the cell size and local data amount, we use the error rates of the learned linear SVM on the Out-Of-Bag (OOB) data as the estimate, which has been shown to be very stable and accurate [2, 3]. By training the inner nodes with linear separability estimated as the error rate on OOB data, the input space is partitioned to favor linear separable cells, leading to well performed local linear SVMs, as illustrated in Figure 1.

Inner Node Split Rule for RSVF Each node corresponds to a partition cell, i.e., a polyhedron, in the input space. The cell associated with each inner node of a SVT is split into two sub-cells, corresponding to the two child nodes . The selection of the optimal split is based on the OOB error rates of the linear SVMs learned for the two sub-cells. The split corresponding to the lowest OOB error rates is used to split the inner node. The OOB error rate of a local linear SVM is computed by counting the number of classification errors of the OOB training data falling into the corresponding cell.

Stop Criterion Each inner node is recursively split until the total OOB error rate of the child nodes (computed using Equation 1), is equal to or larger than that of their parent node, which indicates further splitting will not improve the performance of the SVT.

Training SVT A SVT $\mathcal{T}$ is trained using the bootstrap dataset $\mathcal{X}^*$ and the OOB data $\mathcal{X}\backslash\mathcal{X}^*$. It is constructed by recursively splitting a set of samples $\mathcal{S} \subseteq \mathcal{X}^*$ into two distinct subsets $\mathcal{S}_l$ and $\mathcal{S}_r$ and splitting a set of OOB samples $\mathcal{S}_O \subseteq \mathcal{X}\backslash\mathcal{X}^*$ into two distinct subsets $\mathcal{S}_{Ol}$ and $\mathcal{S}_{Or}$ as well: one parent node is recursively split into two child nodes, until the stop condition is met.

The split quality is measured as the drop of the error rate on the OOB data falling into the split sub-cells:

$$\delta e = e - (p_l e_l + p_r e_r), \tag{1}$$

where the proportions of OOB samples in left and right nodes are represented as p_l and p_r respectively. The error rates of the parent node, left and right child node are denoted as e, e_l and e_r. They are determined by the SVM classifiers trained and tested on those pairs of data $(\mathcal{S}, \mathcal{S}_O)$, $(\mathcal{S}_l, \mathcal{S}_{Ol})$ and $(\mathcal{S}_r, \mathcal{S}_{Or})$ respectively.

Each SVT is trained with the following steps:

Step 1 Randomly propose a set of splitting candidates $\Phi = \{\phi | \phi = (\theta, \omega, \tau)\}$, where θ is the feature dimension index. The dimension of θ is a predefined constant, which means

the number of variables that are involved in random projection. ω is the random projection weight vector and each value of ω is randomly generated in $[0,1]$ and the dimension of ω is the same as θ. τ is the threshold and it is randomly selected from the values computed on $\mathcal{S}$ by $\omega^T \cdot \mathbf{f}_\theta$, where $\mathbf{f}_\theta$ is constructed by taking values from the feature vector at the dimensions determined by θ.

Step 2 Partition the set of samples $\mathcal{S} = \{\mathbf{x}\}$ into left subset and right subset with each ϕ:

$$\mathcal{S}_l(\phi) = \{\mathbf{x} | \omega^T \cdot \mathbf{f}_\theta(\mathbf{x}) < \tau + \delta\tau, \mathbf{x} \in \mathcal{S}\} \tag{2}$$

$$\mathcal{S}_r(\phi) = \{\mathbf{x} | \omega^T \cdot \mathbf{f}_\theta(\mathbf{x}) > \tau - \delta\tau, \mathbf{x} \in \mathcal{S}\} \tag{3}$$

$$\mathcal{S}_{Ol}(\phi) = \{\mathbf{x} | \omega^T \cdot \mathbf{f}_\theta(\mathbf{x}) < \tau + \delta\tau, \mathbf{x} \in \mathcal{S}_O\} \tag{4}$$

$$\mathcal{S}_{Or}(\phi) = \{\mathbf{x} | \omega^T \cdot \mathbf{f}_\theta(\mathbf{x}) > \tau - \delta\tau, \mathbf{x} \in \mathcal{S}_O\}, \tag{5}$$

where $\delta\tau$ is the boundary relaxation factor. Given a node, let τ denote its partition threshold. Those samples fall between $[\tau - \delta\tau, \tau + \delta\tau]$ will be included into both child partitions to avoid the split boundary perturbation. The boundary relaxation factor $\delta\tau$ is determined through cross validation.

