Randomness and Sparsity Induced Codebook Learning with Application to Cancer Image Classification

Quannan Li[1,2], Cong Yao[2,3], Liwei Wang[2,4], Zhuowen Tu[1,2]
[1]Lab of Neuro Imaging, University of California, Los Angeles
[2]Microsoft Research Asia
[3]Huazhong University of Science and Technology
[4]The Chinese University of Hong Kong

Abstract

Codebook learning is one of the central research topics in computer vision and machine learning. In this paper, we propose a new codebook learning algorithm, Randomized Forest Sparse Coding (RFSC), by harvesting the following three concepts: (1) ensemble learning, (2) divide-and-conquer, and (3) sparse coding. Given a set of training data, a randomized tree can be used to perform data partition (divide-and-conquer); after a tree is built, a number of bases are learned from the data within each leaf node for a sparse representation (subspace learning via sparse coding); multiple trees with diversities are trained (ensemble), and the collection of bases of these trees constitute the codebook. These three concepts in our codebook learning algorithm have the same target but with different emphasis: subspace learning via sparse coding makes a compact representation, and reduces the information loss; the divide-and-conquer process efficiently obtains the local data clusters; an ensemble of diverse trees provides additional robustness. We have conducted classification experiments on cancer images as well as a variety of natural image datasets and the experiment results demonstrate the efficiency and effectiveness of the proposed method.

1. Introduction

A large number of applications in machine learning, medical image classification, and computer vision deals with the fundamental representation problem where the data are high-dimensional and live in complex manifolds. With their intrinsic and mathematical properties gradually unfolded, research in three general directions has led to significant progress on classification, recognition, and compression: (1) ensemble learning, (2) divide-and-conquer, and (3) sparse coding. More specifically, four concepts have emerged as being essential to the three directions: (1) voting, (2) randomizing, (3) partitioning, and (4) sparsity.

Ensemble learning approaches such as bagging [3], boosting [13], and random forests [4] have shown to be among the best choices for classifiers [7, 6]. The superior robustness of these ensemble methods comes from the voting/averaging of multiple independent/complementary experts (weak learners). Certain randomness in the data and feature selection stage leads to additional robustness, as shown in the random forests [4] where multiple trees are learned from multiple randomly drawn subsets with the splitting criterion being locally optimal on some random features. In Extremely Randomized Trees [16] and Random Projection Trees [9], the full data sets are used since the randomization in both feature/basis and threshold selection already provide sufficient diversities.

As real data are of high dimension and they typically do not live in a well-regularized space, assuming a Gaussian type distribution leads to limited representational power [28]. When it is hard to fit a global model to the data, divided sub-problems are often easier to deal with. This is a divide-and-conquer strategy [2]. In machine learning, decision tree [25] is a standard approach where training data are recursively partitioned into subsets. The random forests method also has this step in training the individual tree classifier. In addition to tree node splitting, other logic rules such as And/Or can be used as well [8]. The random projection tree [9] also has recursive data partition based on randomly generated bases.

More recently, sparse representations such as compressed sensing [5] and LASSO [27] have gained a great deal of popularity. One message emerging from sparse representation is that high-dimensional data within intrinsic lower dimension can be well represented by sparse samples of high dimension. The robustness of the sparse representation often assumes a subspace of certain regularity, e.g. well-aligned data [32].

In this paper, we tackle the problem of codebook learn-

ing for high dimensional visual data. Inspired by the above observations, we propose a randomized forest sparse coding (RFSC) method. Given a large set of visual data, we train an ensemble of random splitting/projection trees (when we are not sure about the form of the whole data population, it is desirable to perform random partition with certain local optimality); for each leaf node in the tree, we learn a set of bases to best represent the data with sparse coefficients. The overall codebook is a collection of all the bases from all the tree leaves. RFSC carries the ideas of voting, randomizing, partitioning, and sparse coding in a natural way. It's applicable to applications such as natural image classification, and modern cancer diagnosis.

Modern cancer diagnosis largely benefits from high resolution histopathology images, which provide distinctive and reliable cues for discriminating abnormal tissues from normal ones. Therefore, automatic recognition and analysis of cancer in histopathology images are very important assistant means for doctors. In this paper, we verify our algorithm on a collection of colon cancer images. As shown in the experiment section, promising results are obtained.

2. Related Works

As we have discussed, our approach is inspired by the literature in ensemble learning [3, 13, 4], divide-and conquer approaches [2, 25, 8], and sparse representation [5, 27, 32, 21]. Two types of work are particularly related to our approach: tree based splitting/projection methods, e.g., Extremely Randomized Trees [16] and Random Projection Trees [9], and sparse coding based codebook learning techniques [33, 17, 15].

