Journal of Electronic Imaging

JElectronicImaging.org

Joint maximum purity forest with application to image super-resolution

Hailiang Li
Kin-Man Lam
Dong Li

Hailiang Li, Kin-Man Lam, Dong Li, "Joint maximum purity forest with application to image super-resolution," *J. Electron. Imaging* **27**(4), 043005 (2018), doi: 10.1117/1.JEI.27.4.043005.

Joint maximum purity forest with application to image super-resolution

Hailiang Li,[a,*] Kin-Man Lam,[a] and Dong Li[b]
[a]Hong Kong Polytechnic University, Department of Electronic and Information Engineering, Hong Kong, China
[b]Guangdong University of Technology, School of Automation, Guangzhou, China

Abstract. We propose a random-forest scheme, namely joint maximum purity forest (JMPF), for classification, clustering, and regression tasks. In the JMPF scheme, the original feature space is transformed into a compactly preclustered feature space, via a trained rotation matrix. The rotation matrix is obtained through an iterative quantization process, where the input data, inclined to different classes, is clustered to the respective vertices of the feature space with maximum purity. In the feature space, orthogonal hyperplanes, which are employed at the split nodes of the decision trees in a random forest, can effectively tackle the clustering problems. We evaluated our proposed method on public benchmark datasets for regression and classification tasks, and experiments showed that JMPF remarkably outperforms other state-of-the-art random-forest-based approaches. Furthermore, we applied JMPF to image super-resolution (SR) specifically because the transformed, compact features are more discriminative to the clustering-regression scheme. Experimental results on several public datasets also showed that the JMPF-based image SR scheme is consistently superior to recent state-of-the-art image SR algorithms. *© 2018 SPIE and IS&T* [DOI: 10.1117/1.JEI.27.4.043005]

Keywords: random forest; regression and classification; image super-resolution; ridge regression.
Paper 171014 received Nov. 22, 2017; accepted for publication Jun. 15, 2018; published online Jul. 9, 2018.

1 Introduction

Recently, random forest[1,2] has been employed as an efficient classification or regression tool on a large variety of computer-vision applications, such as object classification,[3] recognition,[4] face alignment,[5,6,7] data clustering,[8] and image super-resolution (SR).[9,10] This method is attractive for computer-vision problems, not only for its simple implementation, but also for a number of its merits: (1) it can work efficiently on both training and inference stages, (2) it is feasible for it to be sped up with parallel processing technology, (3) it has an inherent property to handle high-dimensional input features, and (4) it works as an ensemble of tree classifiers, and achieves robust and accurate performance on classification and regression tasks.

Random forest is a machine-learning method using an ensemble of randomized decision trees, and each tree consists of split nodes and leaf nodes that are trained recursively. During the training process, at each split node in a decision tree, a hyperplane is learned to separate data into two groups. Although each decision tree attempts to achieve maximum purity during training, i.e., maximizing the interclass variance and minimizing the intraclass variance, for the clustered data in the two child nodes of each split node, there is no guarantee that the original feature space can meet the expectation of global maximum purity for all the clustered groups. The hyperplanes in a random forest have the orthogonal constraint as shown in Fig. 1, which hinders obtaining the optimal hyperplanes as a support vector machine (SVM)[11,12,13] does in the original feature spaces. In this paper, we aim to break this orthogonal-constraint limitation. With the fixed orthogonal hyperplanes, we propose to rotate the feature space; this is equivalent to rotating the hyperplanes in such a way that global maximum purity on the clustered data can be achieved, as shown in Fig. 2. This strategy can achieve a joint maximum purity for all the split nodes when training a random forest. In summary, we propose a feature preprocessing step, where the original features are transformed into a feature space via a constructed rotation matrix. The performance of the constructed random forest can be improved in the feature space. Similar ideas can be found in the rotation forest proposed in Ref. 3, but the rotation matrices are constructed in a different way. Moreover, the rotation forest[3] essentially is restricted to the classification task only.

Image SR can be performed based on clustering/classification, according to the recent emerging clustering-regression stream,[14,15,9] and the JMPF scheme can achieve remarkable performance on both the classification and regression tasks. Therefore, JMPF is applied to single-image SR in this paper so as to demonstrate its superior performances. In our algorithm, principal components analysis (PCA) is applied to the features for dimensionality reduction. The projected feature space is then rotated to a compact, preclustered feature space via a learned rotation matrix. Finally, for all the split nodes trained for a random forest, their thresholds are directly set to the inherent zero-center orthogonal hyperplanes in the rotated feature space to meet the maximum-purity criterion. Experimental results show that JMPF can achieve more accurate clustering performance, and applying JMPF to image SR can achieve superior quality, compared with state-of-the-art methods.

*Address all correspondence to: Hailiang Li, E-mail: harley.li@connect.polyu.hk

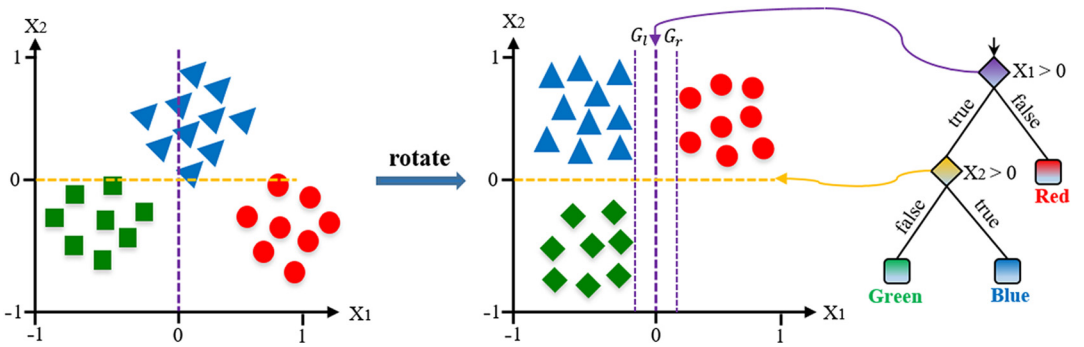

Fig. 1 (a) Three classes of samples in a feature space, which are hard to be clustered with orthogonal hyperplanes; and (b) the samples are rotated, and a decision tree of a random forest is used to cluster the data in the new, rotated feature space.

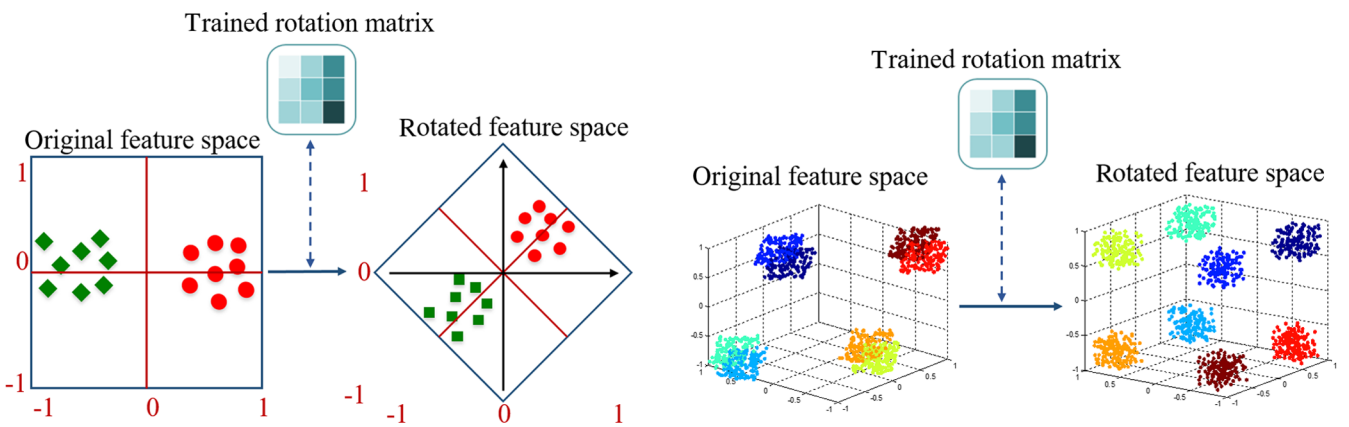

Fig. 2 Two toy examples of rotating a feature space into a more compact clustered feature space: (a) two-dimensional features and (b) three-dimensional features. The feature data are clustered into the vertices of a feature space, by jointly maximizing the purity of all the clustered data.

Having introduced the main idea of our proposed algorithm, the rest of this paper is organized as follows: in Sec. 2, we will describe our proposed JMPF scheme, and present in detail how the rotation matrix is obtained via clustering data into the feature-space vertices. Section 3 will evaluate our proposed method and compare its performance with recent state-of-the-art random-forest-based approaches on regression and classification tasks. In Sec. 4, we will validate the performance of JMPF scheme on single-image SR. Conclusions are given in Sec. 5.

2 Joint Maximum Purity Forest Scheme

2.1 *Random Forest and Our Insights*

In mathematical equation, a random forest is an ensemble of T binary decision trees $\mathcal{T}^t(x): X \to \mathbb{R}^d$, where $t(=1,2,\ldots,T)$ is the index of the trees, $X \in \mathbb{R}^m$ is the data in the m-dimensional feature space, and $\mathbb{R}^d = [0,1]^d$ represents the space of class probability distributions over the label space $Y = \{1,\ldots,d\}$. The training samples are assumed to have a zero mean. Figure 1(a) shows that it is, in general, difficult to find axis-aligned hyperplanes to separate the samples. In Fig. 1(b), the samples are rotated appropriately, so that samples from different classes are clustered into different vertices in a rotated space. Then, the vertical dotted line, $X_1 = 0$, forms the optimal hyperplane for the first split node, whereas the horizontal dotted line, $X_2 = 0$, is the optimal hyperplane for the second split node to cluster all the feature data assigned to this node. This results in separating the three sets of samples (red, green, and blue) into three separate leaf nodes.