Train two linear SVM classifiers on $\mathcal{S}_l$ and $\mathcal{S}_r$, and evaluate them on $\mathcal{S}_{Ol}$ and $\mathcal{S}_{Or}$, accordingly. The error rate $e(\phi)$ is computed using Equ. 1.

Step 3 Compute the optimal ϕ^* producing the largest drop in error rate as the split candidate by Equ. 6:

$$\phi^* = \arg\max_{\phi} \delta e(\phi) \tag{6}$$

Step 4 If $\delta e(\phi^*) > 0$, then recursively split the left and right subset pairs $(\mathcal{S}_l(\phi^*), \mathcal{S}_{Ol}(\phi^*))$ and $(\mathcal{S}_r(\phi^*), \mathcal{S}_{Or}(\phi^*))$; if $\delta e(\phi^*) \leq 0$, then stop the split. The subset $\mathcal{S}$ is used to train the leaf node SVM classifiers.

Each resulting leaf node in the RSVF represents a local input space with high linear separability, which boosted the performance of the corresponding local linear SVM.

2.3 Performance and generalization discussion

The generalization of the RSVF benefits from the randomness injected through random feature selection and bagging, which is also essential to the generalization of random forests [3].

With the injected randomness, the generalization error of RSVF can be proved to converge almost surely using the Law of Large Numbers when the number of SVTs increases. The detailed proof is given in the Appendix A in the supplementary materials. As the number of trees in RSVF increases, for almost surely all Θ, the generalization error e_g of RSVF converges to,

$$P_{\mathbf{X},Y}(P_\Theta(\mathcal{T}(\mathbf{X},\Theta) = Y) - \max_{j \neq Y} P_\Theta(\mathcal{T}(\mathbf{X},\Theta) = j < 0) \tag{7}$$

where $\mathcal{T}$ is an SVT; $\mathbf{X}$ is feature matrix; Y is the label of $\mathbf{X}$; and Θ is a set of parameters ϕ^* associated with the SVT $\mathcal{T}$.

Theoretical analysis on RF has shown that the error rate of RF depends on two factors [3]. First, the correlation between any two trees in the forest: increasing the correlation increases the forest's error rate. Second, the strength of each individual tree in the forest: increasing the strength of the individual trees decreases the forest's error rate. Compared to RF, the proposed RSVF keeps the same correlation among trees but increases the strength of each tree (verified in our experiments), leading to a better performance than the RF. The experiments

Section 3.2 show that trees in RSVF are stronger than their peers in RF. The difference between RSVF and RF lies in the following three aspects: 1) The trees in RSVF are used as the data partitioner and have very small depth while the trees in RF are used as the classifier and have large depth, as shown in Section 3; 2) RSVF is inherently a local learning algorithm while RF is an ensemble of tree classifiers; 3) An inner node of each CART in RF is split using the Gini importance while an inner node of each SVT in RSVF is split according to the linear separability estimated using the OOB error rate of the resulting children node.

3 Experiments

Several benchmark datasets from various applications are used to evaluate the proposed RSVF. Action recognition datasets including, KTH [22], and UCF sports dataset [20] are used. Four UCI datasets [8] widely accepted as the benchmark in machine learning community are included as well. Additionally, we perform image classification task on fifteen scene categories dataset [25].

3.1 The evaluation philosophy

We propose RSVF to provide a generic supervised local learning tool. Therefore, we compared the proposed RSVF with the state-of-the-art supervised learning algorithms including linear SVM, kernel SVM, random forest, and a local learning algorithm SVM-KNN [31]. For the fair comparison, we use the same feature for all compared algorithms. And for each algorithm compared, we search the tuning parameters of the compared algorithm and the best performance is used to compare. We aim at comparing the performances of RSVF with the popular supervised learning algorithms using standard feature; it is not our intention to achieve the best performance on the benchmark datasets. Therefore, for vision applications, we try to avoid using domain specific feature/model such as deformable model, spatial pyramid, and dense tracking feature and only use the standard bag-of-visual-words feature, which frequently serves as the baseline for feature comparison.

In our experimental setup, the only tuning parameter is the number of the support vector trees. For each hypothesis evaluation, we apply the linear SVM from the liblinear package with all default settings (The fixed default value 1.0 for the regularization parameter is used) based on an experiment on the effect of regularization parameter on KTH dataset. The result shows that the RSVF is quite insensitive to the regularization parameter. We vary the regularization parameter from 10^{-8} to 10, resulting a performance perturbation less than 0.5%; the RSVF reaches its best performance when the regularization parameter is larger than 10^{-6}.