Extremely Randomized Tree (ERT) [16] is a variant of random forest. ERTs randomize both the feature selection and the quantization threshold searching process, making the trees less correlated. When used for visual codebook learning (ERC-Forest) in [22], the generated trees are not treated as an ensemble of decision trees, instead, they are referred to as an ensemble of hierarchical spatial partitioners. The samples (image patches) in each leaf node are assumed to form a small cluster in the feature space. The leaves in the forest are uniquely indexed and serve as the codes for the codebook. When a query sample reaches a leaf node, the index of that leaf is assigned to the query sample. A histogram is formed by accumulating the indices of the leaf nodes, which serves as a Bag of Words (BOA) representation. Similar to ERC-Forest, [26] introduces a semantic texton forest using ERT to perform image classification and segmentation.

Random Projection Tree [9] is a variant of k-d tree. The k-d tree splits the data set along one coordinate at the median and recursively builds the tree. Though widely used for spatial partitioning, it suffers from the curse of dimensionality problem. Based on the realization that, high dimension data often lies on low-dimensional manifold, RPT splits the samples into two roughly balanced sets according to a randomly generated direction. This randomly generated direction approximates the principal component direction, and can adapt to the low dimensional manifold. The RPT naturally leads to tree-based vector quantization and an ensemble of RPTrees can be used as a codebook.

We use Extremely Randomized Trees/Random Projection Trees to partition the samples. But instead of splitting the samples till we cannot split any more, we stop early according to certain criterion and find some bases that can best reconstruct all the samples in that node. These bases serve as codes of the codebook.

There are already some methods using sparse coding for codebook learning. In [33], the authors generalize vector quantization to sparse coding, and construct the histogram using multi-scale spatial max pooling. Each patch can be assigned to several (sparse) codes, and thus the reconstruction error can be reduced. Also, this method extends the Spatial Pyramid Matching method [17] to a linear SPM kernel. In [15], Laplace sparse coding preserves the consistency in the sparse representation and alleviates the problem in [33] that similar patches may be assigned to different codes. In [31], a locality-constrained linear coding scheme is proposed that utilizes the locality constraints to project descriptors to their local-coordinate system. This scheme can preserve the property of local smooth sparsity. Compared with these methods, the advantages of RFSC is obvious. One advantage is the efficiency. Utilizing techniques such as ERT and RPT, the sparse coding is performed only in subspaces and the computational burden is greatly reduced.

The second advantage is the potential promotion of the discriminative ability. The label information can easily be used into the tree splitting process (ERT) and the codebook created could have more discriminative power.

3. Randomized Forest Sparse Coding

3.1. Problem Formulation

Suppose we are given a set of training data $S = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{x}_i \in \mathbb{R}^D$ (in a supervised setting, each $\mathbf{x}_i$ is also associate with a label $y_i \in \mathcal{Y} = \{0, ..., K\}$ and thus $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$), our goal is to learn a codebook (set of basis) $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^m$ and $\mathbf{b}_i \in \mathbb{R}^D$ such that

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{w}} \sum_{i=1}^n \left\| \mathbf{x}_i - \sum_{j=1}^m w_{ij}\mathbf{b}_j \right\|_2^2 \\ &s.t.\ \ \forall i, \sum_j |w_{ij}| \leq \tau \end{aligned} \tag{1}$$

The first term in Eqn. (1) minimizes the reconstruction error and the second term gives the sparsity constraints on the reconstruction coefficients. Eqn. (1) actually includes two coupled optimization problems: (1) given $\mathbf{w}$, find the opti-

mal codebook $\mathbf{B}$; (2) given a codebook $\mathbf{B}$, find the best reconstruction coefficients $\mathbf{w}$. A similar formulation appears in [33].

After an optimal basis set $\mathbf{B}^*$ is found, for a new sample $\mathbf{x}$, we can compute its reconstruction coefficients $\mathbf{w}$ via:

$$\begin{aligned} &\min_{\mathbf{w}} \left\| \mathbf{x} - \sum_{j=1}^{m} w_j \mathbf{b}_j \right\|_2^2 \\ &s.t. \quad \sum_j |w_j| \leq \tau \end{aligned} \tag{2}$$

The vector $\mathbf{w}$ can be used to characterize the sample $\mathbf{x}$. In codebook learning, each $\mathbf{b}_j$ serves as a code, and the reconstruction coefficients with respect to the codes are pooled to form a histogram.

In Eqn. (1), the norm of $\mathbf{b}_j$ can be arbitrarily large, making w_{ij} arbitrarily small. Further constraints should be made on $\mathbf{b}_j$. In our paper, we make a reasonable constraint that all the basis in the codebook should be from the training set S, i.e., $\mathbf{B} \subset S$. With this constraint, Eqn. (1) can be transformed into

$$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j=1}^{n} w_{ij} v_j \mathbf{x}_j \right\|_2^2 \tag{3}$$

$$\begin{aligned} &s.t. \quad \sum_j v_j \leq m, v_j \in \{0,1\} \\ &\forall i, \ \sum_j |w_{ij}| \leq \tau \end{aligned} \tag{4}$$

Here, v_j serves as an indicator value $\in \{0,1\}$ and $\mathbf{B} = \{\mathbf{x}_j : \mathbf{x}_j \in S, \ v_j = 1\}$. Eqn. (3) is seemingly more complex than Eqn. (1) with the introduction of $\mathbf{v}$. However, it can be solved more efficiently since the search space for the basis is greatly reduced.

