Supervised Sparse Coding with Decision Forest

Yan Huang, Yuhui Quan* and Tao Liu

***Abstract*—By jointly conducting sparse coding and classifier training, supervised sparse coding has shown its effectiveness in a variety of recognition tasks. However, the existing supervised sparse coding methods often consider linear classification, which limits their discrimination in handling highly-nonlinear data. In this paper, we propose a new supervised sparse coding model by incorporating decision tree classifiers. Since decision trees can well deal with the non-linear properties of data, the introduction of decision trees to sparse coding can noticeably improve the discrimination of coding. Meanwhile, sparse coding is able to produce sparse de-correlated features that decision tree is favor of. For further improvement, we close the loop of sparse coding and decision tree learning with an ensemble framework, which alternatively learns a dictionary for sparse coding and a decision tree for classification. The resulting series of decision trees as well as series of dictionaries are used to construct a decision forest for classification. The proposed method was applied to face recognition and scene classification, and the experimental results have demonstrated its power in comparison with recent supervised sparse coding methods.**

***Index Terms*—sparse coding, dictionary learning, decision tree, decision forest, classification.**

I. Introduction

In recent years, sparse coding has become one of the most popular technologies for data analysis. By finding the most succinct yet effective representation of data in the coding space, sparse coding is capable of discovering and capturing the intrinsic subspaces of data. With such a capability, sparse coding has been widely used in pattern recognition, such as feature selection [1], subspace clustering [2], image analysis [3] and classification [4–7].

Given a set of input signals, the aim of the sparse coding is to represent each signal by the linear combination of a few atoms in a fixed or learned dictionary. The coefficients of the linear combination are expected to be sparse, *i.e.*, as few atoms as possible are selected to represent the signal, and thus the coefficients are also referred to as sparse codes. Generally, the sparse coding problem, *e.g.* the very popular method K-SVD [8], can be formulated as:

$$\min_{\boldsymbol{D},\boldsymbol{C}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{C}\|_F^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{c}_i\|_0 \leq T, \qquad (1)$$

where $\boldsymbol{Y} \in R^{Q\times P}$ is a data matrix consisting of P signals, $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P] \in R^{M\times P}$ is the coding matrix which collects the corresponding sparse codes of $\boldsymbol{Y}$ as columns, and $\boldsymbol{D} \in R^{Q\times M}$ is the dictionary to be learned for maximizing the efficiency of sparse coding. The ℓ_0 pseudo norm $\|\cdot\|_0$ serves as a sparsity measure by counting the number of nonzero entries, and the threshold T controls the sparsity degree as well as the dimensionality of the coding results.

The above sparse coding model only minimizes the data reconstruction error of coding, which may make the results short of discriminability in classification tasks. To enhance the discrimination of the features from sparse coding, the supervised sparse coding approaches, *e.g.* [9–11], joint the sparse coding process and the classification process by enforcing some misclassification penalties on sparse codes when learning dictionaries. In the existing approaches, the linear classification is often used, which limits the power of these approaches to handle the data with highly-nonlinear properties, *e.g.* the linear classifier used in [9, 11] is not suitable for classifying nonlinearly-separable data, and the Fisher discriminant used in [12] is not suitable for characterizing mixtures of Guassians.

To further improve the performance of supervised sparse coding in classification, in this paper we propose a new supervised sparse coding method by incorporating decision tree classifiers in an ensemble framework, which alternatively conducts the sparse coding with dictionary learning and the decision tree construction. The basic idea is that the sparse codes under the learned dictionary are used as attributes for training a decision tree in the current stage, and the important attributes are identified during the construction of the decision tree for learning a better dictionary in the next stage of sparse coding. Repeating such a process results in multiple learned dictionaries as well as decision trees, which are then used to construct a sparse coding based decision forest via voting. The proposed method is applied to face recognition and scene classification, and the experimental results have shown the effectiveness of the proposed method.

