Kernel Sparse Representation for Image Classification and Face Recognition

Conference Paper · September 2010

DOI: 10.1007/978-3-642-15561-1_1 · Source: DBLP

Ivor W Tsang
University of Technology Sydney
222 PUBLICATIONS **9,568** CITATIONS

SEE PROFILE

Liang-Tien Chia
Nanyang Technological University
155 PUBLICATIONS **3,297** CITATIONS

SEE PROFILE

Kernel Sparse Representation for Image Classification and Face Recognition

Shenghua Gao, Ivor Wai-Hung Tsang, Liang-Tien Chia

School of Computer Engineering, Nanyang Technological Univertiy, Singapore
{gaos0004,IvorTsang,asltchia}@ntu.edu.sg

Abstract. Recent research has shown the effectiveness of using sparse coding(Sc) to solve many computer vision problems. Motivated by the fact that kernel trick can capture the nonlinear similarity of features, which may reduce the feature quantization error and boost the sparse coding performance, we propose Kernel Sparse Representation(KSR). KSR is essentially the sparse coding technique in a high dimensional feature space mapped by implicit mapping function. We apply KSR to both image classification and face recognition. By incorporating KSR into Spatial Pyramid Matching(SPM), we propose KSRSPM for image classification. KSRSPM can further reduce the information loss in feature quantization step compared with Spatial Pyramid Matching using Sparse Coding(ScSPM). KSRSPM can be both regarded as the generalization of Efficient Match Kernel(EMK) and an extension of ScSPM. Compared with sparse coding, KSR can learn more discriminative sparse codes for face recognition. Extensive experimental results show that KSR outperforms sparse coding and EMK, and achieves state-of-the-art performance for image classification and face recognition on publicly available datasets.

1 Introduction

Sparse coding technique is attracting more and more researchers' attention in computer vision due to its state-of-the-art performance in many applications, such as image annotation [25], image restoration [20], image classification [28] *etc.* It aims at selecting the least possible basis from the large basis pool to linearly recover the given signal under a small reconstruction error constraint. Therefore, sparse coding can be easily applied to feature quantization in Bag-of-Word(BoW) model based image representation. Moreover, under the assumption that the face images to be tested can be reconstructed by the images from the same categories, sparse coding can also be used in face recognition [26].

BoW model [23] is widely used in computer vision [27, 21] due to its concise representation and robustness to scale and rotation variance. Generally, it contains three modules: (i) Region selection and representation; (ii) Codebook generation and feature quantization; (iii) Frequency histogram based image representation. In these three modules, codebook generation and feature quantization are the most important portions for image presentation. The codebook is a collection of basic patterns used to reconstruct the local features. Each basic pattern is known as a visual word. Usually k-means is adopted to generate the codebook, and each local feature is quantized to its nearest visual word. However, such hard assignment method may cause severe information loss [3, 6], especially for those features located at the boundary of several visual words. To minimize such errors, soft assignment [21, 6] was introduced by assigning each feature to more than one visual words. However, the way of choosing parameters, including the weight assigned to the visual word and the number of visual words to be assigned, is not trivial to be determined.

Recently, Yang *et al.* [28] proposed the method of using sparse coding in the codebook generation and feature quantization module. Sparse coding can learn better codebook that further minimizes the quantization error than k-means. Meanwhile, the weights assigned to each visual word are learnt concurrently. By applying sparse coding to Spatial Pyramid Matching [13] (referred to as: ScSPM), their method achieves state-of-the-art performance in image classification.

Another application of sparse coding is face recognition. Face recognition is a classic problem in computer vision, and has a great potential in many real world application. It generally contains two stages. (i): Feature extraction; and (ii): Classifier construction and label prediction. Usually Nearest Neighbor (NN) [5] and Nearest Subspace(NS) [11] are used. However, NN predicts the label of the image to be tested by only using its nearest neighbor in the training data, therefore it can easily be affected by noise. NS approximates the test image by using all the images belonging to the same category, and assigns the image to the category which minimizes the reconstruction error. But NS may not work well for the case where classes are highly correlated to each other[26]. To overcome these problems, Wright *et al.* proposed a sparse coding based face recognition framework [26], which can automatically selects the images in the training set to approximate the test image. Their method is robust to occlusion, illumination and noise and achieves excellent performance.