Learning a codebook of size greater than e.g. $5,000$ from tens of thousands of samples is computationally demanding. However, recent research reveals that data of real-world complexity often live in complex manifolds. As motivated before, we could perform a divide-and-conquer strategy to partition the data space into local subspaces. Within a subspace, it is then much more efficient to learn bases for a sparse representation.

3.2. Randomized Forest Data Partition

In this section, we take the Extremely Randomized Tree (ERT) [16] and Random Projection Trees (RPT) as examples to illustrate the data projection process. Both ERT and RPT partition the samples recursively in a top-down manner. ERT adopts the label information and uses normalized Shannon entropy as the criterion to select features. RPT is unsupervised and it does not need any label information; it splits the data via a hyperplane normalized to each individual randomly generated projection bases.

3.2.1 Discriminative Partition via Extremely Randomized Tree

Given a labeled sample set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, ERT proceeds by randomly selecting a subset of features from the feature pool $\{f_i, 1 \leq i \leq D\}$. For each selected feature f_i, a threshold θ_i is sampled according to a uniform distribution (in [22], a Gaussian distribution adapted to the feature values in that dimension). Based on the features selected and thresholds sampled, boolean tests $\{T_i : \mathbf{x}(i) < \theta_i\}$ can be used to split the set S. If T_i = true, $\mathbf{x}$ goes to the left branch S_1, else, $\mathbf{x}$ goes to the right branch S_2.

To select the best boolean test for splitting, the normalized Shannon entropy was used:

$$Score(S, T_i) = \frac{2 \cdot I_{\mathcal{Y},T_i}(S)}{H_{\mathcal{Y}}(S) + H_{T_i}(S)} \tag{5}$$

where, $I_{\mathcal{Y},T_i}(S) = H_{\mathcal{Y}}(S) - \sum_{p=1}^{2} \frac{n_p}{n} H_{\mathcal{Y}}(S_p)$. $I_{\mathcal{Y},T_i}(S)$ is the information gain, a non-negative scalar denoting the uncertainty reduced by the test T_i. $H_{\mathcal{Y}}(S) = -\sum_{y \in \mathcal{Y}} \frac{n_y}{n} \log_2(\frac{n_y}{n})$ denoting the entropy of class distribution of the original set S. $H_{T_i}(S) = -\sum_{p=1}^{2} \frac{n_p}{n} \log_2(\frac{n_p}{n})$ denotes the entropy for the test T_i that splits the data into two branches. The T_i with the largest $Score(S, T_i)$ is selected.

The use of $H_{T_i}(S)$ as a normalization term in Eqn. (5) was first introduced in [25] to resolve the bias problem: the criterion $I_{\mathcal{Y},T}(S)$ will be biased towards the attributes leading to more branches. In codebook learning, since we are using binary splitting, this bias problem is not a concern. In fact, the use of $H_{T_i}(S)$ as a normalization term will favor uneven splitting, making the forest more unbalanced.

In our randomized forest sparse coding scheme (RFSC), it is desirable to have balanced trees, so we use a slightly modified form of Eqn. (5):

$$Score(S, T_i) = \frac{2 \cdot I_{\mathcal{Y},T_i}(S)}{H_{\mathcal{Y}}(S) + 1 - H_{T_i}(S)} \tag{6}$$

Since $H_{T_i}(S)$ is a concave function and it achieves the maximum value 1 when the numbers of samples in S_1 and S_2 are the same, this criterion can make the trees more balanced.

3.2.2 Unsupervised Splitting via Random Projection Tree (RPT)

At each node, RPT chooses a random unit projection $\mathbf{b} \in \mathbb{R}^D$, and splits the samples into two roughly equal-sized sets. The random projection and thresholding also serve as a type of boolean test. We use the splitting criterion as

$$T := \mathbf{x}^T \mathbf{b} \leq \left(\text{median}(\mathbf{z}^T \mathbf{b} : \mathbf{z} \in S) + \delta \right).$$

Here δ is a random perturbation that adapts to the structure of S. Splitting around the median value makes the splitting balanced while the perturbation δ introduces certain randomness [9].

Since RPTs can automatically adapt to the low dimensional manifold of the dataset S, the samples in the leaf

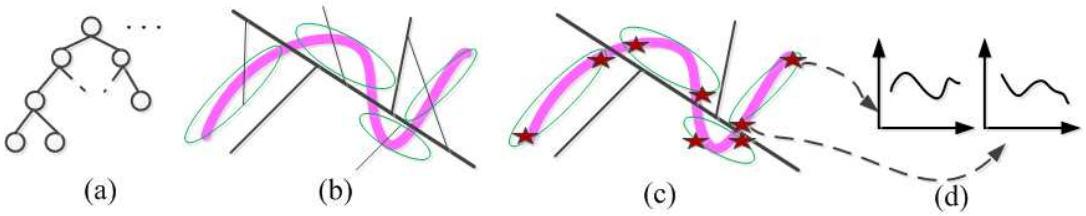

Figure 1. Illustration of the idea of RFSC using Random Projection Tree (best viewed in color). (a) The forest consists of ensemble of random projection trees; (b) The spatial partition of the dataset by one tree (A copy from [12]). A cell stands for a leaf node. The width of the separation line indicates the level of the tree. (c) For RFSC, it does not build the tree to fine level. At certain level when local manifold structures are found, bases (indicated by the red stars) are learned for the local structure in each cell. (d) For the samples in each cell, their reconstruction coefficients with respect to the bases are different.