The benefits of incorporating sparse coding and decision tree are two-way. On the one hand, by generating a tree model with an iterative attribute splitting process, the decision tree can well handle the nonlinear properties of data in classification. Moreover, the decision tree enjoys both the discriminability and the robustness to deal with irrelevant features [13]. Thus, by utilizing the feedback from the decision tree, the effectiveness of the sparse coding process can be significantly improved. On the other hand, the construction of a decision tree is actually to divide the feature space with disjoint hyper-rectangles. As a result, the lower correlation of the dimensions of input signals have in the feature space, the shallower and the more efficient a decision tree becomes. Fortunately, the sparse coding with dictionary learning can effectively de-correlate the input data in the coding space, as the data are aligned to their subspaces in sparse representations with the learned dictionary atoms as the coordinate axes. In other words, the dimensions of the features from sparse

All the authors are from School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China. Yuhui Quan would like to thank the support by National Natural Science Foundation of China (Grant No. 61602184, 61872151), Natural Science Foundation of Guangdong Province (Grant No. 2017A030313376), Science and Technology Program of Guangzhou (Grant No. 201707010147), and Fundamental Research Funds for the Central Universities (x2js-D2181690). Asterisk denotes the corresponding author.

coding have lower the correlation than those of the original ones. Therefore, the use of sparse coding for preprocessing can noticeably improve the compactness and efficiency of the decision tree. Such a benefit is illustrated in Fig. I.

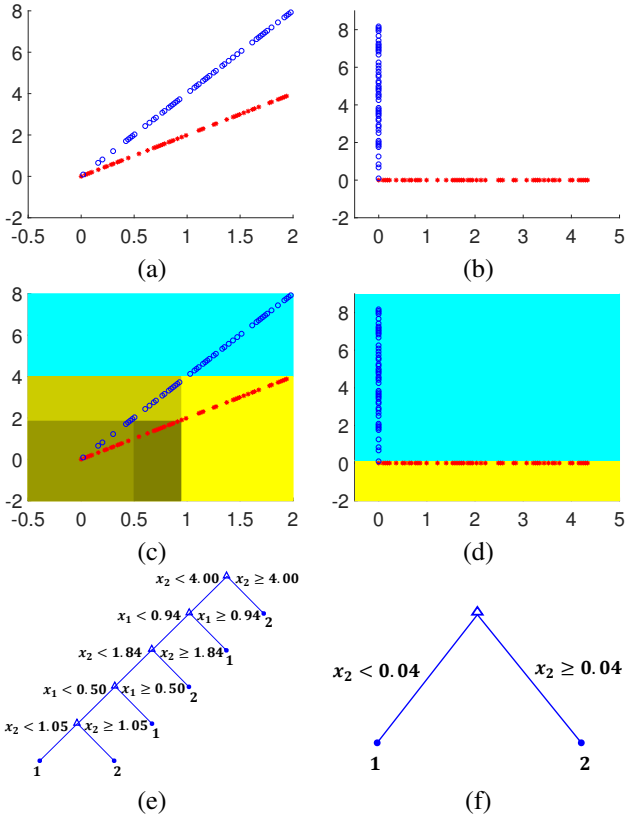

Fig. 1. The motivation of combining sparse coding and decision tree. (a) The raw data. (b) The sparse coding results using model (1) on the data in (a) with two dictionary atoms and sparsity level $T = 1$, where each axis corresponds to one dictionary atom. (c) Many rectangles are needed to well capture the decision boundary of the two classes of data in (a). (d) Only two rectangles are able to perfectly capture the decision boundary of the two classes of data in (b). (e) The decision tree constructed from (a). (f) The decision tree constructed from (b) which is much shallower than that of (e).

II. Our method

A. Basic idea and framework

The real data often has high correlations among their dimensions due to redundancy, *e.g.*, two pixels in an image patch tend to show positive correlations in their intensities. As discussed in Fig. I, the sparse coding can yield more succinct representation with lower correlation, which is beneficial to the improvement of the efficiency of the decision tree. On the other hand, when dealing with non-linear data, the reconstructive sparse coding cannot guarantee obtaining a linearly-separable representation from the data. Existing methods such as [9, 11, 14] incorporate linear classifiers into sparse coding models to induce the linear separability in the results. Yet, these methods cannot guarantee that the coding results are linearly separable. This problem can be remedied by introducing some nonlinear classifiers.

Inspired by the two aspects above, we unify the sparse coding and the decision tree construction in a computational framework. Basically, the sparse coding module (*i.e.* sparse encoder) and decision tree construction module (*i.e.* tree constructor) are alternatively conducted to iteratively improve the performance of each module. For further improvement, we borrowed the idea of ensemble learning [15] to construct a powerful framework, which integrates multiple learners to obtain better prediction. At each stage of our proposed framework, with the sparse codes fed by the sparse encoder, the tree constructor learns a compact decision tree, and then the tree constructor feeds back the classification results to the sparse encoder, including the misclassified data as well as the effectiveness score of each dimension of the sparse representation (*i.e.* the discrimination of a dictionary atom) in classification. According to the feedback, the sparse encoder re-samples a set of training data to learn a new dictionary with some good atoms inherited from the previous stage, and then feeds forward the sparse code of the selected data under the learned dictionary to the tree constructor. Such a process is repeated until the classification accuracy is sufficiently high or it reaches the maximal iteration number. After the iteration, the learned dictionaries and the constructed decision trees are combined to generate a sparse coding based decision forest. In the next, we will detail each step of the proposed framework.