It can be seen from Fig. 1(b) that, for each split node, the optimal hyperplane with more generalization capability is the one that can achieve maximum purity in clustering samples into two groups. For example, the vertical dotted line is the first optimal hyperplane because it clusters all the red training samples into the right child node, whereas all the blue and green samples are clustered into the left child node, where the left margin G_l and the right margin G_r are equal.

The training of a whole random forest is to train all of its decision trees, by choosing the candidate features and thresholds for each of the split nodes, where the feature candidates and thresholds are determined using a random bagging strategy. In the prediction stage, each decision tree returns a class probability $p_t(y|x)$ for a given query sample $x \in \mathbb{R}^m$, and the final class label y is then obtained via averaging over all the decision trees, as follows:

$$y^* = \arg\max_y \frac{1}{T}\sum_{t=1}^{T} p_t(y|x). \tag{1}$$

The splitting function for a split node is denoted as $s(v;\Theta)$, where v is a sample and Θ is typically parameterized by two values: (i) a feature candidate $\Theta^i\{1,\ldots,m\}$, and

(ii) a threshold $\Theta^t \in \mathbb{R}$. The splitting function is defined as follows:

$$s(v;\Theta) = \begin{cases} 0, & \text{if } v(\Theta^i) < \Theta^t, \\ 1, & \text{otherwise,} \end{cases} \tag{2}$$

where the outcome defines to which child node the sample v is routed, and 0 and 1 are the two labels for the left and right child nodes, respectively. Each node chooses the best splitting function Θ^* out of a randomly sampled set $\{\Theta^t\}$ by minimizing the following equation:

$$I = \frac{|L|}{|L|+|R|}H(L) + \frac{|R|}{|L|+|R|}H(R), \tag{3}$$

where L and R are the sets of samples that are routed to the left and the right child nodes, respectively, and $|S|$ represents the number of samples in the set S. The most important part in Eq. (3) is $H(S)$, which is a criterion to describe the data information in the sample set S. Mathematically, $H(S)$ is the local score for a set of samples (S is either L or R), which is normally calculated using entropy as in Eq. (4), but it can be replaced by variance[9,6,7] or the Gini index[2]

$$H(S) = -\sum_{k=1}^{K}\{p(k|S)\log[p(k|S)]\}, \tag{4}$$

where K is the number of classes in S, and $p(k|S)$ is the probability for class k, given the set S. For the regression problem, the differential entropy

$$H(q) = \int_y q(y|x)\log[q(y|x)]d_y, \tag{5}$$

over continuous outputs can be employed, where $q(y|x)$ denotes the conditional probability of a target variable given the input sample. Assuming $q(.,.)$ to be a Gaussian distribution and having only a finite set S of samples, the differential entropy can be written in closed form as follows:

$$H_{\text{Gauss}}(S) = \frac{K}{2}[1-\log(2\pi)] + \frac{1}{2}\log[\det(\Sigma_S)], \tag{6}$$

where $\det(\Sigma_S)$ is the determinant of the estimated covariance matrix of the target variables in S. For training each decision tree in a random forest, the goal on each split node is to maximize the information gain (IG) by reducing the entropy after splitting. IG is defined as follows:

$$\text{IG} = \text{entropy(parent)} - [\text{average entropy(children)}]. \tag{7}$$

As each decision tree is a binary tree and each step is to split a current node (a parent set S) into two child nodes (the L and R sets), IG can be described as follows:

$$\arg\max_{\mathcal{H}} \text{IG} = \arg\max_{L,R} H(\text{S}) - \frac{|L|}{|L|+|R|}H(L) - \frac{|R|}{|L|+|R|}H(R), \tag{8}$$

where $\mathcal{H}$ is the optimal hyperplane of the split node, and Eq. (8) is the criterion function of each split node when training a decision tree. As we can see from Fig. 1(b), all the optimal hyperplanes for the split nodes are achieved independently and locally.

As each optimal hyperplane is obtained from a subset of feature candidates with the randomly bagging strategy, there is no guarantee of obtaining a global optimum with respect to all the hyperplanes in all the split nodes. An intuitive thinking, which was inspired by the data distribution in Fig. 1(b), is to achieve a global optimum by jointly considering all the hyperplanes of all the split nodes, as described by the following equation, which can be solved through a greedy approach

$$\max_{\widehat{\mathcal{H}}_k} \text{IG}_{\text{global}} = \arg\max_{\widehat{\mathcal{H}}_k} \prod_{k=1}^{\mathcal{K}} \text{IG}_k, \tag{9}$$

where $\mathcal{K}$ is the total number of split nodes that a training sample has routed through a decision tree. As there is no mathematical solution to the problem described in Eq. (9), an alternative way (i.e., an approximate method) to numerically solving [Eq. (9)] is to jointly maximize the purity of the clustered data groups at each of the split nodes. This also means that all the data are clustered into the vertices of the feature space, as shown in Fig. 2.

2.2 *Joint Maximum Purity Forest Scheme*

By studying the mechanism of a random forest, we can see that the random-forest approach has some critical properties as do other powerful classifiers, such as SVM[11,13] and AdaBoost (short for "Adaptive Boosting").[16] Both SVM and AdaBoost work as to approximate the Bayes decision rule—known to be the optimal classifiers—by minimizing a margin-based global loss function. Each threshold in a decision tree of a random-forest works as a hyperplane, and each single decision tree, similar to AdaBoost, acts as a weak classifier and attempts to minimize its global loss greedily and recursively,[16,17] working through from the root node down to leaf nodes in a binary tree.

As shown in Fig. 2, a number of split nodes, which have their hyperplanes orthogonal to each other, are required to separate the samples into different nodes. However, if we can transform the samples, whose mean is zero, to the respective corners of the feature space, i.e., $\{-1,1\}^m$ for m-dimensional features, the feature data can be easily and accurately separated by the orthogonal (either vertical or horizontal) hyperplanes, which contain the space center $\{0\}^m$, as shown in Fig. 1(b). The insight behind this is that the data are clustered into the feature-space vertices (the corners in a 2-D feature space means that the data points belong to $\{-1,1\}^2$ as the data have been normalized in the range of $[-1,1]$).

To tackle the original feature data X, which is not ideally clustered in the vertices of the feature space or close to them, as shown in Fig. 1(a), an intuitive idea is to rotate the feature space (this is equivalent to rotating the hyperplanes). This transformation clusters the feature data compactly into the feature-space vertices $\{-1,1\}^m$, with a total of 2^m vertices. Therefore, a possible solution to the problem described in Eq. (10) is to rotate the data features by a rotation matrix $\mathcal{R}^{m\times m}$, as shown in Fig. 2, through which the original feature space X is transformed into a more compact clustered feature space, where all the feature data are clustered close to their

inclined feature-space vertex B. This solution can be mathematically defined as follows:

$$\min \|B - X\mathcal{R}\|_F^2 \quad \text{s.t. } B \in \{-1,1\}^{n\times m}, \quad \mathcal{R}^T\mathcal{R} = I, \quad (10)$$

where $\|\cdot\|_F$ is the Frobenius norm, $X \in \mathbb{R}^{n\times m}$ contains n samples, each of which is a m-dimensional feature vector arranged in a row, and is zero-centered, i.e., all the feature vectors are demeaned by subtracting the mean vector from each feature vector.

This idea of clustering data into the feature-space vertices can also be found in locality-sensitive hashing (LSH)[18] and the image representation in Ref. 19. In Ref. 18, a simple and efficient alternating minimization scheme was proposed to find a rotation matrix for zero-centered feature data, which minimizes the quantization errors by mapping the feature data to the vertices of a zero-centered binary hypercube. The method is termed as iterative quantization (ITQ), which can work on multiclass spectral clustering and the orthogonal Procrustes problem. Yu et al.[20] proposed using a circulant matrix to speed up the computation because the circulant structure enables the use of fast Fourier transformation (FFT). As the computation of the rotation matrix in the training and testing stages is ignorable, we choose a similar scheme to ITQ[18] to determine the rotation matrix $\mathcal{R}$, (we throw away the final quantization matrix B described in Eq. (10), which is used for hashing in Ref. 18.) through which the original feature space X can be transformed into a compact clustered feature space: $\tilde{X} = X\mathcal{R}$, where the data are clustered into the respective vertices in the feature space. After this transformation, a random forest with globally joint maximum purity, with the clustered data, can be trained through all the hyperplanes in the split nodes of each decision tree. Based on this idea, our proposed scheme is called joint maximum purity forest (JMPF). This idea is similar to the "kernel trick" in SVM.[11,13] When data are not linearly separable, the "kernel trick" can help it to be linearly separable in the new, higher dimensional feature space.

The rotation forest[3] aims at building accurate and diverse classifiers, for tackling the famous accuracy–diversity dilemma. The rotation forest[3] and the JMPF scheme share a similar idea, such that the original features are transformed into a feature space via a learned "rotation" matrix, and the performance of random forest can be improved in the feature space. However, this random-forest scheme is different from our proposed one because the method to construct the respective "rotation" matrices is different, as well as where they are applied. The rotation forest[3] works with feature extraction as a preprocessing step, which consists of splitting the feature set into subsets, applying PCA separately to each subset, and finally reassembling an extracted feature set while keeping all the components via a constructed "rotation" matrix. In summary, the main idea of the rotation forest algorithm is to strengthen the individual accuracy and diversity of every weak classifier simultaneously in the ensemble of trees.

2.3 *Learning the Rotation Matrix via Clustering Data into Feature-Space Vertices*

Assuming that $x \in \mathbb{R}^m$ is one point in the m-dimensional feature space X (zero-centered data), the respective vertices in the zero-centered binary hypercube space can be denoted as $\text{sgn}(x) \in \{-1,1\}^m$, and there is a total of 2^m vertices in the m-dimensional feature space. It is easy to see from Fig. 2 that $\text{sgn}(x)$ is a vertex in the feature space, such that it is the closest to x in terms of Euclidean distance. We denote a binary code matrix $B \in \{-1,1\}^{n\times m}$, whose rows $b = \text{sgn}(x) \in B$. For a matrix or a vector, $\text{sgn}(.)$ applies the sign operation to it elementwise.