Existing work based on sparse coding only seeks the sparse representation of the given signal in original signal space. Recall that kernel trick [22] maps the non-linear separable features into high dimensional feature space, in which features of the same type are easier grouped together and linear separable. In this case we may find the sparse representation for the signals more easily, and the reconstruction error may be reduced as well. Motivated by this, we propose Kernel Sparse Representation(KSR), which is the sparse coding in the mapped high dimensional feature space.

The contributions of this paper can be summarized as follows: (i): We propose the idea of kernel sparse representation, which is sparse coding in a high dimensional feature space. Experiments show that KSR greatly reduces the feature reconstruction error. (2): We propose KSRSPM for image classification. KSRSPM can be regarded as a generalized EMK, which can evaluate the similarity between local features accurately. Compared with EMK, our KSRSPM is more robust by using quantized feature other than the approximated high dimensional feature. (3): We extend KSR to face recognition. KSR can achieve more discriminative sparse codes compared with sparse coding, which can boost the performance for face recognition.

The rest of this paper is organized as follows: In Section 2, we describe the details of KSR, including its objective function and its implementation. By incorporating KSR into SPM framework, we propose KSRSPM in Section 3. We also emphasize the relationship between our KSRSPM and EMK in details. Image classification performance on several public available datasets are also reported at the end of this section. In Section 4, we use KSR for face recognition. Results comparisons between sparse coding and KSR on Extended Yale B Face Dataset are listed in this section. Finally, we conclude our work in Section 5.

2 Kernel Sparse Representation and Implementation

2.1 Kernel Sparse Representation

For general sparse coding, it aims at finding the sparse representation under the given basis $U(U \in \mathbb{R}^{d\times k})$, while minimizing the reconstruction error. It equals to solving the following objective.

$$\begin{aligned} &\underset{U,v}{min}\|x - Uv\|^2 + \lambda\|v\|_1 \\ &subject\ \ to: \quad \|u_m\|^2 \leq 1 \end{aligned} \tag{1}$$

where $U = [u_1, u_2, ..., u_k]$. The first term of Equation (1) is the reconstruction error, and the second term is used to control the sparsity of the sparse codes v. Empirically larger λ corresponds to sparser solution.

Suppose there exists a feature mapping function ϕ: $\mathcal{R}^d \to \mathcal{R}^K$, $(d < K)$. It maps the feature and basis to the high dimensional feature space: $x \to \phi(x)$, $U = [u_1, u_2, ..., u_k] \to \mathcal{U} = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]$. We substitute the mapped features and basis to the formulation of sparse coding, and arrive at kernel sparse representation(KSR):

$$\underset{U,v}{min} \quad \|\phi(x) - \mathcal{U}v\|^2 + \lambda\|v\|_1 \tag{2}$$

where $\mathcal{U} = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]$. In our work, we use Gaussian kernel due to its excellent performance in many work [22, 2]: $\kappa(x_1, x_2) = exp(-\gamma\|x_1 - x_2\|^2)$. Note that $\phi(u_i)^T\phi(u_i) = \kappa(u_i, u_i) = exp(-\gamma\|u_i - u_i\|^2) = 1$, so we can remove the constraint on u_i. Kernel sparse representation seeks the sparse representation for a mapped feature under the mapped basis in the high dimensional space.

2.2 Implementation

The objective of Equation (2) is not convex. Following the work of [28, 14], we optimize the sparse codes v and the codebook $\mathcal{U}$ alternatively.

Learning The Sparse Codes in New Feature Space When the codebook U is fixed, the objective in Equation (2) can be rewritten as:

$$\begin{aligned} &\underset{v}{min} \quad \|\phi(x) - \mathcal{U}v\|^2 + \lambda\|v\|_1 \\ =&\kappa(x,x) + v^T K_{UU} v - 2v^T K_U(x) + \lambda\|v\|_1 \\ =&L(v) + \lambda\|v\|_1 \end{aligned} \tag{3}$$

where $L(v) = 1 + v^T K_{UU} v - 2v^T K_U(x)$, K_{UU} is a $k * k$ matrix with $\{K_{UU}\}_{ij} = \kappa(u_i, u_j)$, and $K_U(x)$ is a $k * 1$ vector with $\{K_U(x)\}_i = \kappa(u_i, x)$. The objective is the same as that of sparse coding except for the definition of K_{UU} and $K_U(x)$. So we can easily extend the Feature-Sign Search Algorithm[14] to solve the sparse codes. As for the computational cost, they are the same except for the difference in calculating kernel matrix.