nodes observe local subspaces. The local structures of all the leaf nodes thus collectively comprise the global structure of the data set S (Fig. 1 (b)).

3.2.3 Basis Pursuit at the Leaf Nodes

Both ERT and RPT build the trees to the fine scale and use the leaf nodes as the codes. Instead of building the trees of very deep level, RFSC stops at some relatively higher level (e.g., when the number of samples is less than M). At such nodes, the local manifold structure is assumed to be relatively simple and regularized. RFSC seeks a set of bases to sparsely represent the subspaces at those nodes. This process can be illustrated using Random Projection Tree in Fig. 1 in which a visualization is displayed and RPT tends to split the data along the principal component direction (Fig. 1 (b)). For RFSC, when the local structure is relatively regularized, it seeks some bases (the red stars) to sparsely represent the local subspace. Different from RPT or ERT that use the mean of the local subspace or a single index to represent the cell, the information conveyed via the reconstruction coefficients with respect to each basis (Fig. 1 (d)) is richer and more informative. Note that the bases in different clusters could be spatially close to each other. As an illustration, see the two bases on the bottom right in Fig. 1(c). From this point of view, the number of basis and the redundancy would increase. However, multiple graphs [29] could help to smooth the boundaries of the overall data representation, and thus, lead to enhanced overall performance. Also, according to Theorem 1 in the justification part, the total number of bases in all the leaf nodes is bounded. Since at each node when the splitting process stops, there are generally $80 \sim 200$ samples (depending on the codebook size) and $3 \sim 10$ bases, the computational overhead of subspace learning is not significant compared with directly pursuing basis from the entire sample set.

3.3. Optimization Scheme

The constraint that $v_j \in \{0, 1\}$ makes Eqn. (3) a hard problem. In this subsection, we present two schemes to solve this optimization problem. The first one is to relax v_j to a real value and use coordinate descent algorithm to optimize on $\mathbf{w}$ and $\mathbf{v}$ iteratively. The second one is a greedy pursuit approach that selects the bases one by one.

Convex Relaxation The first optimization scheme is to relax the values of v_j to real numbers and use ℓ^1 constraint $\sum_j |v_j| \leq m$ instead of ℓ^0 like constraint in Eqn. (3). Putting this constraint as a regularization term, we can transform this problem into an equivalent form:

$$\frac{1}{2}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \sum_{j=1}^{n} w_{ij} v_j \mathbf{x}_j\right\|_2^2 + \lambda_1 \sum_{i,j}|w_{ij}| + \lambda_2 \sum_j |v_j| \tag{7}$$

Here, $v_j \in \mathbb{R}$. λ_1 and λ_2 are regularization parameters that make the trade-offs between the residue and the norms of the weight vectors.

There are two sets of variables $\mathbf{w}$ and $\mathbf{v}$ in Eqn. (7). To optimize Eqn. (7), we adopt an EM-like algorithm that iterates by fixing one set of variables and optimize on the other set using coordinate descent algorithm [14].

By fixing $\mathbf{w}$, we can get the updated value $\tilde{v}_j$ of v_j as:

$$\tilde{v}_j = \frac{\text{shrink}\left(\sum_i w_{ij}\beta_{ij}, \lambda_2\right)}{\sum_i w_{ij}^2 \mathbf{x}_j^T \mathbf{x}_j} \tag{8}$$

Here, $\beta_{ij} = \mathbf{x}_j^T\left(\mathbf{x}_i - \sum_{k\neq j} w_{ik} v_k \mathbf{x}_k\right)$. $\text{shrink}(f, \lambda)$ is an operator to shrink the value of f toward 0:

$$\text{shrink}(f, \lambda) = \begin{cases} f - \lambda & \text{if} \quad f > \lambda \\ 0 & \text{if} -\lambda \leq \lambda \leq \lambda \\ f + \lambda & \text{if} \quad f < -\lambda \end{cases} \tag{9}$$

By fixing $\mathbf{v}$, we get the updated value $\tilde{w}_{ij}$ of w_{ij} as:

$$\tilde{w}_{ij} = \frac{\text{shrink}\left(v_j\beta_{ij}, \lambda_1\right)}{v_j^2 \mathbf{x}_j^T \mathbf{x}_j} \tag{10}$$