Our objective is to minimize the error between the feature X and the feature-space vertices B, i.e., $\min \|B - X\|^2$. As we can see in Fig. 2, when the feature space is rotated, the feature points will be more concentrated around their nearest vertices, which means that the quantization error will become smaller. Therefore, the minimization problem of $\min \|B - X\|^2$ is equivalent to minimizing the error of the zero-centered data with respect to the Frobenius norm, as in the following equation:

$$Q(B,\mathcal{R}) = \|B - X\mathcal{R}\|_F^2, \quad \text{s.t. } B \in \{-1,1\}^{n\times m}, \quad \mathcal{R}^T\mathcal{R} = I. \quad (11)$$

Therefore, the task of this minimization problem is to determine an optimal rotation matrix $\mathcal{R}$ to satisfy Eq. (11). As there are two variables in Eq. (11), the expectation–maximization (E–M) algorithm is applied to cluster data into the feature-space vertices, such that a local minimum of the binary code matrix B and the rotation matrix $\mathcal{R}$ are computed simultaneously.

The idea of rotating feature data to minimize the error between the transformed data and the feature-space vertex B can also be found in Ref. 19, which showed that the rotation matrix $\mathcal{R}$ can be initialized randomly, and then iterated to converge to the required rotation matrix. Two iteration steps will be performed. In every iteration, each feature vector in the feature space is first quantized to the nearest vertex of the binary hypercube, i.e., to a vertex in B, and then the rotation matrix $\mathcal{R}$ is updated to minimize the quantization error with B fixed. These two alternating steps are described in detail below

1. Fix $\mathcal{R}$ and update B

$$\begin{aligned} Q(B,\mathcal{R}) &= \|B - X\mathcal{R}\|_F^2 \\ &= \|B\|_F^2 + \|X\|_F^2 - 2tr(B\mathcal{R}^T X^T) \\ &= n \times m + \|X\|_F^2 - 2tr(B\mathcal{R}^T X^T). \end{aligned} \quad (12)$$

Because the zero-centered data matrix X is fixed, minimizing Eq. (12) is equivalent to maximizing the following equation:

$$r(B\mathcal{R}^T X^T) = \sum_{i=1}^{n}\sum_{j=1}^{m} B_{ij}\tilde{X}_{ij}, \quad (13)$$

where $\tilde{X}_{ij}$ is an element of $\tilde{X} = X\mathcal{R}$. To maximize Eq. (13) with respect to B, $B_{ij} = 1$ whenever $\tilde{X}_{ij} \geq 0$ and $B_{ij} = -1$ otherwise, i.e., $B = \text{sgn}(X\mathcal{R}) \in \{-1,1\}^m$.

2. Fix B and update $\mathcal{R}$

The problem of fixing B to obtain a rotation matrix based on the objective function Eq. (11) is related to the classic

orthogonal Procrustes problem,[21,22,23] in which a rotation matrix is determined to align one point set with another.

In our algorithm, these two point sets are the zero-centered dataset X and the quantized matrix B, respectively Therefore, a closed-form solution for $\mathcal{R}$ is available, by applying SVD on the $m \times m$ matrix XB^T to obtain $U\Omega V^T$ (Ω is a diagonal matrix), then set $\mathcal{R} = UV^T$ to update $\mathcal{R}$.

2.4 *Proof of the Orthogonal Procrustes Problem*

For completeness, we prove the orthogonal Procrustes problem, whose solution can be found in Refs. 21, 22, and 23. The orthogonal Procrustes problem is a matrix approximation problem. In its classic form, given two matrices B and X, a rotation matrix $\mathcal{R}$, subject to $\mathcal{R}^T\mathcal{R} = I$ and the mapping $X\mathcal{R}$ being closest to B, can be determined as follows:

Problem definition:

$$\min_{\mathcal{R}} \|B - X\mathcal{R}\|_F^2 \quad \text{s.t. } \mathcal{R}^T\mathcal{R} = I. \tag{14}$$

Proof:

$$\begin{aligned}\|B - X\mathcal{R}\|_F^2 &= tr(B - X\mathcal{R})(B^T - \mathcal{R}^T X^T) \\ &= tr(B - X\mathcal{R})(B^T - \mathcal{R}^T X^T) \\ &= tr(BB^T) - 2tr(BX^T\mathcal{R}^T) + tr(\mathcal{R}XX^T\mathcal{R}^T),\end{aligned} \tag{15}$$

Thus, $\min_{\mathcal{R}}\|B - \mathcal{R}X\|_F^2$ is equivalent to maximizing

$$\begin{aligned}tr(BX^T\mathcal{R}^T) &= tr(U\Omega V^T\mathcal{R}^T) \quad (\text{SVD on } BX^T = U\Omega V^T) \\ &= tr(\Omega V^T\mathcal{R}^T U) \quad (\text{denote: } Z = V^T\mathcal{R}^T U) \\ &= tr(\Omega Z) \\ &= tr\sum_i Z_{i,i}\Omega_{i,i} \\ &\le \sum_i \Omega_{i,i},\end{aligned} \tag{16}$$

The last inequality holds because Z is also an orthonormal matrix, and $\sum_j Z_{i,j}^2 = 1$, $Z_{i,i} \le 1$. The objective function can be maximized if $Z = I$, i.e.,

$$\mathcal{R} = UV^T$$

3 Joint Maximum Purity Forest for Regression and Classification

The proposed JMPF can be applied to classification and regression tasks. In this section, we will first present the overall JMPF algorithm, and then compare it with the original random forest, as well as some state-of-the-art random-forest variants.

3.1 *Workflow of Joint Maximum Purity Forest*

Most random-forest-based models[9,6,24,25] share a similar workflow, as shown in Fig. 3, in which the main task on training a decision tree is to decide the thresholds for the split nodes and learn the regressors or classifiers in the leaf nodes. Rigid regression or linear regression is often employed in the leaf nodes for the prediction task because rigid regression has a closed-form solution, whereas linear regression is an efficient optimization tool, and the LibLinear package[26] can be used to fine-tune its configurations.

Compared with conventional random forests, our JMPF scheme has one additional step, as shown in the left side of Fig. 3, the rotation matrix. The JMPF scheme transforms the original feature space by rotating it into a more compact, preclustered feature space, using a rotation matrix learned through clustering the feature vectors iteratively into the vertices of a feature space. The whole workflow of our proposed algorithm, the JMPF scheme, is shown in Fig. 3.

3.2 *Inherent Zero-Center Hyperplanes as Thresholds for Split Nodes*

In training a random forest, the two main operations for training each split node are to choose the feature candidate(s), and to determine the threshold, in which a random bagging strategy is employed to make sure each of the trees is different from each other so as to avoid overfitting. In the rotated compact preclustered feature space, the inherent zero-center hyperplanes are inherently the optimal thresholds (to meet the max-purity criterion on two clustered data groups). Therefore, these inherent zero-center hyperplanes can directly be set as the thresholds to achieve optimal classification performance on training a random forest. Compared with conventional random forests, our proposed JMPF only needs to choose the feature candidate(s) to split data at split nodes. These inherent zero-center hyperplanes can speed up the training process for a random forest.

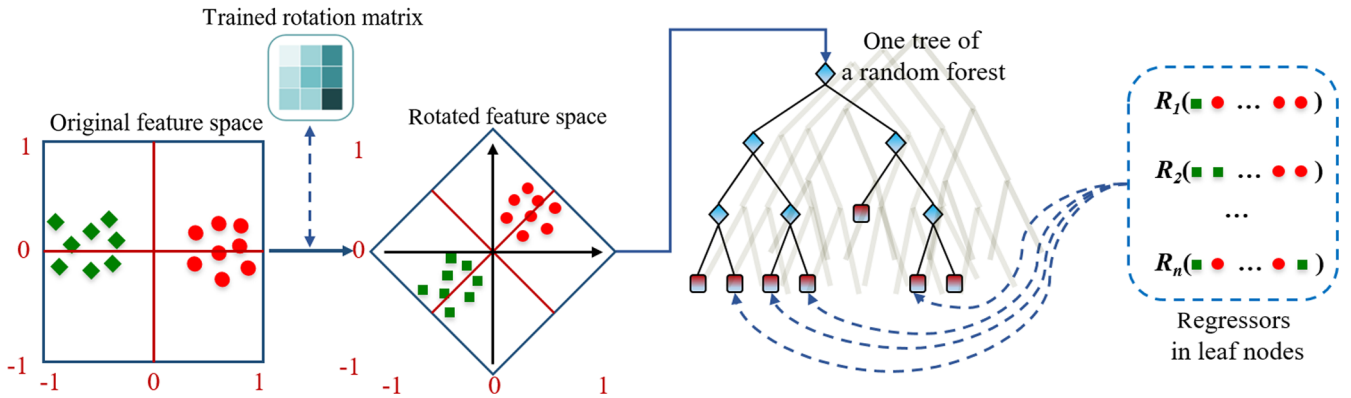

Fig. 3 An overview of the workflow of the JMPF-based random forest.

3.3 *Experimental Results on JMPF Regression and Classification*

To evaluate the performances of the proposed JMPF, we test it with 15 standard machine-learning tasks, 7 for classification, and 8 for regression, respectively. The datasets used in the experiments are summarized in Table 1. We use standard performance evaluation metrics: error rate for classification and root mean squared error (RMSE) for regression, unless otherwise specified.