3.2 The letter recognition experiments on UCI dataset

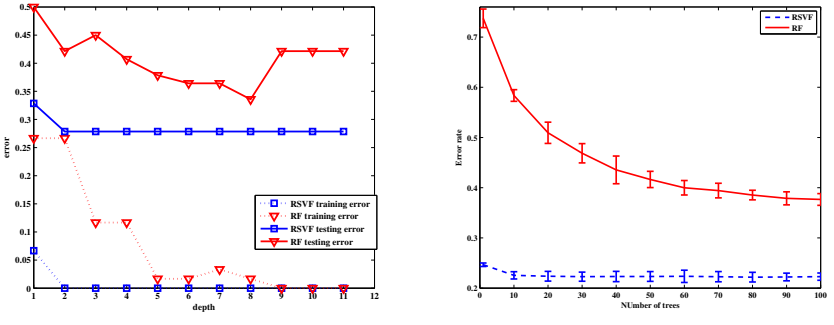
The letter recognition dataset [8] contains 20000 samples in 26 categories. We follow the recommended setting in UCI machine learning repository [8], where we have 16000 samples for training and 4000 samples for testing. Two experiments are carried out on this dataset. The first experiment is one-vs-all binary classification where we compare RSVF with the two baselines, Linear SVM and RF. The second experiment is a multiclass classification task where each leaf node in the RSVF is a multi-class SVM classifier (one-vs-all). The results in Table 1 show how differently the RF and RSVF respond to different tree depth as we have discussed in Section 2.3.

k	t	LSVM	RF	RF3	RSVF	K	SVM-KNN
		0.3243				2	0.0568
2	1		0.9363		0.2825	4	0.0715
3	1		0.9280		0.244	8	0.1125
5	1		0.7483		0.1438	16	0.1785
7	1		0.5770		0.12	32	0.2690
22	1		0.1858				
5	100		0.525		0.0703		
7	100		0.3633		**0.0338**		
22	100		0.0393	0.041			

Table 1: Performance comparison with Random Forests (RF) and Linear SVM (LSVM) on letter dataset. The column k is the maximum depth of each tree; the column t is the number of trees in the forest. Note, for RF, depth larger than 22 will not improve the performance any more; RF3 is proposed in [4].

Evaluation with multiclass classifier Table 1 shows the error rates of various classifiers on the letter dataset. From the experiments, the SVM-KNN seems to be parameter sensitive with the number of nearest neighbors selected. Linear SVM performs worst, because the feature for this dataset is not linearly separable. Comparing the best results achieved by RSVF and RF, in order for RF to achieve a performance close to RSVF, the depth of trees has to increase drastically. Since the generalization capability goes down as the tree depth goes up, the obtained RSVF has better generalization capability than RF as shown in the experiment below.

Overfitting We design an experiment to show the overfitting property of both RSVF and RF. We randomly leave 70% of 160000 examples (from training data) for test and the rest 30% for training. Both RSVF and RF have 10 trees and the error rate for depth from 1 to 10 is reported in Fig. 3(a). When the depth is larger than 8, the RF training error decreases while its testing error goes up, which shows that RF is overfitting.



(a) Demonstration of overfitting

(b) Error rate VS number of trees

Figure 3: Fig. 3(a) shows the overfitting of RF. On training dataset of letter dataset, with 70% left for testing and 30% for training. They both have 10 trees. In Fig. 3(b), it shows the relationship between the error rate and the number of trees in the forest. It is evaluated on a random split of dataset Scene-15. x-axis represents the number of trees in the RSVF and RF; y-axis represents the error rate. Note, for visualization purpose, the length of each error bar is two times of the actual standard deviation.

3.3 The action recognition experiments

The KTH dataset: The KTH dataset [23] has been used as a benchmark dataset to evaluate action recognition algorithms. It contains six actions (boxing, hand clapping, hand waving, jogging, running and walking) performed by 25 subjects in 4 scenes. We follow the original

Method	LSVM	RF	RSVF	SVM-KNN	χ^2-KSVM	RBF-KSVM
KTH*	92.59%	91.67%	**93.98%**	87.04%	92.59%	92.13%
UCF	65.7 ± 5.8%	61.5 ± 7.3%	**72.2 ± 5.4%**	48.4 ± 5.6%	66.3 ± 6.6%	62.3 ± 6.7%
Scene15	75.1 ± 0.3%	63.3 ± 0.9%	**78.3** ± 0.4%	59.9 ± 0.9%	76.9 ± 0.4%	75.7 ± 0.6%