B. Training stage

Let $\{(\boldsymbol{y}_i, t_i)\}_{i=1}^{P}$ denote the given training samples, where $\boldsymbol{y}_i \in R^Q$ is the feature vector, and $t_i \in \{1, 2, \ldots, K\}$ is the class label for total K classes. For convenience, let $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_P]$ and $\boldsymbol{t} = [t_1; \ldots; t_P]$. In the training stage, the sparse coding phase and decision tree construction phase are alternatively iterated. At the nth iteration, the learned dictionary, the corresponding coding matrix, and the constructed decision tree are denoted by $\boldsymbol{D}^{(n)}$, $\boldsymbol{C}^{(n)}$, and $\mathcal{T}^{(n)}$ respectively. The total number of iterations is denoted by N. Given a feature $\boldsymbol{x} \in \mathcal{R}^M$ with proper dimension, a decision tree $\mathcal{T}$ predicts the label $l \in \{1, 2, \ldots, K\}$ of $\boldsymbol{x}$ by $l = \mathcal{T}(\boldsymbol{x})$.

1) Sparse coding: In the initial sparse coding phase (*i.e.* $n = 1$), we conduct sparse coding on the input data $\boldsymbol{Y}$ by solving the minimization of problem (1), that is

$$[\boldsymbol{D}^{(1)}, \boldsymbol{C}^{(1)}] = \underset{\boldsymbol{D},\boldsymbol{C}}{\operatorname{argmin}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{C}\|_F^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{c}_i\|_0 \leq T. \tag{2}$$

There are many effective numerical solvers available for this problem. In this paper, we use an efficient alternating iterative scheme, which iteratively updates the sparse code $\boldsymbol{C}$ via the OMP algorithm [16] and calculates the dictionary atoms via the proximal method [17]. In the later phases (*i.e.* $n \geq 2$), with the feedbacks (*i.e.* discriminative atoms $\boldsymbol{D}_0^{(n-1)}$ and misclassified samples $\boldsymbol{Y}_0^{(n-1)}$) from the decision tree $\mathcal{T}^{(n-1)}$, we conduct the sparse coding as follows:

$$[\boldsymbol{D}^{(n)}, \boldsymbol{C}^{(n)}] = \underset{\boldsymbol{D},\boldsymbol{C}}{\operatorname{argmin}} \|[\boldsymbol{Y}, \beta\boldsymbol{Y}_0^{(n-1)}] - [\boldsymbol{D}_0^{(n-1)}, \boldsymbol{D}]\boldsymbol{C}\|_F^2, \quad \text{subject to} \quad \forall i, \|\boldsymbol{c}_i\|_0 \leq T, \tag{3}$$

where $\beta \in \mathcal{R}$ is a weight. Note that $\boldsymbol{Y}^{(n)} = [\boldsymbol{Y}, \beta\boldsymbol{Y}_0^{(n-1)}]$ equals to re-sampling the samples from $\boldsymbol{Y}$ that is misclassified

in $\mathcal{T}^{(n-1)}$ and β is the weight for controlling the emphasis on the misclassified samples in the current phase. This problem can be solved by a simple modification on the aforementioned alternating iterative scheme, where we do not update the fixed dictionary atoms during the dictionary learning process.

At the end of each sparse coding phase, we obtain the dictionary $\boldsymbol{D}^{(n)}$ and sparse coefficient matrix $\boldsymbol{C}^{(n)}$. Then we store $\boldsymbol{D}^{(n)}$ and use $\boldsymbol{C}^{(n)}$ to train the decision tree $\mathcal{T}^{(n)}$.