We first evaluate the proposed approach on two real applications, one for classification (Table 2) and another one for regression (Table 3). Our proposed JMPF is compared with the original random forest (denoted as RF), and the alternating decision forest (ADF)[24] and the rotation forest proposed in Ref. 3 for the classification task, and the alternating regression forest (ARF)[25] for the regression task. The rotation forest in Ref. 3 is restricted to classification because the data are required to be labeled for its training. Furthermore, we compare JMPF with JMPF + ADF/ARF to demonstrate that our algorithm can be integrated with other methods. We follow the experimental settings in Refs. 24 and 25. We set the maximum tree depth at 15, and the minimum number of samples in a split node is set at 5. The experiments were repeated five times, and the average error and standard deviation were both measured. The results are shown in Tables 2 and 3, for the classification and regression tasks, respectively. In terms of accuracy, our proposed JMPF significantly outperforms the standard random forest on all the classification and regression tasks. Compared with RF, JMPF achieves an average improvement of 23.57% on the classification tasks, and an average improvement of 23.13% on the regression tasks. Compared with the rotation forest in Ref. 3, our algorithm can achieve relatively better performance. For the seven given cases, JMPF performs better in three cases, and only two cases for the rotation forest in Ref. 3.

Table 1 The properties of the standard machine-learning datasets used for classification and regression. The top 7 are used for classification (c) and the bottom 8 for regression (r). (3/4 means 75% training and 25% testing).

Dataset	#Train	#Test	#Feature	#Classes or TargetDim
(c)char74k	66,707	7400	64	62
(c)gas sensor	11,128	2782	128	6
(c)isolet	6238	1558	617	26
(c)letterorig	16,000	4000	16	26
(c)pendigits	7494	3498	16	10
(c)sensorless	46,800	11700	48	11
(c)usps	7291	2007	256	10
(r)delta ailerons	7129*3/4	7129/4	5	1
(r)delta elevators	5720	3807	6	1
(r)elevators	8752	7847	18	1
(r)kin8nm	8192*3/4	8192/4	8	1
(r)price	159*3/4	159/4	15	1
(r)pyrim	74*3/4	74/4	27	1
(r)stock	950*3/4	950/4	10	1
(r)Wiscoin BreastCancer	194*3/4	194/4	32	1

Our method also consistently outperforms the state-of-the-art variants: ADF/ARF. Moreover, the performance of our JMPF algorithm can be further improved by integrating with ADF and ARF, denoted as JMPF + ADF/ARF. As shown in Tables 2 and 3, compared with RF, JMPF + ADF achieves an average 27.86% improvement on the classification tasks, whereas JMPF + ARF achieves an average 26.88% improvement on the regression tasks compared to the standard random forest. These results on diverse tasks clearly demonstrate the effectiveness of our proposed approach.

3.4 *Discussions on Experimental Results*

The computational complexity of JMPF is similar to that of the standard random forest. As shown in the workflow of JMPF in Fig. 3, only one additional step, which rotates the feature space, is required when compared with the standard random forest. For a small dataset (e.g., feature dimension <500 and data size $<10,000$), the computation required to compute the rotation matrix for clustering data into the feature-space vertices is acceptable in the training stage (around 10 s for MATLAB), and negligible in the testing stage. When the dimension becomes larger, PCA can be employed for dimensionality reduction. If the size of the dataset increases, such that using PCA still involves heavy computation, bagging can be used to achieve comparable accuracy and the overall extra computation will be insignificant.

To study the stability of JMPF, we choose the letterorig dataset for classification and the kin8nm dataset for regression, and the respective results are shown in Figs. 4(a) and 4(b), respectively. In the experiments, the number of trees, i.e., the number of weak classifiers in the random forest, varies from 10 to 200, and we have three observations. First, as shown in Fig. 4, when the number of trees increases, the performance of all the algorithms improves. For classification, as shown in Fig. 4(a), when the number of trees is >100, the errors are converged to become steady. On the contrary, for the regression task as shown in Fig. 4(b), the errors are almost stable when the number of trees is >20 to 30. Second, the results show that JMPF consistently outperforms ADF and RF, irrespective of the number of trees used. Finally, Fig. 4 clearly shows that JMPF can integrate with ADF or ARF to further improve its performance.

4 Image Super-Resolution Based on Joint Maximum Purity Forest

4.1 *Overview of Image Super-Resolution and Related Works*

Image SR, which recovers a high-resolution (HR) image from one single image or a few low-resolution (LR) images,

Table 2 Comparison of classification performances on seven datasets, which can be found at UCI machine-learning repository.[27] RF, standard random forest; ADF, alternating decision forest;[24] JMPF: proposed algorithm, JMPF + ADF: our proposed algorithm embedded into ADF, and RotationF: the rotation forest algorithm.[3] $\#\mathcal{H}$ is the number of randomly chosen hyperplane(s) on training a split node in random forest. λ is the error scale. The percentages in brackets for JMPF and JMPF + ADF are the reduction rates in RMSE compared with the RF algorithm.

Dataset	$\#\mathcal{H}$	RF	ADF	JMPF	JMPF + ADF	RotationF	λ
char74k	1	2.26 ± 0.02	2.17 ± 0.01	2.15 ± 0.02 (05%)	2.11 ± 0.02 (07%)	1.77 ± 0.03	10^{-1}
	3	2.45 ± 0.03	2.24 ± 0.02	2.21 ± 0.03 (10%)	2.14 ± 0.02 (12%)		
	5	2.45 ± 0.02	2.23 ± 0.02	2.21 ± 0.02 (10%)	2.14 ± 0.02 (13%)		
gas sensor	1	5.66 ± 0.53	5.24 ± 0.54	4.21 ± 0.25 (26%)	3.96 ± 0.51 (30%)	5.46 ± 0.80	10^{-3}
	3	6.26 ± 0.04	5.95 ± 0.32	4.62 ± 0.30 (26%)	4.42 ± 0.37 (30%)		
	5	6.47 ± 0.33	5.75 ± 0.79	4.78 ± 0.46 (26%)	4.16 ± 0.32 (36%)		
isolet	1	6.93 ± 0.28	6.21 ± 0.34	6.15 ± 0.38 (11%)	5.87 ± 0.24 (15%)	7.21 ± 0.53	10^{-2}
	3	6.50 ± 0.20	6.31 ± 0.33	6.27 ± 0.33 (04%)	5.93 ± 0.18 (09%)		
	5	7.01 ± 0.36	6.53 ± 0.26	6.38 ± 0.25 (09%)	5.97 ± 0.21 (15%)		
letterorig	1	6.37 ± 0.10	4.42 ± 0.08	4.11 ± 0.09 (35%)	3.54 ± 0.11 (45%)	5.88 ± 0.31	10^{-2}
	3	6.89 ± 0.20	5.20 ± 0.13	4.86 ± 0.27 (29%)	4.15 ± 0.19 (40%)		
	5	6.74 ± 0.26	5.08 ± 0.10	4.63 ± 0.26 (31%)	4.03 ± 0.13 (40%)		
pendigits	1	3.53 ± 0.12	3.23 ± 0.11	2.91 ± 0.07 (17%)	2.85 ± 0.14 (19%)	2.95 ± 0.21	10^{-2}
	3	3.42 ± 0.17	3.38 ± 0.16	2.97 ± 0.12 (13%)	2.92 ± 0.10 (15%)		
	5	3.50 ± 0.18	3.28 ± 0.18	3.05 ± 0.08 (13%)	3.00 ± 0.09 (14%)		
sensorless	1	1.82 ± 0.02	0.97 ± 0.03	0.32 ± 0.01 (82%)	0.25 ± 0.01 (86%)	0.15 ± 0.02	10^{-1}
	3	1.03 ± 0.16	0.39 ± 0.01	0.29 ± 0.01 (71%)	0.28 ± 0.01 (73%)		
	5	0.90 ± 0.15	0.51 ± 0.22	0.27 ± 0.05 (70%)	0.24 ± 0.03 (73%)		
usps	1	6.13 ± 0.18	6.15 ± 0.21	6.09 ± 0.22 (01%)	5.96 ± 0.21 (03%)	6.31 ± 0.20	10^{-2}
	3	6.53 ± 0.20	6.52 ± 0.19	6.29 ± 0.10 (04%)	6.21 ± 0.25 (05%)		
	5	6.55 ± 0.23	6.44 ± 0.20	6.39 ± 0.06 (02%)	6.21 ± 0.11 (05%)		

has been hot in research in the field of image processing for decades. SR is a well-known, illposed problem, which needs technical skills and knowledge from mathematics and machine learning. Prior methods on SR are mainly based on edge preserving techniques, such as new edge-directed interpolation,[29] soft-decision adaptive interpolation,[30] directional filtering and data-fusion,[31] and modified edge-directed interpolation.[32]

The neighbor-embedding (NE) methods[33,34] set the milestone on the patch-learning-based SR approach. In this approach, each LR patch is approximated as a linear combination of its nearest LR neighbors in a collected dataset, while its HR counterpart can be reconstructed using the corresponding HR neighbors with the same coefficients, based on the nonlinear manifold learning. Although the NE method is simple and practical, it requires a huge dataset (millions of patches) to achieve good reconstruction quality and it is computationally intensive because the k-nearest neighbor (k-NN) algorithm[9,10] is used in searching neighboring patches in a huge dataset. Instead of using the patches extracted directly from natural images, Yang et al.[35] employed sparse coding[36,35] to represent image patches of large size efficiently, which opens the era for sparse coding in the image inverse problems.

The sparse-coding super-resolution (ScSR) approach is a framework in which the HR counterpart of an LR patch can be reconstructed using two learned dictionaries, with the sparse constraint on the coefficients via the following equation:

Table 3 Comparison of regression performances on eight datasets, which can be found at Ref. 28. RF, standard random forest; ARF, alternating regression forest;[25] JMPF: proposed algorithm, JMPF + ARF: our proposed algorithm embedded into ARF. λ is the error scale. The number of randomly chosen hyperplanes $\#\mathcal{H}$ is 3. The percentages in brackets for JMPF and JMPF + ARF are the reduction rates in RMSE compared with the RF algorithm.