Table 2: Recognition accuracy on KTH, Scene-15 and UCF sports datasets. *Note: since the training and the testing sets are fixed in the KTH dataset, we just follow the standard setup so that our result can be compared with [7, 10, 12, 27].

experiment setup in [22], which are adopted by most of action recognition works including [7, 10, 12, 27]. Actions of 16 subjects are used for training and actions of 9 subjects are used for testing. There are 599 samples in this KTH dataset, where 383 samples are assigned for training and 216 for testing. STIPs [27] are used as the basic descriptor, and a bag-of-visual-words model is used to generate a histogram as the feature vector for each video. The codebook is generated using a K-means algorithm with $K = 2000$, resulting in a 2000 dimensional feature vector. The classification result by RSVF is compared favorably with linear SVM, random forests, kernel SVMs (using χ^2 kernel and RBF kernel), and SVM-KNN as shown in Table 2. In this experiment, we use RSVF with 100 trees. During the training process, we randomly check 44 θ vectors, 2 ω vectors and 40 τ values, which results in a set of $44 \times 2 \times 40$ ϕ candidates. For the other experiments (other than KTH), we only use one dimensional ω to split the nodes. For KTH dataset, the feature vector generated is quite sparse. There are only 10-30 non-zero values at each dimension. It is hard to get a balanced split with only one dimension for the training samples. Therefore, using the combination of two random variables to split the nodes is necessary.

The UCF Sports dataset: The UCF Sports dataset contains 10 actions: swinging, diving, kicking, lifting, horse-riding, running, skateboarding, swing (high bar), swing-golf, walking. There are in total 150 videos with large variance in each action category. To increase the number of training samples, we follow the same scheme in [27] that adds flipped version to the training data. Instead of using the leave-one-out experiment setup [27, 28], we make the experiment more challenging by randomly selecting 40% of 150 videos for testing and the rest for training. The whole learning-testing process is repeated 10 times to show the statistical significance. The average and standard deviation of each compared supervised learning algorithm is reported in Table 2.

On both of the KTH and UCF dataset, the RSVF outperforms other methods significantly. The feature used here is simple STIP descriptor [27] with bag-of-visual-words model. With the recently proposed features such as dense trajectories for action recognition [28], the accuracy will likely be further improved.

3.4 The Image classification experiments on the Scene-15 dataset

The Scene-15 dataset consists of 15 scene categories. Each category has 200 to 400 images with average size 300×250 pixels. Following the original experiment setup on this dataset [25], we leave 100 images per category for training and the rest of the images for testing. The whole learning-testing process is repeated for 10 times; the average and standard deviation of each compared supervised learning algorithm is reported in Table 2.

To form the feature vector, we extract SIFT descriptors from every image densely. The grid spacing is 8×8, and patch size is 16×16. Bag of words model is used to generate histogram feature for each image. The codebook is trained by K-means algorithm, which is fixed to 4096. The performance of our baseline Linear SVM is comparable to that of [25] which uses a similar feature . With the feature generating method from [30] and RSVF, the performance can be further improved.

Type	Best in [24]	Linear SVM	RBF-SVM	RSVF	RF
dna	0.059 ± 0.005	0.088 ± 0.017	0.054 ± 0.010	**0.052 ± 0.008**	0.056 ± 0.011
wine	0.030 ± 0.029	0.023 ± 0.024	0.016 ± 0.022	**0.002 ± 0.007**	0.014 ± 0.016
iris	0.057 ±0.022	0.038 ± 0.026	0.032 ± 0.025	**0.029 ±0.048**	0.041 ±0.029
glass	0.232 ±0.047	0.408 ± 0.091	0.300 ± 0.059	**0.223 ± 0.068**	0.234 ± 0.055

Table 3: Performance comparison on UCI datasets. The results in the first column is obtained from [24].

On the Scene-15 dataset, we also investigate the relationship between the error rate and the number of trees in RSVF. As illustrated in Figure 3(b), more SVTs in the RSVF will lead to better performance. Additionally, because the SVT in RSVF is a stronger classifier compared to decision trees in RF, RSVF usually requires much less trees than RF.