For the sake of efficiency, in practice, instead of using the pathwise coordinate descent algorithm [14] that sweeps all the variables sequentially,we adopt an adaptive and greedy sweeping version [19] that sweeps the variable that decreases the objective function most at each iteration. For w_{ij} and its updated value $\tilde{w}_{ij}$, the decrease of the objective function is

$$\Delta R_{w_{ij}} = \frac{1}{2}v_j^2 \|\mathbf{x}_j\|_2^2 (w_{ij} - \tilde{w}_{ij}) \left(w_{ij} + \tilde{w}_{ij} - \frac{2v_j\beta_{ij}}{v_j^2\|\mathbf{x}_j\|_2^2} \right) + \lambda_1(|w_{ij}| - |\tilde{w}_{ij}|) \quad (11)$$

For v_j and its updated value $\tilde{v}_j$, the decrease of the objective function is

$$\Delta R_{v_j} = \sum_{i=1}^n \frac{1}{2}w_{ij}^2 \|\mathbf{x}_j\|_2^2 (v_j - \tilde{v}_j) \left(v_j + \tilde{v}_j - \frac{2w_{ij}\beta_{ij}}{w_{ij}^2\|\mathbf{x}_j\|_2^2} \right) + n\lambda_2(|v_j| - |\tilde{v}_j|) \quad (12)$$

At each iteration, the updating rule is to select the variable that leads to the largest decrease in the objective function. Thus, when fixing $\mathbf{v}$ to optimize on $\mathbf{w}$,

$$w_{ij}^* = \arg\max_{w_{ij}} \Delta R(w_{ij}) \quad (13)$$

and when fixing $\mathbf{w}$ to optimize on $\mathbf{v}$

$$v_j^* = \arg\max_{v_j} \Delta R(v_j) \quad (14)$$

In our experiment, the adaptive and greedy sweeping proves efficient for practical use. After the optimization process converges, we rank the samples according to their v_j. The first m samples with the largest non-zero v_j are selected as the basis.

Greedy Pursuit Approach Starting from an empty basis collection, the greedy pursuit approach selects the basis one by one. Suppose some l samples $\mathbf{B}_l = \{\mathbf{x}_{s_i}, 0 \le i \le l, 1 \le s_i \le n\}$ have been selected from the n samples, i.e., $v_{s_i} = 1$. To select the $(l+1)$th basis, we optimize the following function:

$$s_{l+1} = \min_{k \notin \{s_i\}} \frac{1}{2} \sum_{i=1}^n \left\| \mathbf{x}_i - \sum_{j \in \{s_i\}} w_{ij}\mathbf{x}_j - w_{ik}\mathbf{x}_k \right\|_2^2 + \lambda_1 \sum_{i=1}^n \sum_{j \in \{s_i\}} |w_{ij}| + \lambda_1 \sum_{i=1}^n |w_{ik}| \quad (15)$$

According to Eqn. (15), the sample that reconstructs all the n patches together with the first l selected basis is selected as the $(l+1)$th basis.

The greedy approach finds suboptimal solution to Eqn. (3). But it's more efficient than the convex relaxation approach, and in practice, we find that its performance is comparable with the convex relaxed solution. Thus in some of our experiments, we only use this greedy approach.

3.4. Theoretical justification

In this section, we give some theoretical justification to our approach. Our institution is to show that the three steps in randomized forest sparse coding: (1) ensemble of trees, (2) randomized projection tree, and (3) sparse coding leads to the same complexity level in the number of basis as to the original data.

Given $S = \{\mathbf{x}_i, i = 1..n\}$ with $\mathbf{x}_i \in \mathbb{R}^D$, assume that $\mathbf{x}_i$ lives in the intrinsic lower dimension $d \ll D$. It can be seen that the number of basis needed to reconstruct S is bounded. Following the definition of Assouad dimension [1] [9]:

Definition: For any point $\mathbf{x} \in \mathbb{R}^D$ and $r > 0$, let $B(\mathbf{x}, r) = \{||\mathbf{x} - \mathbf{z}|| \le r\}$ denote the closed ball of radius r centered at $\mathbf{x}$. The Assouad dimension of $S \in \mathbb{R}^D$ is the smallest integer d such that for any ball $B(\mathbf{x}, r) \in \mathbb{R}^D$, the set $B(\mathbf{x}, r) \cap S$ can be covered by 2^d balls of radius $r/2$.

Theorem 1 *The number of basis needed to reconstruct S by Randomized Forest Sparse Coding (RFSC) is $O(2^{d \log d})$.*

Proof:

Fixing radius r, suppose we want to create a codebook such that each basis function covers $r/2$, a size of $O(2^d)$ codebook is required to cover the entire dataset S, according to the definition of Assouad dimension.

The main result in [9] shows that $O(d \log d)$ levels of a random projection/partition tree would reach cells with radius $r/2$. Therefore, the number of cells is $O(2^{d \log d})$. Suppose there are k trees in the forest, and in each leaf node, l basis are found, then the number of the basis becomes $O(kl2^{d \log d})$. As k and l are generally small and can be kept constant, the bound still reduces to:

$$O(2^{d \log d}).$$

Although RFSC slightly increases the size of the codebook compared to $O(2^d)$, since d is generally small ($d \ll D$), this is reasonably bounded.