Dataset	RF	ARF	JMPF	JMPF + ARF	λ
Delta ailerons	2.970 ± 0.001	2.967 ± 0.006	1.952 ± 0.003 (34%)	1.946 ± 0.002 (34%)	10^{-4}
Delta elevators	2.360 ± 0.002	2.338 ± 0.008	1.635 ± 0.001 (30%)	1.610 ± 0.006 (32%)	10^{-3}
Elevators	0.638 ± 0.001	0.635 ± 0.001	0.619 ± 0.001 (03%)	0.606 ± 0.001 (05%)	10^{-2}
kin8nm	2.622 ± 0.002	2.545 ± 0.003	1.962 ± 0.003 (25%)	1.667 ± 0.005 (36%)	10^{-1}
Price	7.281 ± 0.755	6.663 ± 0.794	5.460 ± 0.627 (25%)	5.234 ± 0.666 (28%)	10^{1}
Pyrim	1.440 ± 0.008	1.042 ± 0.347	1.031 ± 0.017 (28%)	0.631 ± 0.018 (56%)	10^{-1}
Stock	2.878 ± 0.022	2.823 ± 0.038	2.744 ± 0.019 (05%)	2.678 ± 0.021 (07%)	10^{0}
Wiscoin breast cancer	3.669 ± 0.041	3.130 ± 0.044	3.081 ± 0.008 (16%)	3.036 ± 0.023 (17%)	10^{1}

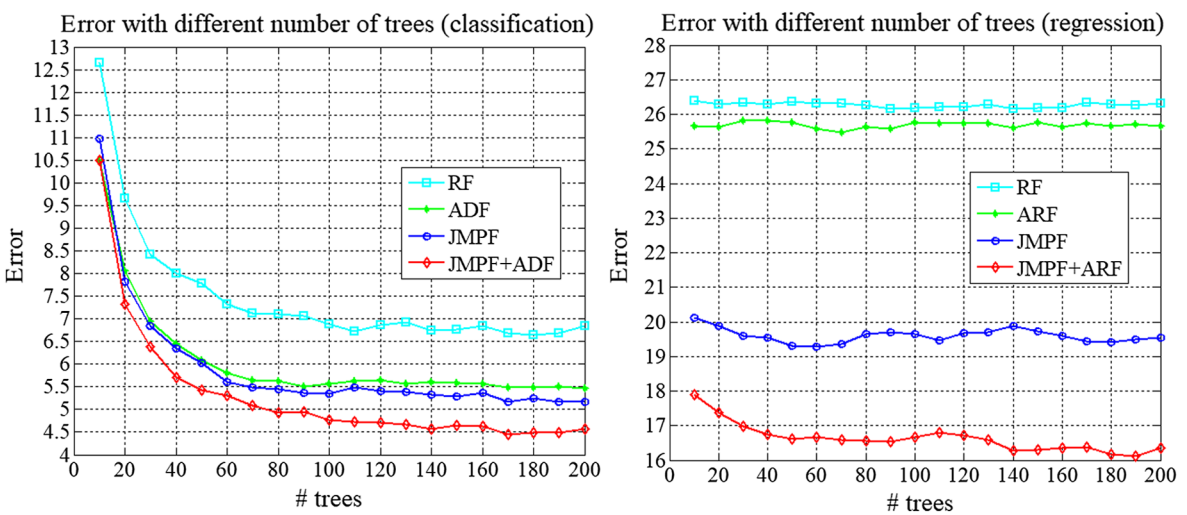

Fig. 4 Performance with different numbers of trees for (a) classification and (b) regression (dataset for classification is letterorig and dataset for regression is kin8nm, error scale: 10^{-2}, the number of hyperplanes $\#\mathcal{H}$ randomly selected in each split node is 3).

$$y \approx D_l, \quad x \approx D_h\alpha, \quad \alpha \in \mathbb{R}^k \quad \text{with } \|\alpha\|_0 \ll k, \tag{17}$$

where k is the number of coefficients. The compact LR and HR dictionaries can be jointly learned with a sparsity constraint, using the following sparse representation:

$$D_h, D_l = \arg\min_{D_h, D_l} \|x - D_h\alpha\|_2^2 + \|y - D_l\alpha\|_2^2 + \lambda\|\alpha\|_0, \tag{18}$$

where y and x are the LR patch and the corresponding HR patch, and D_l and D_h are the LR and HR dictionaries learned from the LR and the corresponding HR patch samples, respectively. The value of n in $\|\alpha\|_n$ is the sparsity factor of the coefficients α. $\|\alpha\|_0$ is l^0-norm, which is the number of nonzero coefficients in α. For each LR patch y of an input LR image Y, the problem of finding the sparse coefficients α can be formulated as follows:

$$\min \|\alpha\|_0 \quad \text{s.t. } \|D_l\alpha - y\|_2^2 \le \varepsilon \tag{19}$$

or

$$\min \|\alpha\|_0 \quad \text{s.t. } \|FD_l\alpha - Fy\|_2^2 \le \varepsilon, \tag{20}$$

where F is a linear or nonlinear feature-extraction operator on the LR patches, which makes the LR patches more discriminative from each other. Typically, F can be chosen as a high-pass filter, and a simple high-pass filter can be obtained, by subtracting the input from the output of a low-pass filter, as in the early work in Ref. 37. In Refs. 14, 38, 15, and 35, first- and second-order gradient operators are employed on up-sampled versions of LR images, then four patches are extracted from

these gradient maps at each location, and they are concatenated to become feature vectors. The four 1-D filters used to extract the derivatives are as follows:

$$\left.\begin{array}{l} F_1 = [-1, 0, 1], F_2 = F_1^T \\ F_3 = [1, 0, -2, 0, 1], F_4 = F_3^T \end{array}\right\}. \tag{21}$$

The ideal regularization term for the sparse constraint on the coefficients α is the l^0-norm (nonconvex), but, based on greedy matching, it still leads to an NP-hard problem. Alternatively, Yang et al.[35] relaxed it to l^1-norm, as shown in the following equation:

$$\min \|\alpha\|_1 \quad \text{s.t. } \|FD_l\alpha - F\mathrm{y}\|_2^2 \le \varepsilon. \tag{22}$$

The Lagrange multiplier provides an equivalent formulation as follows:

$$\min_\alpha \|FD_l\alpha - F\mathrm{y}\|_2^2 + \lambda\|\alpha\|_1, \tag{23}$$

where the parameter λ balances the sparsity of the solution and the fidelity of the approximation to y. However, the effectiveness of sparsity was challenged in Refs. 15 and 39, as to whether real sparsity can help image classification and restoration, or locality property can achieve the same effect. Timofte et al.[14] proposed an anchored neighborhood regression (ANR) framework, which relaxes the sparse decomposition optimization (l^1-norm) of Refs. 38 and 35 to a ridge regression (l^2-norm) problem. Therefore, an important step in the ANR model is the relaxation of the l^1-norm in Eq. (23) to the l^2-norm least-squares minimization constraint, as follows:

$$\min_\alpha \|FD_l\alpha - F\mathrm{y}\|_2^2 + \lambda\|\alpha\|_2. \tag{24}$$

This l^2-norm constraint problem can be solved with a closed-form solution from the ridge regression[40] theory. Based on the Tikhonov regularization/ridge-regression theory, the closed-form solution of the coefficients is given as follows:

$$\alpha = (D_l^T D_l + \lambda I)^{-1} D_l^T F y. \tag{25}$$

We assume that the HR patches share the same coefficient α as their corresponding LR patches, i.e., $x = D_h\alpha$. From Eq. (25), we have

$$x = D_h(D_l^T D_l + \lambda I)^{-1} D_l^T F y. \tag{26}$$

Therefore, the HR patches can be reconstructed by: $x = P_G F\mathrm{y}$, where P_G can be considered a projection matrix, which can be calculated offline, as follows:

$$P_G = D_h(D_l^T D_l + \lambda I)^{-1} D_l^T. \tag{27}$$

Ridge regression allows the coefficients α to be calculated, by multiplying the projection matrix P_G with the extracted feature Fy, as described in Eqs. (26) and (27). More importantly, the projection matrix P_G can be precomputed, and this offline learning enables significant speed-up at the prediction stage.

Timofte et al.[15] further extended the ANR approach to the A+ approach, which learns regressors from all the training samples, rather than from a small quantity of neighbors of the anchor atoms as ANR does. Later, there are numerous variants and extended approaches, based on ANR and A+.[39,41,42,43,44,45,46,47,48] By investigating the ANR model, Li and Lam[39] found that the weights of the supporting atoms can be of different values to represent their similarities to the anchor atom. Based on this idea, the normal collaborative representation (CR) model in ANR is generalized to a weighted model, named as weighted collaborative representation (WCR) model, as follows:

$$\min_\alpha \|FD_l\alpha - Fy\|_2^2 + \|\lambda_{\mathrm{WCR}}\alpha\|_2, \tag{28}$$

where λ_{WCR} is a diagonal matrix. The weights on the diagonal atoms are proportional to their similarities to the anchor atom. Similarly, the closed-form solution for the coefficients can be calculated offline, as follows:

$$\alpha^* = (D_l^T D_l + \lambda_{\mathrm{WCR}})^{-1} D_l^T F\mathrm{y}, \tag{29}$$

and the projection matrix is given as follows:

$$P_G^* = D_h(D_l^T D_l + \lambda_{\mathrm{WCR}})^{-1} D_l^T. \tag{30}$$

The WCR model can further improve the ANR or A+ model in terms of image quality, but it is still time consuming to find the most similar anchor atoms in a dictionary, and this will always hinder it in real-time applications.