2) Tree construction: In the tree constructor phase, we use the sparse code matrix $\boldsymbol{C}^{(n)}$ of $\boldsymbol{Y}^{(n)}$ from the sparse encoder as input features, as well as the corresponding label vector $\boldsymbol{t}^{(n)}$ for supervision, to train a decision tree $\mathcal{T}^{(n)}$. In other words, we view each atom of the learned dictionary $\boldsymbol{D}^{(n)}$ as an attribute and employ $\boldsymbol{C}^{(n)}$ as the values on these attributes. Formally, we write this process as

$$\mathcal{T}^{(n)} = fittree(\boldsymbol{C}^{(n)}, \boldsymbol{t}^{(n)}), \tag{4}$$

where $fittree$ denotes a decision tree constructor. Regarding this task, there are many efficient approaches available. For simplicity, we choose the traditional approach CART [18] in this paper. Other new approaches can also be used for possible improvement. Briefly speaking, the key in the decision tree construction is the criterion of the node splitting on attributes. In CART, a binary classification decision tree is generated from the input variables by splitting the nodes with the Gini index g defined by

$$g(\boldsymbol{p}) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2, \tag{5}$$

where $\boldsymbol{p} = [p_1, \cdots, p_K]$ and p_k is the probability of kth class on a certain node. In essence, the Gini index g measures the impurity of an attribute.

After the decision tree $\mathcal{T}^{(n)}$ is obtained, we can get two types of feedbacks, including the misclassified samples in $\mathcal{T}^{(n)}$ and the intermediate attribute splitting results. The misclassified samples are collected as $\boldsymbol{Y}_0^{(n)}$ and used for the re-sampling in the next sparse coding phase. The attribute splitting results encode much information about the discriminability of the learned dictionary $\boldsymbol{D}^{(n)}$, and we utilize the information to collect discriminative dictionary atoms as $\boldsymbol{D}_0^{(n)}$ and feed back them to the sparse encoder, which is done by:

1) The first S attributes (*i.e.* dictionary atoms) used in node splitting are selected as the discriminative ones. Since the number of samples used for earlier attribute splitting is larger than that in the latter one, the earlier an attribute is used for splitting, the more discriminative the corresponding dictionary atom is.

2) The attributes which repeat more than R times in splitting test are also regarded as the discriminative ones.

C. Forest generation and test stage

The sparse coding phase and decision tree phase are alternatively iterated until the stopping criterion is satisfied. As a result, we obtain a series of dictionaries $\boldsymbol{D}^{(1)}, \cdots, \boldsymbol{D}^{(N)}$ with improved representative power and a series of decision trees $\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(N)}$ that can form a powerful decision forest. The decision forest is constructed and used as follows. Given a test sample $\boldsymbol{y}_{\text{test}}$, we first conduct a series of sparse coding under $\{\boldsymbol{D}^{(n)}\}_{n=1}^{N}$ by solving

$$\begin{aligned} \boldsymbol{c}_{\text{test}}^{(n)} = \underset{\boldsymbol{c}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y}_{\text{test}} - \boldsymbol{D}^{(n)}\boldsymbol{c}\|_F^2, \\ \text{subject to} \quad \|\boldsymbol{c}\|_0 \leq T, \end{aligned} \tag{6}$$

for $n = 1, \cdots, N$. Each of these problems can be solved by the OMP algorithm [16]. Then the decision trees are assembled into a decision forest $\mathcal{F}$ which is defined by

$$\mathcal{F}(\boldsymbol{y}_{\text{test}}) = \underset{k=1,\cdots,K}{\operatorname{argmax}} \sum_{n=1}^{N} w^{(n)} \delta(\mathcal{T}^{(n)}(\boldsymbol{c}_{\text{test}}^{(n)}) - k). \tag{7}$$

where $\delta(x) = 1$ if $x = 0$ and 0 otherwise, and each decision tree $\mathcal{T}^{(n)}$ is weighted according to its classification accuracy $a^{(n)}$ in training as follows:

$$w^{(n)} = 0.5 * log(a^{(n)}/(1-a^{(n)})). \tag{8}$$

Note that the classification accuracy $a^{(n)}$ is calculated as the ratio between the number of correct predictions and total number of predictions in $\mathcal{T}^{(n)}$.

III. Experiments

Our method was evaluated with face recognition and scene classification. For the convenience of presentation, we denote the proposed method in the following by SCDF (Sparse Coding based Decision Forest). In order to demonstrate the benefits of combining sparse coding and decision tree, we constructed two baseline methods for the comparison with SCDF: (1) the decision tree method used in our method, which is denoted by DeTree; and (2) the decision tree method based on the sparse codes obtained from the original input feature, which can be viewed as one loop in the proposed method and is denoted by SC-DeTree. Besides, we compared our method with Joint [14], DLSI [19], FDDL [12], L0DL [17], K-SVD [8], D-KSVD [11], LC-KSVD [10], EasyDL [20], MCDL [5]. Through all the experiments, the parameters of the baseline methods are finely tuned for fair comparison.