Learning Codebook When v is fixed, we learn the codebook U. Due to the large amount of features, it is hard to use all the feature to learn the codebook. Following the work [28, 2], we random sample some features to learn the codebook U, then use the learnt U to sparsely encode all the features. Suppose we randomly sample N features, then we rewrite the objective as follows (m, s, t are used to index the columns number of the codebook.):

$$\begin{aligned} f(U) &= \frac{1}{N}\sum_{i=1}^{N}[\|\phi(x_i) - \mathcal{U}v_i\|^2 + \lambda\|v_i\|_1] \\ &= \frac{1}{N}\sum_{i=1}^{N}[1 + \sum_{s=1}^{k}\sum_{t=1}^{k} v_{i,s}v_{i,t}\kappa(u_s, u_t) - 2\sum_{s=1}^{k} v_{i,s}\kappa(u_s, x_i) + \lambda\|v_i\|_1] \end{aligned} \tag{4}$$

Since U is in the kernel ($\kappa(u_i, .)$), it is very challenging to adopt the commonly used methods, for example, Stochastic Gradient Descent method [2] to find the optimal codebook. Instead we optimize each column of U alternatively. The derivative of $f(U)$ with respect to u_m is (u_m is the column to be updated):

$$\frac{\partial f}{\partial u_m} = \frac{-4\gamma}{N}\sum_{i=1}^{N}[\sum_{t=1}^{k} v_{i,m}v_{i,t}\kappa(u_m, u_t)(u_m - u_t) - v_{i,m}\kappa(u_m, x_i)(u_m - x_i)] \tag{5}$$

To find the optimal u_m, we set $\frac{\partial f}{\partial u_m} = 0$. However, it is not easy to solve the equation due to the terms with respect to $\kappa(u_m, .)$. As a compromise, we use the approximate solution to replace the exact solution. Similar to fixed point algorithm [12], in the n^{th} u_m updating iteration, we use the result of u_m in the $(n-1)^{th}$ updating iteration to compute the part in the kernel function. Denote the u_m in the n^{th} updating process as $u_{m,n}$, then the equation with respect to $u_{m,n}$ becomes:

$$\begin{aligned} \frac{\partial f}{\partial u_{m,n}} &\cong \frac{-4\gamma}{N}\sum_{i=1}^{N}[\sum_{t=1}^{k} v_{i,m}v_{i,t}\kappa(u_{m,n-1}, u_t)(u_{m,n} - u_t) - v_{i,m}\kappa(u_{m,n-1}, x_i)(u_{m,n} - x_i)] \\ &= 0 \end{aligned}$$

When all the remaining columns are fixed, it becomes a linear equation of $u_{m,n}$ and can be solved easily. Following the work [2], the codebook is initialized as the results of k-means.

3 Application I: Kernel Sparse Representation for Image Classification

In this Section, we apply kernel sparse representation in SPM framework, and propose the KSRSPM. On the one hand, KSRSPM is an extension of ScSPM [28] by replacing sparse coding with KSR. On the other hand, KSRSPM can be regarded as the generalization of Efficient Match Kernel(EMK) [2].

3.1 Sparse Coding for Codebook Generation

k-means clustering is usually used to generate the codebook in BoW model. In k-means, the whole local feature space $X = [x_1, x_2, \ldots, x_N]$ (where $x_i \in \mathbb{R}^{d\times 1}$) is split into k clusterings $S = [S_1, S_2, \ldots, S_k]$. Denote the corresponding clustering centers as $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{d\times k}$. In hard assignment, each feature is only assigned to its nearest cluster center, and the weight the feature contributing to that center is 1. The objective of k-means can be formulated as the following optimization problem:

$$\begin{aligned} &\underset{U,S}{min} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - u_i\|^2 = \underset{U,V}{min} \sum_{i=1}^{N} \|x_i - Uv_i\|^2 \\ &subject \;\; to: \quad Card(v_i) = 1, |v_i| = 1, v_i \succeq 0, \forall i. \end{aligned} \tag{6}$$