Schulter et al.[9] adopted the random forest as a classifier, and the regressors are learned from the patches in the leaf nodes. With the same number of regressors, these random-forest-based methods[9,10,49,50] can perform on a par with the A+ method in terms of accuracy. However, they achieve a significant improvement in speed because the sublinear search in random forests can remarkably reduce the regressors' search complexity.

Recently, deep learning has become another research hotspot, which has been successfully applied to image SR[51–54] and achieved promising performance, particularly in terms of image quality. In Refs. 51 and 52, a convolutional neural-network-based image super-resolution was proposed, in which an end-to-end mapping between LR and HR images is learned through a deep convolutional neural network. Reference 53 presented an SR approach with very deep networks at extremely high learning rates. The deep network's convergence rate is sped up by means of residual learning. Meanwhile, Ref. 54 presented a generative adversarial network (GAN)-based deep residual network model for image SR (SRGAN), in which content loss and adversarial loss are combined as an image perceptual loss function. The proposed deep residual network in Ref. 54 can super-resolve photo-realistic textures from four-times down sampled images, and an extensive mean-opinion-score criterion was proposed to test the perceptual quality gain using SRGAN. Although deep-learning-based approaches can achieve superior performance compared with other SR methods, their heavy computation is always a big obstacle to their extensive applications with real-time requirements, where the graphics processing unit may not be available, such as smart mobile phones.

4.2 *JMPF-Based Image Super-Resolution*

The recent emerging stream[15,55] on single-image SR is to formulate the problem as a clustering-regression problem, which can be solved with machine-learning tools. These approaches are learning-based methods, which attempt to reconstruct an HR image from patches with the help of an external database. These methods first decompose an image into patches, then partition them into clusters. Regressors are then trained for each of the clusters, which generate mappings from an input LR patch's feature to its corresponding HR patch (see Fig. 5). In the testing stage, an LR query image follows the same procedures to cut into patches and to extract features, which are then assigned to the corresponding clusters using the k-NN algorithm[9,10] or random forest.[14,15,19] The respective HR patches are constructed through regressors learned for the clusters (see Fig. 6). This kind of clustering-regression algorithms, based on random forest,[14,15,19] has achieved state-of-the-art performance in single image SR, both in terms of accuracy and efficiency because of the use of ensemble learning and

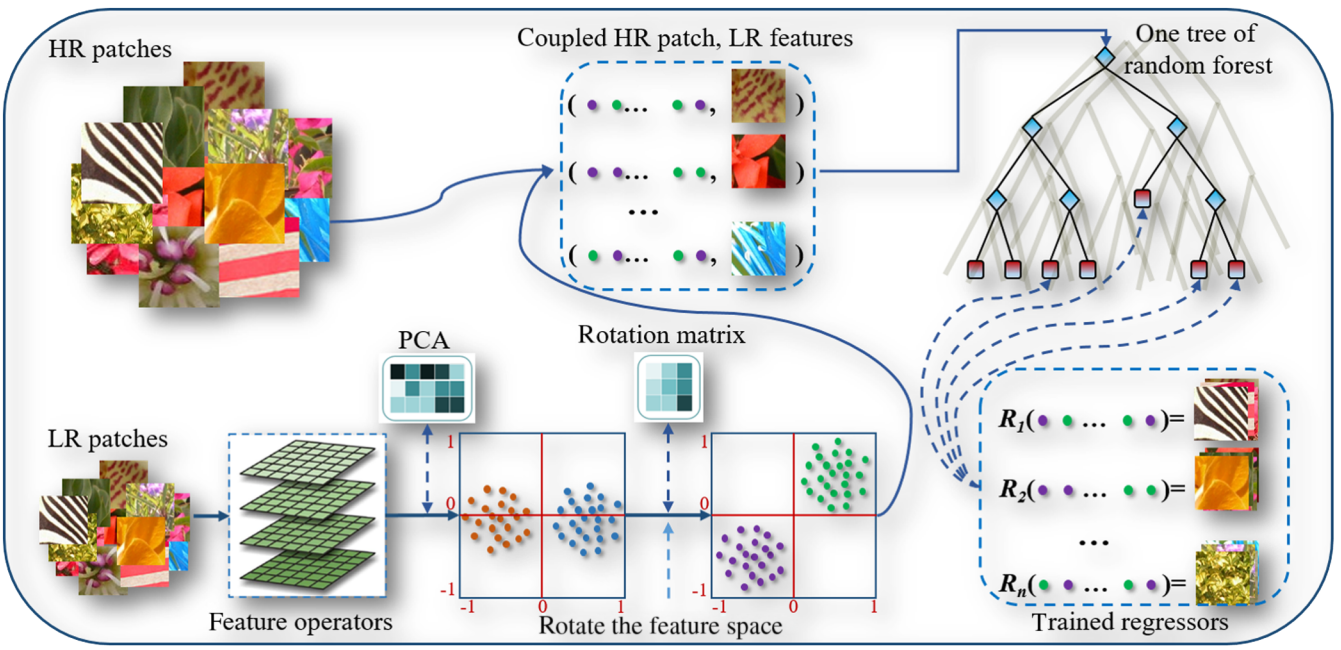

Fig. 5 An overview of the training process of the JMPF-based method for image SR.

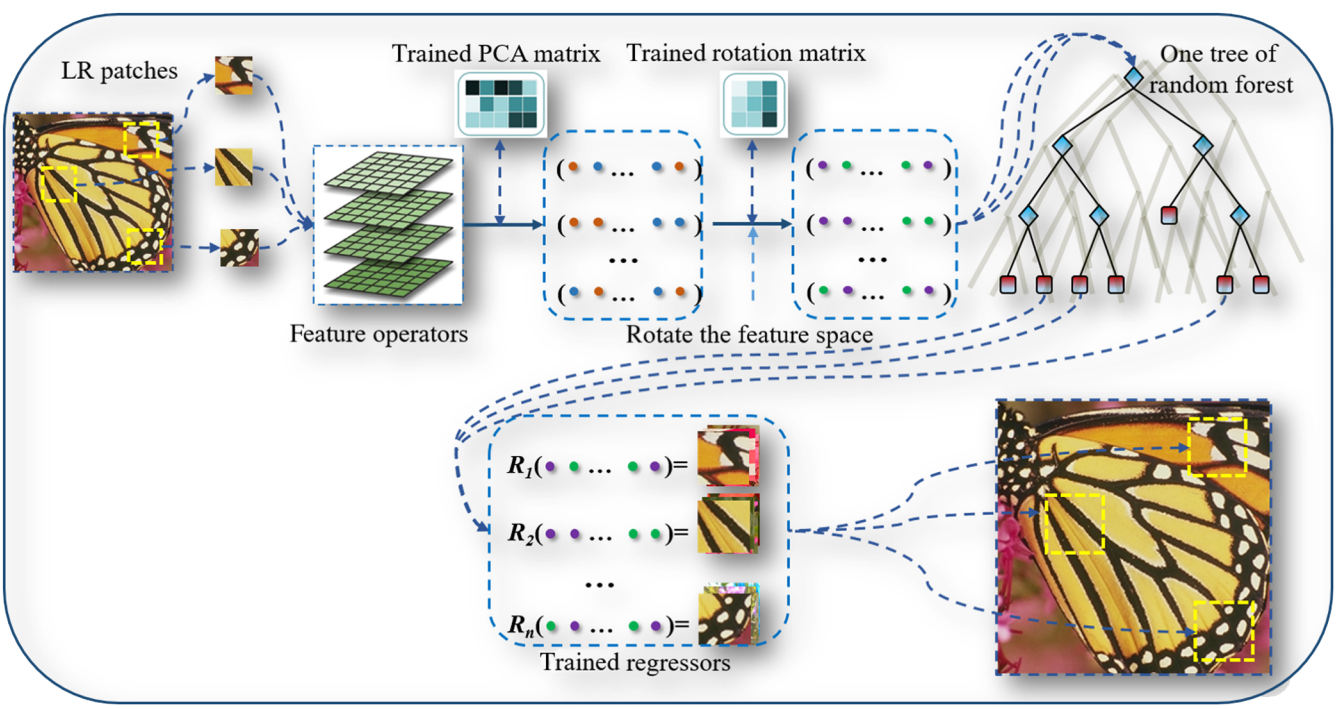

Fig. 6 An overview of the testing process of the JMPF-based method for image SR.

sublinear search. As JMPF achieves promising results on both classification and regression tasks, it can be employed for image SR for better performances.

An overview of the training and testing processes of the proposed JMPF-based image SR method is shown in Figs. 5 and 6, respectively. In our method, the first- and second-order gradients are extracted as features from each patch, followed by PCA for dimensionality reduction. These features are then rotated into a more compact, preclustered feature space. Finally, all the thresholds are directly set to the inherent zero-center hyperplanes when training the random forest, and similar to other algorithms, the regressors at the leaf nodes are computed using the rigid regression algorithms. This approach is named as JMPF-based image SR method.

4.3 *Working Processes of JMPF-Based Image Super-Resolution*

JMPF has shown its better performance for clustering and classification than other random forest methods. As image SR can be considered as a clustering/classification problem, using JMPF is likely to result in better performance. The image SR training and testing processes of our proposed JMPF-based method are described in Algorithms 1 and 2, respectively.

4.4 *Experimental Results on JMPF-Based Image Super-Resolution*

In this section, we evaluate our image SR algorithm on some standard image SR datasets, including Set 5, Set14, and B100,[56] and compare it with a number of classical or state-of-the-art methods. These include conventional bicubic interpolation, sparse representation SR (Zeyde),[38] ANR,[14] A+,[15] standard random forest (RF),[9] and ARFs.[9] We set the same parameters for all the random-forest-based algorithms: the number of trees in the random forest is 10, and the maximum depth of each tree is 15.