A. Face recognition on AR-Face

Face recognition is to identify the face images from different known persons. In our experiment, the AR-Face database [21] was chosen for the evaluation, which consists of a set of frontal-view face images with different facial expressions, illuminations, and occlusions. Following the standard protocols used for evaluating sparse coding based classification methods, *e.g.* [5, 9, 10, 20], we used a subset including 100 individuals and total 2600 face images from the database. Each face image was first cropped to 165×120 and then randomly projected onto a 540-dimensional feature vector. Then the subset was randomly divided into two disjoint subsets, one including 20 persons for training and the rest for test.

The parameters of the proposed method were set as follows: the dictionary size $M = 500$, the number of iterations $N = 400$, the sparsity level $T = 18$, the resampling weight $\beta = 0.01$, and the parameters S and R for discriminative atom

selection were set to 4 and 2 respectively. The recognition results of all compared methods on the AR-Face dataset are listed in Table I, where the variances (if available) are reported over 20 runs.

TABLE I
FACE RECOGNITION RESULTS (%) ON AR-FACE.

Method	Accuracy	Method	Accuracy	Method	Accuracy
Joint [14]	88.24	K-SVD [8]	86.5	MCDL [5]	95.21 ± 1.20
DLSI [19]	89.80	D-KSVD [11]	88.80	DeTree	21.83 ± 1.08
FDDL [12]	92.00	LC-KSVD [10]	93.70	SC-DeTree	58.67 ± 0.97
L0DL [17]	94.40	EasyDL [20]	94.40	SCDF	$\mathbf{96.17 \pm 0.70}$

B. Scene classification on Scene-15

Scene classification is to assign the class label to each image based on the scene environments it contains. In our evaluation, the Scene-15 dataset [22] with a wide range of outdoor and indoor scenes was chosen. There are 15 scene categories and total 4485 scene images on the Scene-15 dataset. The images in the dataset are around in the resolution of 250×300, with 210 to 410 images per class. We randomly selected 100 samples in each category for training, and the rest for test. The 3000-dimensional SIFT-based spatial pyramid features [22] extracted from the original images were used as the input.

The parameters of the proposed method were set as follows: the dictionary size $M = 450$, the number of iterations $N = 350$, the sparsity level $T = 8$, the resampling weight $\beta = 0.01$, and the atom selection parameters $S = 22$ and $R = 2$. The classification accuracies of all compared methods on the Scene-15 dataset are shown in Table II, where the variance (if available) is reported over 20 runs.

TABLE II
SCENE CLASSIFICATION RESULTS (%) ON SCENE-15.

Method	Accuracy	Method	Accuracy	Method	Accuracy
Joint [14]	88.20	KSVD [8]	86.70	MCDL [5]	97.35 ± 0.31
DLSI [19]	92.46	D-KSVD [11]	89.10	DeTree	70.45 ± 1.02
FDDL [12]	98.35	LC-KSVD [10]	92.90	SC-DeTree	81.80 ± 1.45
L0DL [17]	93.1	EasyDL [20]	98.46	SCDF	$\mathbf{98.58 \pm 0.20}$

C. Result analysis and discussion

The above results have demonstrated the effectiveness of the proposed method, where the proposed method outperformed all the other compared methods on the datasets. Particularly, since the Joint, D-KSVD and LC-KSVD methods combine the linear prediction penalty with sparse coding, the improvement of the proposed method over these three methods does show the benefits of introducing the nonlinear classification instead of the linear one into sparse coding. It is worth noting that our method yielded better results than two recent sparse coding methods, *i.e.* EasyDL and MCDL. These two methods also use the idea of ensemble learning, where a number of linear classifiers are integrated to handle the nonlinear properties of data. The improvement of the proposed method over these two methods actually indicates that integrating a decision forest into sparse coding can better handle nonlinearities of data than the ensemble of simple linear classifiers.

The comparison with the baseline methods also supports the motivation of our method. It can be seen that, when handling data with high correlation, the simple decision tree classifier does not work well, *e.g.* only 21.83% accuracy gained on the AR-Face dataset. Combining the decision tree with the sparse coding can noticeably boost the performance, which indicates the benefits of using sparse representation for decision tree.