3.5 Experiments on more datasets from UCI repository

We evaluate the RSVF on 4 more datasets from UCI repository. The experiment setup follows [24]; 25% data are randomly selected as testing dataset, and the rest 75% are considered as training dataset. Each algorithm runs 10 times on the dataset. The average error rate and the standard deviation are reported in Table 3. On the UCI datasets, the RSVF outperforms the state-of-the-art results reported in [24], and it also performs better than linear SVM, non-linear SVM with RBF kernel, and RF.

3.6 Computational complexity

The complexity of Linear SVM is $O(n)$ for classification, where n is the dimension of the feature vectors. The complexity of RF for classification is $O(tk)$, where t is the number of trees in the forest and k is the depth of the trees. The SVTs in RSVF has lower depth, no more than 5 in all of our experiments except the letter recognition where the depth is 7, which are negligible compared with feature dimension in most of the vision applications. Note the depth of SVTs are determined automatically. Therefore, the complexity of RSVF is $O(tn)$, where t is the number of trees in the forest and n is the dimension size of feature vectors. Comparing RSVF with Linear SVM, the number of trees is usually not very big, which makes its complexity comparable to Linear SVM. Our extensive evaluations show that RSVF outperforms RF and linear/non-linear SVM significantly. Therefore, considering both the computational complexity and the performance, RSVF is a very promising alternative to SVM and RF. The RSVF is easily parallelized due to the independence among SVTs. For our experiments, We set up a Hadoop cluster with capacity of 126 tasks to train each tree of the RSVF concurrently.

4 Conclusion

We proposed the Randomized Support Vector Forest (RSVF) for classification. Compared to liner SVM, RSFV does not assume the linear separability of the data. Compared to RF, RSFV has a much smaller depth resulting in better generalization capability and is less likely to overfit. Compared to the local learning algorithms, the RSVF avoid the extra model complexity introduced by the data cautioner. The performance of RSFV was evaluated extensively on several standard datasets from various vision applications. The proposed RSVF outperforms linear SVM, Random Forests (RF), Kernel SVM and SVM-KNN on all the datasets.

Appendix A: Proof of the Convergence of the RSVF

Suppose the training data set, $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$, is obtained by draw randomly from the distribution of the random vector $(\mathbf{X}, Y)$, where $\mathbf{X}$ is the feature and Y

is the class label. Given K SVTs $\mathcal{T}_1(\mathbf{x}), \mathcal{T}_2(\mathbf{x}), \ldots, \mathcal{T}_K(\mathbf{x})$, we define the RSVF's margin function for $(\mathbf{X}, Y)$ as,

$$f(\mathbf{X}, Y) = \frac{1}{K}\sum_k I(\mathcal{T}_k(\mathbf{X}) = Y) - \max_{j \neq Y} \frac{1}{K}\sum_k I(\mathcal{T}_k(\mathbf{X}) = j)$$

where $I(.)$ is the indicator function. The margin function $f(\mathbf{X}, Y)$ measures the extent to which the average vote at $\mathbf{X}, Y$ for the right class exceeds the majority vote for any other class. The generalization error is then given by

$$\mathcal{E}_g = P_{\mathbf{X},Y}(f(\mathbf{X}, Y) < 0),$$

where $P_{\mathbf{X},Y}$ is the probability over $\mathbf{X}$ and Y.

Recall that in STEP 1 of the training an SVT (at page 6 of the paper), the kth SVT is constructed from a random vector ϕ_k, the splitting candidate, generated independently from the same distribution Φ. We represent the kth SVT as: $\mathcal{T}_k(\mathbf{X}) = \mathcal{T}(\mathbf{X}, \phi_k)$.

As mentioned in Section 2.3, the RSVF does not overfit while the number of trees increases. More rigorously, we are going to show the following theorem.

Theorem.
While the number SVTs increases, for almost surely all sequences $\phi_1, \ldots, \phi_k, \ldots$, the generalization error $\mathcal{E}_g$ converges to

$$P_{\mathbf{X},Y}(P_\Phi(\mathcal{T}(\mathbf{X}, \Phi) = Y) - \max_{j \neq Y} P_\Phi(\mathcal{T}(\mathbf{X}, \Phi) = j < 0).$$

Proof. It is sufficient to show that there is a set of probability zero $\mathcal{C}$ on the sequence space $\phi_1, \phi_2, \ldots$ such that outside of $\mathcal{C}$, for all $\mathbf{x}$,

$$\frac{1}{N}\sum_{n=1}^{N} I(\mathcal{T}(\phi_n, \mathbf{x}) = j) \to P_\Phi(\mathcal{T}(\Phi, \mathbf{x}) = j).$$