4. Experiments

To evaluate the effectiveness of the proposed codebook learning algorithm, we conducted extensive classification experiments on a collection of cancer images and a variety of natural image datasets: Graz-02 image set, the INRIA Horse dataset, and the PASCAL 2005 image set.

As the baselines, we obtained the source code for ERC-Forest from the authors of [22] and implemented the RPTs according to [9]. In our experiments, the feature vectors are used without any normalization, which is sometimes done in subspace learning and sparse coding (we found that performing normalization does not affect the overall performance in the experiments reported here). For each leaf node, 5 bases are learned. For the Graz-02 image set, $\lambda_1 = 2$ and $\lambda_2 = 6$, while for the INRIA Horse dataset and

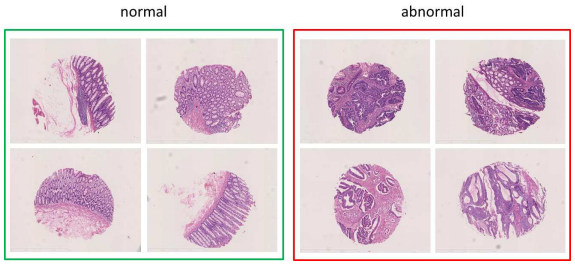

Figure 2. Cancer image examples. The images in the green box are normal samples. i.e. there are no cancerous cells. The images in the red box are abnormal samples, i.e. there are cancerous cells.

the PASCAL 2005 image set, $\lambda_1 = 15$ and $\lambda_2 = 6$. To solve the subspace learning problem via sparse coding defined in Eqn. (3), 10 iterations between $\mathbf{w}$ and $\mathbf{v}$ are enough to find a good sparse solution.

In the following, we use RFSC to denote subspace learning via sparse coding under Extremely Randomized Trees; RPT-SC denotes subspace learning on Random Projection Trees. For RFSC and RPT-SC, the postfix "-Cvx" refers to using the convex relaxation version and "-Gdy" regards to using the greedy basis pursuit version. For the classification task of Cancer Images, the performance is measured using the Area under the curve of the ROC curve, while for natural image classification, the performance is measured using the classification accuracy at the Equal Error Rate and the reported accuracies are the averages of 10 rounds of execution. As can be seen from the experiments, our method achieves comparable or superior performance with the alternatives.

4.1. Experiments on Cancer Images

Dataset: We used a histopathology image data set with 60 colon images (30 cancer images and 30 non-cancer images). Some example images from this dataset are shown in Fig. 2. The images are in the resolution of 1280×1024. All the images are labeled as cancer or non-cancer by two pathologists independently. If disagreement happens for a certain image, these two pathologists together with a third senior pathologist will carefully examine and discuss until final agreement. **Experimental Setup:** Before feature extraction, the original images are down-sampled with a factor of 2. Since no obvious spatial regularities are observed from the images (Fig. 2), we didn't densely compute local features and construct Bag-of-Features (BOF) vectors. Instead, we randomly sample $N = 200$ local patches (32×32) for each image. Each patch is represented by Lab color histogram, Local Binary Pattern [23], and SIFT [20]. For the proposed method, each patch is encoded by the proposed coding schemes RFSC or RPT-SC; for the baseline, we use the raw feature. Random Forests [4] is adopted as the strong classifier for its simplicity and high performance. The overall classification score of an image is the mean of the scores of all the patches. Half of the images in the dataset are chosen randomly for training and the rest for testing. We run the experiments 5 times for each method and report the averaged performance. For RFSC and RPT-SC, the convex relaxation versions are used. We compare the Area under the ROC curve between RPT-SC, RFSC, ERC-Forest, RPT, and raw feature. The Area Under Curve (AUC) for the methods are RPT-SC 0.98, RFSC 0.987, RPT 0.927, ERC-Forest 0.95, and raw feature 0.967 respectively; our method performs better than the alternatives.

4.2. Experiments on Natural Images

The reconstruction coefficients are pooled in the natural image classification task. To pool the reconstruction coefficients, unless otherwise stated, max-pooling is used as in [33]. The pooled reconstruction coefficients of the trees are concatenated to form a histogram leaving the voting process till the classification step. SVM is used as the classification model, and the linear kernel is used. To understand better the behavior of the competing codebook learning algorithms, in all the following image classification experiments, we do not include the adaptive saliency map process. This makes the image classification performance of ERC-Forest slightly worse than that reported in [22]. However, this performance degeneration is understandable and in accordance with the case illustration in [22]. Focusing on the core codebook learning part helps to better validate the underlying benefits of our method against the competing algorithms.

GRAZ-02 dataset [24] GRAZ-02 image set consists of three object categories and one counter-category: Bicycles (365 images), Cars (420 images), Person (311 images) and None (380 images). For each category, the categorization task is to distinguish the object category from the counter-category, None. Similar to [22], we also pick the two hardest cases: Cars vs. None and Bikes vs. None. Patches are sampled from the images and 768-D wavelet feature vectors are used as the descriptors.