Experimental results are shown in Tables 4 and 5, where JMPF is our proposed JMPF-based image SR method, and JMPF$^-$ is a trimmed version, such that the thresholds for the split nodes are not the inherent zero-center hyperplanes but set by the standard random-forest bagging strategy. We use the same training images (91 images) for all the algorithms as previous works[14,38,15,9] do. However, for JMPF$^+$, 100 more images from the General-100 dataset[42] are used, so as to check whether more training samples can further improve our proposed algorithm.

Table 4 tabulates the performances, in terms of the average peak signal-to-noise ratio (PSNR) scores, of our proposed algorithm and other image SR methods, on the three datasets with different magnification factors. For the Set5 and Set14 datasets, with different magnification factors, our proposed JMPF-based algorithm can achieve a comparable performance to other recent state-of-the-art methods, such as A+ and ARF. As those random-forest-based algorithms may not be stable on small datasets, when evaluation works on extensive datasets, such as B100, our proposed algorithm JMPF can stably outperform A+ and ARF for all magnification factors (×2, ×3, and ×4). Moreover, the objective quality metrics on PSNR also show that the JMPF algorithm can achieve a better performance when more samples are used for training, as shown from JMPF$^+$ in Table 4. Table 5 shows more details of the performances in datasets Set5.

To compare the visual quality of our proposed JMPF-based SR algorithm to other methods, Fig. 7 shows the reconstructed HR images using different methods. Some regions in the reconstructed images are also enlarged, so as to show the details in the images. In general, our proposed method can produce better quality images, particularly in areas with rich texture, which verifies the feature discrimination of the proposed JMPF scheme.

Algorithm 1 JMPF-based image SR training process.

Input: $\{x_i^l, x_i^h\}_{i=1}^N$: training LR-HR patch pairs, N is the number of training samples.

Output: The random forest and ridge regression projection matrices: $\wp = (P_1, \ldots, P_T)$, in leaf nodes, where T is the number of regressors; the PCA projection matrix $\mathcal{M}$ and the rotation matrix $\mathcal{R}$.

1: Discriminative features calculated from patch images based on first and second-order (horizontal and vertical) gradients; ⇒ {Eq. (21)}

2: Apply PCA on the features to compute the PCA projection matrix $\mathcal{M}$;

3: Train a JMPF-based random forest by clustering the feature data, whose dimensionality is reduced by using PCA, into feature-space vertices, which can rotate the feature space into a compact preclustered feature space, at the same time obtain the rotation matrix $\mathcal{R}$; ⇒ {Eq. (11)}

4: Train ridge regression projection matrices: $\wp = (P_1, \ldots, P_T)$, from the LR-HR patch pairs in all the leaf nodes. ⇒ {Eq. (27)}

Algorithm 2 JMPF-based image SR testing stage.

Input: Testing LR image I^l, the trained JMPF-based random forest and ridge regression projection matrices: $\wp = (P_1, \ldots, P_T)$ in leaf nodes; the trained PCA projection matrix $\mathcal{M}$ and the trained rotation matrix $\mathcal{R}$.

Output: Super-resolved image I^h.

1: Extract discriminative features for all the patches of image I^l; ⇒ {Eq. (21)}

2: Perform feature dimensionality reduction via the PCA projection matrix $\mathcal{M}$;

3: Rotate feature space into a compact preclustered feature space via the rotation matrix $\mathcal{R}$;

4: For LR patches from image I^l, based on their features, classify them into the corresponding leaf nodes of the trained random forest;

5: Produce I^h through all the image patches from image I^l by ridge regression with the trained projection matrices: $\wp = (P_1, \ldots, P_T)$ from the training stage. ⇒ {Eq. (26)}

Table 4 Results of the proposed method, compared with state-of-the-art methods on 3 datasets, in terms of PSNR (dB), with three different magnification factors (×2, ×3, ×4).

Dataset	scale	bicubic	Zeyde[38]	ANR[14]	A+[15]	RF[9]	ARF[9]	JMPF$^-$	JMPF	JMPF$^+$
Set5	×2	33.66	35.78	35.83	36.55	36.52	36.65	36.53	36.59	36.70
	×3	30.39	31.92	31.93	32.59	32.44	32.53	32.51	32.59	32.67
	×4	28.42	29.74	29.74	30.28	30.10	30.17	30.14	30.17	30.24
Set14	×2	30.23	31.81	31.80	32.28	32.26	32.33	32.27	32.32	32.42
	×3	27.54	28.68	28.66	29.13	29.04	29.10	29.12	29.13	29.24
	×4	26.00	26.88	26.85	27.33	27.22	27.28	27.29	27.30	27.37
B100	×2	29.32	30.40	30.44	30.78	31.13	31.21	31.16	31.23	31.31
	×3	27.15	27.87	27.89	28.18	28.21	28.26	28.26	28.30	28.37
	×4	25.92	26.51	26.51	26.77	26.74	26.77	26.78	26.81	26.87

Table 5 Detailed results of the proposed method, compared with state-of-the-art methods on the dataset Set5, in terms of PSNR (dB) using three different magnification factors (×2, ×3, ×4).

Set5(×2)	Bicubic	Zeyde[38]	ANR[14]	A+[15]	RF[9]	ARF[9]	JMPF$^-$	JMPF	JMPF$^+$
Baby	37.05	38.22	38.42	38.52	38.47	38.48	38.40	38.45	38.45
Bird	36.82	39.91	40.03	41.06	40.98	41.15	40.82	40.99	41.11
Butterfly	27.43	30.64	30.54	32.02	32.27	32.66	32.58	32.50	32.79
Head	34.85	35.62	35.72	35.82	35.69	35.73	35.68	35.73	35.78
Woman	32.14	34.53	34.53	35.31	35.19	35.24	35.15	35.28	35.38
Average	33.66	35.78	35.85	36.55	36.52	36.65	36.53	36.59	36.70
				Set5(×3)					
Baby	33.91	35.13	35.13	35.23	35.25	35.15	35.11	35.16	35.14
Bird	32.58	34.62	34.63	35.53	35.23	35.31	35.25	35.46	35.49
Butterfly	24.04	25.93	25.92	27.13	27.00	27.39	27.46	27.48	27.73
Head	32.88	33.61	33.64	33.82	33.73	33.73	33.72	33.79	33.76
Woman	28.56	30.32	30.31	31.24	30.98	31.08	31.03	31.06	31.24
Average	30.39	31.92	31.93	32.59	32.44	32.53	32.51	32.59	32.67
				Set5(×4)					
Baby	31.78	33.13	33.07	33.3	33.26	33.16	33.09	33.12	33.12
Bird	30.18	31.75	31.82	32.5	32.21	32.26	32.27	32.33	32.47
Butterfly	22.10	23.67	23.58	24.4	24.32	24.56	24.55	24.44	24.63
Head	31.59	32.23	32.34	32.5	32.35	32.37	32.35	32.45	32.47
Woman	26.46	27.94	27.88	28.6	28.38	28.48	28.44	28.50	28.53
Average	28.42	29.74	29.74	30.28	30.10	30.17	30.14	30.17	30.24

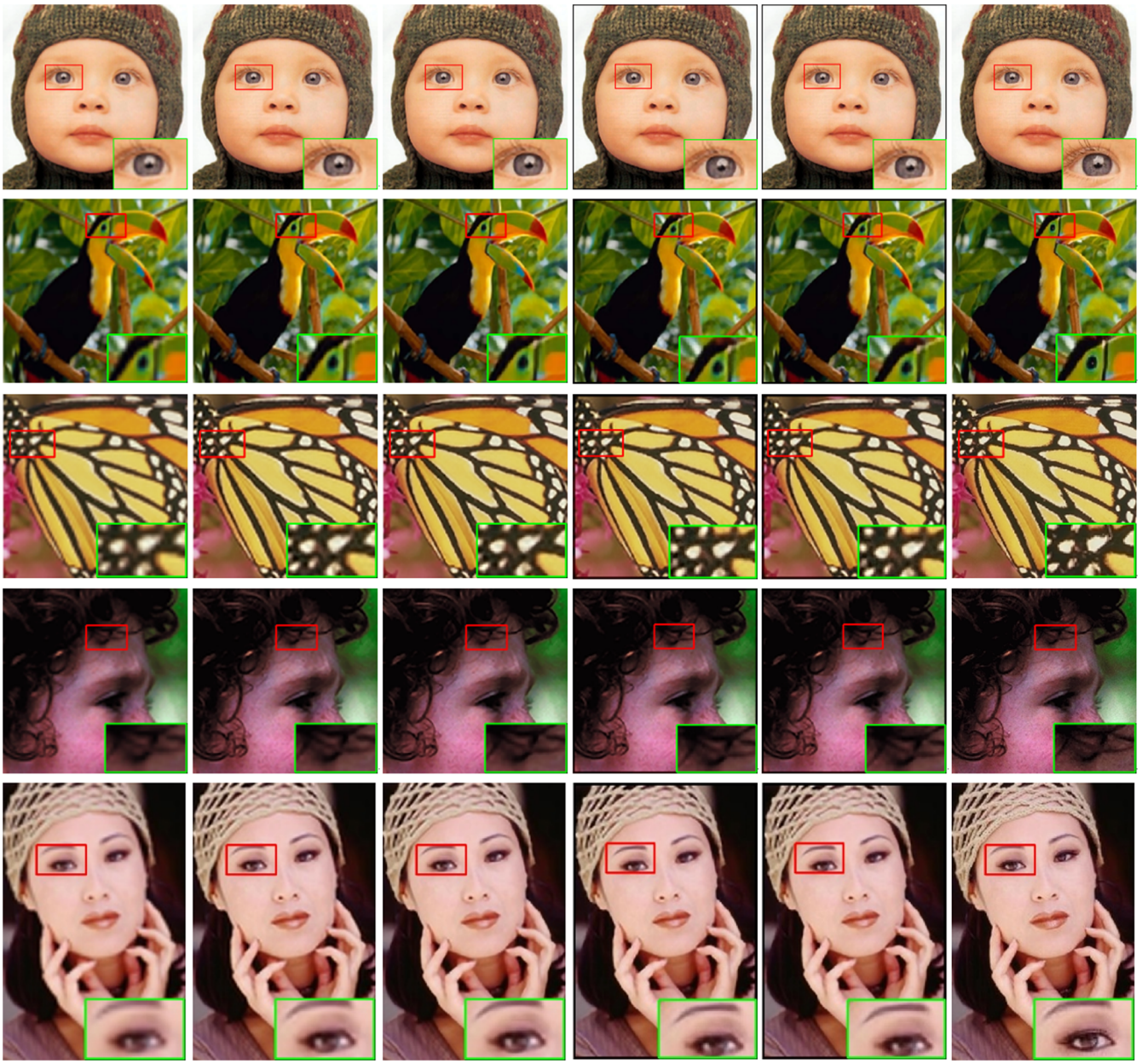

Fig. 7 Super-resolved (×3) images from Set5: (a) bicubic, (b) ANR,[14] (c) A+,[15] (d) ARF,[9] (e) proposed algorithm JMPF, and (f) ground truth. The results show that our JMPF-based algorithm can produce more details.