For a fixed training set and a fixed ϕ_k, the set of all $\mathbf{x}$ such that $\mathcal{T}(\phi_k, \mathbf{x}) = j$ is a union of polyhedrons in the feature space. If Φ is a discrete random vector, which is true for SVT, there is only a finite number of such unions of polyhedrons in the feature space. Denote this finite number as F, and these unions as $U_1, \ldots, U_F$. Define $\gamma(\Phi) = k$ if $\{\mathbf{x} : \mathcal{T}(\Phi, \mathbf{x}) = j\} = U_k$. Let N_k be the number of times that $\gamma(\phi_n) = k$ in the first N trials. Then

$$\frac{1}{N}\sum_{n=1}^{N} I(\mathcal{T}(\phi_n, \mathbf{x}) = j) = \frac{1}{N}\sum_k N_k I(\mathbf{x} \in U_k).$$

By the Law of Large Numbers, $N_k = \frac{1}{N}\sum_{n=1}^{N} I(\gamma(\phi_n) = k)$ converges a.s. to $P_\Phi(\gamma(\Phi) = k)$. Taking unions of all the sets on which convergence does not occur for some value of k gives a set $\mathcal{C}$ of zero probability such that outside of $\mathcal{C}$,

$$\frac{1}{N}\sum_{n=1}^{N} I(\mathcal{T}(\Phi, \mathbf{x}) = j) \to \sum_k P_\Phi(\gamma(\Phi) = k) I(\mathbf{x} \in U_k),$$

where

$$\sum_k P_\Phi(\gamma(\Phi) = k) I(\mathbf{x} \in U_k) = P_\Phi(\mathcal{T}(\Phi, \mathbf{x}) = j)$$

□

References

[1] Leon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 4: 888–900, 1992.

[2] Leo Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[3] Leo Breiman and E. Schapire. Random forests. In *Machine Learning*, volume 45, pages 5–32, 2001.

[4] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[5] O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, 10(5):1055–1064, Sep 1999. ISSN 1045-9227. doi: 10.1109/72.788646.

[6] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 – 27, jan 1967. ISSN 0018-9448. doi: 10.1109/TIT.1967.1053964.

[7] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72, 2005.

[8] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 2000.

[10] Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using mined hierarchical compound features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5): 883–897, 2011.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. The Elements of Statistical Learning. Springer, 2001.

[12] Nazli Ikizler-Cinbis and Stan Sclaroff. Object, scene and actions: combining multiple features for human action recognition. In *Proceedings of the 11th European conference on Computer vision: Part I*, ECCV'10, pages 494–507, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15548-0, 978-3-642-15548-2.

[13] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[15] T. D. Lemmond, A. O. Hatch, B. Y. Chen, D. A. Knapp, L. J. Hiller, M. J. Mugge, and W. G. Hanley. Discriminant random forests. In *2008 International Conference on Data Mining, DMIN 2008*, pages 55–61, 2008.

[16] Gjorgji Madjarov, Dejan Gjorgjevikj, and Tomche Delev. Ensembles of Binary SVM Decision Trees. In *ICT Innovations*, pages 181–188, 2010.

[17] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[18] Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 195–201. Morgan Kaufmann, 1997.

[19] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

[20] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action MACH a spatio-temporal Maximum Average Correlation Height filter for action recognition. In *CVPR*, 2008.

[21] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[22] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, volume 3, pages 32–36 Vol.3, 2004.

[23] Daniel F. Schwarz, Inke R. König, and Andreas Ziegler. On safari to random jungle: a fast implementation of random forests for high-dimensional data. *Bioinformatics*, 27 (3):439, 2011.

[24] Chunhua Shen and Zhihui Hao. A direct formulation for totally-corrective multi-class boosting. In *CVPR*, pages 2585–2592, 2011.

[25] Cordelia Schmid Svetlana Lazebnik and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *In CVPR*, pages 2169–2178, 2006.

[26] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998. ISBN 978-0-471-03003-4.

[27] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[28] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011.

[29] Bangpeng Yao, Aditya Khosla, and Fei-Fei Li. Combining randomization and discrimination for fine-grained image categorization. In *CVPR*, pages 1577–1584, 2011.

[30] Junsong Yuan, Ming Yang, and Ying Wu. Mining discriminative co-occurrence patterns for visual recognition. In *CVPR*, pages 2777–2784, 2011.

[31] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *In CVPR*, pages 2126–2136, 2006.