To make a direct comparison with [22] and [24], we conduct the experiment according to the setting in [22]: the first 300 images of each category are used and 5 trees are trained. We use the greedy version of RFSC and vary the codebook size from 5000 to 9000. From Table 1 and Table 2, we observe that, RFSC-Gdy performs better than ERC-Forest and the method in [24]. Although RPT-SC-Gdy does not outperform ERC-Forest, it still performs better than RPT on both of the two cases. RFSC-Gdy outperforms RPT-SC, indicating the promotion of the discriminative ability by introducing the label information in the divide-and-conquer process. Even without the adaptive saliency map process, the accuracy (83.9%) of RFSC-Gdy on the case of Bikes vs. None approaches that reported in [22] (84.4%). Note that we only use 5 bases in each leaf node to represent a 768 dimensional

Table 1. Comparison of the accuracy on the case of Cars vs. None in the GRAZ-02 images [24].

size of codebook	5000	6000	7000	8000	9000
[24]	70.5%				
ERC-Forest	71.3%	73.5%	74.5%	74.7%	74.8%
RPT	66.5%	66.6%	65.3%	67.7%	66.9%
RFSC-Gdy	73.4%	74.3%	**75.7%**	74.9%	74.3%
RPT-SC-Gdy	68%	69.8%	69%	69.5%	68.2%

Table 2. Comparison of the accuracy for Bikes vs. None in the GRAZ-02 images [24].

size of codebook	5000	6000	7000	8000	9000
[24]	77.8%				
ERC-Forest	78.8%	78%	78.5%	78.5%	78.5%
RPT	73.3%	74.3%	74.1%	75.1%	74.4%
RFSC-Gdy	80.7%	**83.9%**	80.8%	81.3%	80%
RPT-SC-Gdy	76.5%	76.8%	76.1%	76.7%	76%

Table 3. Comparison of the accuracy using all the images for Cars vs. None in the GRAZ-02 images [24].

size of codebook	5000	6000	7000	8000	9000
ERC-Forest	67.2%	67%	68.6%	68.8%	71.3%
Leaf-Kmeans	68.2%	70.9%	73%	72.6%	73.2%
RFSC-Cvx-1tree	72.6%	72.2%	71.4%	75%	75%
RFSC-Cvx	75%	75%	73.7%	73.1%	75.2%
RFSC-Gdy	74.3%	75.5%	74.5%	74.8%	**75.5%**

Table 4. Comparison of the accuracy using all the images for Bikes vs. None in the GRAZ-02 images [24].

size of codebook	5000	6000	7000	8000	9000
ERC-Forest	77.8%	78.3%	78.3%	79.1%	78.8%
Leaf-Kmeans	75.1%	74.4%	79.7%	78.7%	79.5%
RFSC-Cvx-1tree	77.8%	78.2%	78.6%	79.5%	79.5%
RFSC-Cvx	80%	82.2%	**82.6%**	81.4%	81.8%
RFSC-Gdy	81.5%	80.3%	81.5%	80.8%	80.9%

feature space; the large improvement in classification accuracy not only proves the relative regularity in the local subspaces, but also supports the formulation in Eqn. (1) and the effectiveness of the sparse representation.

We also conduct the experiments using all the images instead of the first 300 images. Average-pooling is adopted here and the results are reported in Table 3 and Table 4. The performance of the two optimization schemes is similar: for the case of Cars vs. None, RFSC-Gdy achieves the best accuracy 75.5% and for Bikes vs. None, RFSC-Cvx achieves the best accuracy 82.6%. RFSC-Cvx-1tree refers to using one randomized tree instead of the forest, an ensemble of trees. It performs worse than RFSC. This justifies the benefit of using ensembles and is in accordance with the spirit in ensemble learning: the randomized partition process provides sufficient diversities among codes of the forest, and the concatenated codebook produces better and more robust results via voting.

We do not compare RFSC and RPT-SC with directly performing dictionary learning on the image classification task since solving Eqn. (1) directly when $m = 5,000$ or $9,000$ is time consuming. However, benefiting from the divide-and-conquer process, it takes less than 1 hour for RFSC and RPT-SC to build a forest with 5 trees and $9,000$ codes. This improvement in efficiency stems from seeking a small amount of bases from hundreds of patches instead of seeking thousands of bases from tens of thousands of training patches. Other efficient algorithms such as [18] can be used to solve Eqn. (1), but the conclusion of the improvement in efficiency induced by the divide-and-conquer process still holds. RFSC and RPT-SC are also very efficient at the testing stage. It takes about 0.5 second to process an image and pooling the reconstruction coefficients. As a comparison, it would take around 30 seconds for K-Means to assign patches to the codes for an image when the feature vector is of dimension 768 and the codebook size K is $5,000$.