There are two important parameters in our experiments: the sparsity level T which is about the dimension of the space that the data lies at and the number of iteration N which determines the number of decision trees used for the ensemble. In order to evaluate the effects of these two parameters to the performance of our method, we conducted the experiments for parameter analysis, where T or N is sequentially increased while freezing other parameters to observe the change of classification accuracy. The results are shown in Fig. 2, from which we can see that our method shows stability to the setting of T and N in a reasonable range. It is observed that when T is too small, the performance is low. This is because small T cause large reconstructive error. When T reaches a moderate value, the performance becomes optimal. As T increases, the performance drops a bit due to the fact that when T goes beyond the true dimension of data, the learned decision forest become overfitting. It can be also observed that, with the increase of N, the performance is improved accordingly. When N is sufficiently large, the performance saturates and becomes stable. This clearly demonstrates the benefits of using the ensemble strategy in our method.

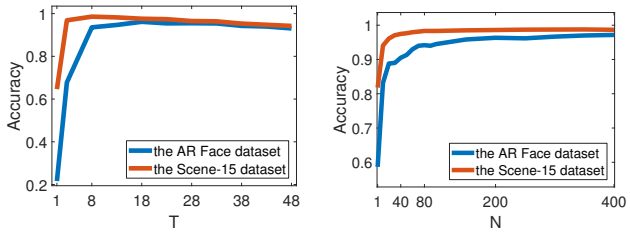

Fig. 2. The influence of parameters to the performance of the proposed method, including (a) the sparsity level and (b) the number of decision trees.

IV. CONCLUSIONS

In this paper, we proposed a supervised sparse coding approach by combining sparse coding and decision tree construction by alternatively conducting one module according to the feedback from the other. With an ensemble strategy, the proposed approach generates a series of dictionaries for sparse coding and multiple decision trees as a forest for classification. We evaluated the performance of the proposed approach with face recognition and scene classification. The power of the proposed approach has been demonstrated by its improvement over the state-of-the-art ones as well as the baselines. In future, we would like to investigate more ensemble techniques for sparse coding.

REFERENCES

[1] X. Cai, F. Nie, and H. Huang, "Exact top-k feature selection via l2, 0-norm constraint." in *International Joint Conference on Artificial Intelligence*, vol. 13, 2013, pp. 1240–1246.

[2] Y. Chen, G. Li, and Y. Gu, "Active orthogonal matching pursuit for sparse subspace clustering," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 164–168, 2018.

[3] K. N. Ramamurthy, J. J. Thiagarajan, P. Sattigeri, and A. Spanias, "Ensemble sparse models for image analysis," *arXiv preprint arXiv:1302.6957*, 2013.

[4] S. Wilson and C. K. Mohan, "Coherent and noncoherent dictionaries for action recognition," *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 698–702, 2017.

[5] Y. Quan, Y. Xu, Y. Sun, and Y. Huang, "Supervised dictionary learning with multiple classifier integration," *Pattern Recognition*, vol. 55, pp. 247–260, 2016.

[6] Y. Zhang, Z. Jiang, and L. S. Davis, "Learning structured low-rank representations for image classification," in *Computer Vision and Pattern Recognition*. IEEE, 2013, pp. 676–683.

[7] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[8] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[9] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 1697–1704.

[10] Z. Jiang, Z. Lin, and L. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[11] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2691–2698.

[12] M. Yang, D. Zhang, and X. Feng, "Fisher discrimination dictionary learning for sparse representation," in *International Conference on Computer Vision*. IEEE, 2011, pp. 543–550.

[13] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[14] D.-S. Pham and S. Venkatesh, "Joint learning and dictionary construction for pattern recognition," in *Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Asilomar Conference on Signals, Systems and Computers*. IEEE, 1993, pp. 40–44.

[17] C. Bao and H. Ji, "Dictionary learning for sparse coding: Algorithms and convergence analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1356–1369, 2016.

[18] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[19] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3501–3508.

[20] Y. Quan, Y. Xu, Y. Sun, Y. Huang, and H. Ji, "Sparse coding for classification via discrimination ensemble," in *Computer Vision and Pattern Recognition*, 2016, pp. 5839–5847.

[21] A. M. Martinez, "The AR face database," *Computer Vision Center, Technical Report*, vol. 24, 1998.

[22] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2006, pp. 2169–2178.