5 Conclusions

In this paper, we have proposed a random-forest scheme, namely the JMPF scheme, which rotates the feature space into a compact, clustered feature space, by jointly maximizing the purity of all the feature-space vertices. In the preclustered feature space, orthogonal hyperplanes can work effectively in the split nodes of a decision tree, which can improve the performance of the trained random forest. Compared with the standard random forests and the recent state-of-the-art variants, such as ADFs and ARFs, our proposed random-forest method inherits the merits of random forests (fast training and testing, multiclass capability, etc.), and yields promising results on both classification and regression tasks. Experiments have shown that our method achieves an average improvement of about 20% for classification and regression on publicly benchmarked datasets. Furthermore, our proposed scheme can integrate with other methods, such as ADF and ARF, to further improve the performance. The source code of our algorithm is available to download at: https://github.com/HarleyHK/JMPF.

We have also applied JMPF to single-image SR specifically. We tackle image SR as a clustering-regression problem, and focus on the clustering stage, which happens at the split nodes of each decision tree. By employing the JMPF strategy, we rotate the feature space into a preclustered feature space, which can cluster samples into different subspaces more compactly in an unsupervised

problem. The compact preclustered feature space can provide the optimal thresholds for the split nodes in decision trees, which are the zero-centered orthogonal hyperplanes. Our experimental results on intensive image benchmark datasets, such as B100, show that the proposed JMPF-based image SR approach can consistently outperform recent state-of-the-art algorithms, in terms of PSNR and visual quality.

References

1. Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.* **9**(7), 1545–1588 (1997).
2. L. Breiman, "Random forests," *Mach. Learn.* **45**(1), 5–32 (2001).
3. J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: a new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(10), 1619–1630 (2006).
4. J. Gall and V. Lempitsky, "Class-specific Hough forests for object detection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2009).
5. V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1867–1874 (2014).
6. S. Ren et al., "Face alignment at 3000 fps via regressing local binary features," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1685–1692 (2014).
7. H. Li et al., "Cascaded face alignment via intimacy definition feature," *J. Electron. Imaging* **26**(5), 053024 (2017).
8. F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *Conf. on Neural Information Processing Systems (NIPS)*, Vol. 2, p. 4 (2006).
9. S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3791–3799 (2015).
10. J. Salvador and E. Pérez-Pellitero, "Naive bayes super-resolution forest," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 325–333 (2015).
11. C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discovery* **2**(2), 121–167 (1998).
12. N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, New York (2000).
13. C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.* **20**(3), 273–297 (1995).
14. R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 1920–1927 (2013).
15. R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian Conf. on Computer Vision (ACCV)*, pp. 111–126, Springer (2014).
16. Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Int. Conf. on Machine Learning (ICML)*, Vol. 96, pp. 148–156 (1996).
17. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Ann. Stat.* **29**(5), 1189–1232 (2001).
18. Y. Gong et al., "Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(12), 2916–2929 (2013).
19. H. Jégou et al., "Aggregating local descriptors into a compact image representation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3304–3311, (2010).
20. F. Yu et al., "Circulant binary embedding," in *Int. Conf. on Machine Learning (ICML)*, pp. 946–954 (2014).
21. P. H. Schönemann, "A generalized solution of the orthogonal Procrustes problem," *Psychometrika* **31**(1), 1–10 (1966).
22. J. Wang et al., "Optimized Cartesian k-means," *IEEE Trans. Knowl. Data Eng.* **27**(1), 180–192 (2015).
23. B.-F. Wu et al., "Active appearance model algorithm with K-nearest neighbor classifier for face pose estimation," *J. Marine Sci. Technol.* **22**(3), 285–294 (2014).
24. S. Schulter et al., "Alternating decision forests," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 508–515 (2013).
25. S. Schulter et al., "Alternating regression forests for object detection and pose estimation," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 417–424 (2013).
26. R.-E. Fan et al., "LIBLINEAR: a library for large linear classification," *J. Mach. Learn. Res.* **9**, 1871–1874 (2008).
27. A. Frank and A. Asuncion, "UCI machine learning repository," http://archive.ics.uci.edu/ml (2010).
28. L. Torgo and A. Asuncion, "Regression data sets," http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html (2014).
29. X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Trans. Image Process.* **10**(10), 1521–1527 (2001).
30. X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," *IEEE Trans. Image Process.* **17**(6), 887–896 (2008).
31. L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE Trans. Image Process.* **15**(8), 2226–2238 (2006).
32. W.-S. Tam, C.-W. Kok, and W.-C. Siu, "Modified edge-directed interpolation for images," *J. Electron. Imaging* **19**(1), 013011 (2010).
33. H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 275–282 (2004).
34. M. Bevilacqua et al., "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *British Machine Vision Conf. (BMVC)* (2012).
35. J. Yang et al., "Image super-resolution via sparse representation," *IEEE Trans. Image Process.* **19**(11), 2861–2873 (2010).
36. D. Zhang and J. He, "Hybrid sparse-representation-based approach to image super-resolution reconstruction," *J. Electron. Imaging* **26**(2), 023008 (2017).
37. W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.* **22**(2), 56–65 (2002).
38. R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Int. Conf. on Curves and Surfaces*, pp. 711–730, Springer (2010).
39. H. Li and K.-M. Lam, "Fast super-resolution based on weighted collaborative representation," in *IEEE the 19th Int. Conf. on Digital Signal Processing (DSP)*, pp. 914–918 (2014).
40. A. N. Tikhonov, V. I. A. k. Arsenin, and F. John, *Solutions of Ill-Posed Problems*, Winston, Washington, DC (1977).
41. R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1865–1873 (2016).
42. J. Jiang et al., "Single image super-resolution via locally regularized anchored neighborhood regression and nonlocal means," *IEEE Trans. Multimedia* **19**(1), 15–26 (2017).
43. K. Zhang et al., "Learning multiple linear mappings for efficient single image super-resolution," *IEEE Trans. Image Process.* **24**(3), 846–861 (2015).
44. Y. Zhang et al., "Adaptive local nonparametric regression for fast single image super-resolution," in *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE (2015).
45. Y. Zhang et al., "Image super-resolution based on dictionary learning and anchored neighborhood regression with mutual incoherence," in *IEEE Int. Conf. on Image Processing (ICIP)*, Québec City, QC, Canada, pp. 591–595 (2015).
46. Y. Zhang et al., "CCR: clustering and collaborative representation for fast single image super-resolution," *IEEE Trans. Multimedia* **18**(3), 405–417 (2016).
47. D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," *Comput. Graphics Forum* **34**(2), 95–104 (2015).
48. E. Agustsson, R. Timofte, and L. Van Gool, "Regressor basis learning for anchored super-resolution," in *IEEE Int. Conf. on Pattern Recognition (ICPR)*, Cancun, Mexico (2016).
49. J.-J. Huang and W.-C. Siu, "Learning hierarchical decision trees for single image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.* **27**, 937–950 (2015).
50. J.-J. Huang et al., "Fast image interpolation via random forests," *IEEE Trans. Image Process.* **24**(10), 3232–3245 (2015).
51. C. Dong et al., "Learning a deep convolutional network for image super-resolution," in *European Conf. on Computer Vision (ECCV)*, pp. 184–199, Springer (2014).
52. C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conf. on Computer Vision (ECCV)*, pp. 391–407, Springer (2016).
53. J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654 (2016).
54. C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).
55. C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 561–568 (2013).
56. D. Martin et al., "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. of Eighth IEEE Int. Conf. on Computer Vision (ICCV)*, Vol. 2, pp. 416–423 (2001).

Hailiang Li received his MSc degree from Department of Automation of Xiamen University, China, in 2004. He is studying in the Hong Kong Polytechnic University as a part-time PhD student. His research interests include image super-resolution, face alignment, and Bayesian inference. Currently, he works as a software engineer with Hong Kong Applied Science and Technology Research Institute (ASTRI) and his work is related to image processing, computer vision, and machine learning.

Kin-Man Lam is a professor at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. He is the VP-Publications of the Asia-Pacific Signal and Information Processing Association (APSIPA). He serves as an associate editor of Digital Signal Processing and APSIPA Trans. on Signal and Information Processing, and an editor of HKIE Trans. His research interests include human face recognition, image and video processing, and computer vision.

Dong Li received his BEng degree in computer science and his MEng degree in computer science from Tianjin University, Tianjin, in 2006 and 2009, respectively, and his PhD from Hong Kong Polytechnic University, Hong Kong, in 2014. Currently, he is an assistant professor with the Guangdong University of Technology. His research interests include topics in the fields of computer vision, pattern recognition, and image processing, such as feature matching, face recognition, and color correction.