Table 5. Comparison of the accuracy on the INRIA Horse dataset [11].

method	ERC-Forest	RPT	RFSC-Gdy	RPT-SC-Gdy
Accuracy	79.2%	75.7%	**85.9%**	80.4%
size of Codebook	9000	7000	5000	8000

Table 6. Comparison of the accuracy on PASCAL 2005 image set [10].

method	motobikes	cars	bikes	person
ERC-Forest	96%	95%	89%	90.9%
RFSC-Gdy	**96.4%**	**95.3%**	**90.6%**	**91.4%**

INRIA Horse Dataset [11] INRIA horse dataset contains 170 horses taken from the Internet with different sizes and poses. The training/splitting ratio of this dataset and the size of the codebook were not reported in [22], so we randomly selected 85 horse images for training and varied the size of codebook from $5,000$ to $9,000$. The SIFT descriptor is used to describe the patches and we used the dense SIFT implementation in [30]. The greedy pursuit approach was used. In Table 5, we report the best accuracy of each method and the size of the codebook at which the best accuracy is achieved. From Table 5 we observe that, RFSC-Gdy performs better than ERC-Forest and RPT-SC-Gdy performs better than RPT. The performance of ERC-Forest has the potential to be improved if the size of the codebook increases. We did not carry out the experiment, since even without estimating the saliency map and with small codebook, RFSC-Gdy has already achieved better result (85.9%) than that reported in [22] (85.3%).

PASCAL 2005 image set [10] We also compare our method with ERC-Forest on PASCAL 2005 image set. The results are shown in Table 6. RFSC-Gdy achieves better results on all of the 4 categories than ERC-Forest.

5. Conclusion

In this paper, we have introduced a codebook learning method called randomized forest sparse coding that integrates three concepts: ensemble, divide-and-conquer and sparse coding. Justifications for the effectiveness and efficiency of our method are also provided. The proposed scheme is applied to both the Cancer Image Classification and natural image classification and observes significant improvement in performance. Future work includes the following issues: first, we will investigate other means of feature representation. Second, the spatial pyramid representation of RFSC will be investigated. Third, since there exists certain redundancy in RFSC, we will improve the coding schemes to reduce the redundancy.

Acknowledgment: This work is supported by Office of Naval Research Award N000140910099 and NSF CAREER award IIS-0844566.

References

[1] P. Assouad. Plongements lipschitziens dans rn. *Bull. Soc. Math. France*, (4):429–448, 1983. 5

[2] J. L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980. 1, 2

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 1, 2

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 1, 2, 6

[5] E. Candes and T. Tao. Near-optimal signal recovery from random projections: universal encoding strategies. *IEEE Trans. Inform. Theory*, 52(2):5406–5425, 2005. 1, 2

[6] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML*, pages 96–103, 2008. 1

[7] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, pages 161–168, 2006. 1

[8] Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In *NIPS*, 2007. 1, 2

[9] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, pages 537–546, 2008. 1, 2, 3, 5

[10] M. Everingham and A. Zisserman. The 2005 pascal visual object classes challenge. In *MLCW*, pages 117–176, 2005. 7

[11] V. Ferrari, F. Jurie, and C. Schmid. Accurate Object Detection with Deformable Shape Models Learnt from Images. In *CVPR*, 2007. 7

[12] Y. Freund, S. Dasgupta, M. Kabra, and N. Verma. Learning the structure of manifolds using random projections. In *NIPS*, volume 20, 2007. 4

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Sys. Sci.*, 55(1), 1997. 1, 2

[14] J. Friedma, T. Hastie, H. Hofling, and R. Tibshirani. Pathwise Coordinate Optimization. *The Annals of Applied Stat.*, 2007. 4, 5

[15] S. Gao, I. W. hung Tsang, L. tien Chia, and P. Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, 2010. 2

[16] P. G. June, D. Ernst, and L. Wehenkel. Extremely Randomized Trees. In *Machine Learning*, volume 36, 2003. 1, 2, 3

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2

[18] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007. 7

[19] Y. Li and S. Osher. Coordinate descent optimization for ℓ^1 minimization with application to compressed sensing; a greedy algorithm. In *CAM Report*, 2009. 5

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 6

[21] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. In *IEEE Trans. on PAMI*, To appear. 2

[22] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Trans. on PAMI*, 30(9):1632–1646, 2008. 2, 3, 5, 6, 7

[23] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002. 6

[24] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic Object Recognition with Boosting. *IEEE Trans. on PAMI*, 28(3):416–431, 2006. 6, 7

[25] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1, 1986. 1, 2, 3

[26] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 2

[27] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 56(1):267–288, 1996. 1, 2

[28] M. Turk. Eigenface for recognition. *Journal of Cognitive Neuroscience*, 1991. 1

[29] T. Uno, M. Sugiyama, and K. Tsuda. Efficient construction of neighborhood graphs by the multiple sorting method. In *CoRR*, 2009. 4

[30] A. Vedaldi and B. Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *ACM Multimedia*, pages 1469–1472, 2010. 7

[31] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2

[32] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. on PAMI*, 31(2), 2009. 1, 2

[33] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *CVPR*, 2009. 2, 3, 6