Here V is a clustering indices, $V = [v_1, v_2, \ldots, v_N]$ (where $v_i \in \mathbb{R}^{k\times 1}$). Each column of V indicates which visual word the local feature should be assigned to. To reduce the information loss in feature quantization, the constraint on v_m is relaxed. Meanwhile, to avoid each feature being assigned to too many clusters, the sparse constraint is imposed on v_m. Then, we arrive at the optimization problem of sparse coding:

$$\begin{aligned} &\underset{U,V}{min} \sum_{i=1}^{N} \|x_i - Uv_i\|^2 + \lambda \|v_i\|_1 \\ &subject \;\; to: \quad |u_j| \le 1, \forall j = 1, \ldots, k. \end{aligned} \tag{7}$$

3.2 Maximum Feature Pooling and Spatial Pyramid Matching Based Image Representation

Following the work of [28, 4], we use maximum pooling method to represent the images. Maximum pooling uses the largest responses to each basic pattern to represent the region. More specifically, suppose one image region has D local features, and the codebook size is k. After maximum pooling, each image will be represented by a k dimensional vector y, and the l^{th} entry is the largest response to the l^{th} basis vector of all the sparse codes in the selected region(v_D is the sparse codes of the D^{th} feature in this local region, and v_{Dl} is the l^{th} entry of v_D):

$$y_l = max\{|v_{1l}|, |v_{2l}|, \ldots, |v_{Dl}|\} \tag{8}$$

SPM technique is also used to preserve the spatial information. The whole image is divided into increasing finer regions, and maximum pooling is used in each subregion.

3.3 KSRSPM – An Generalization of Efficient Matching Kernel

Besides interpreted as an extension of ScSPM [28], KSRSPM can also be interpreted as a generalization of Efficient Matching Kernel (EMK) [2]. Let $X = [x_1, x_2, \ldots, x_p]$ be a set of local features in one image, and $V(x) = [v_1(x), v_2(x), \ldots, v_p(x)]$ are the corresponding clustering index vector in Equation (6). In BoW model, each image is presented by a normalized histogram $\bar{v}(X) = \frac{1}{|X|}\sum_{x\in X} v(x)$, which characterizes its visual word distribution. By using linear classifier, the resulting kernel function is:

$$K_B(X,Y) = \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y} v(x)^T v(y) = \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y} \delta(x,y) \tag{9}$$

where

$$\delta(x,y) = \begin{cases} 1, & v(x) = v(y) \\ 0, & otherwise \end{cases} \tag{10}$$

$\delta(x,y)$ is positive definite kernel, which is used to measure the similarity between two local features. However, such hard assignment based local feature similarity measuring method increases the information loss and reduces classification accuracy. Thus a continuous kernel is introduced to more accurately measure the similarity between local feature x and y:

$$K_S(X,Y) = \frac{1}{|X||Y|}\sum_{x\in X}\sum_{y\in Y} k(x,y) \tag{11}$$

Here $k(x,y)$ is positive definite kernel, which is referred to as local kernel. This is related to the normalized sum match kernel [19, 9].

Due to the large amount of local features, directly using local kernel is both storage and computation prohibitive for image classification. To decrease the computation cost, Efficient Match Kernel(EMK) is introduced. Under the definition of finite dimensional kernel function [2], $k(x,y) = \phi(x)^T\phi(y)$, we can approximate $\phi(x)$ by using low dimensional features $\bar{v}_x$ in the space spanned by k basis vectors $H = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]$:

$$\min_{H, v_x} \quad \|\phi(x) - Hv_x\|^2 \tag{12}$$

In this way, each image can be represented by $\bar{v}(X)_{new} = \frac{1}{|X|} H \sum_{x \in X} v_x$ beforehand. As a consequence, the computation speed can be accelerated.

EMK maps the local feature to high dimensional feature space to evaluate the similarity between local features more accurately, and uses the approximated feature Hv_x to construct the linear classifier for image classification. It can be summarized as two stages: (i): $x \xrightarrow{\phi} \phi(x)$: Map the feature to new feature space; (ii): $\phi(x) \xrightarrow{H} \bar{v}_x$: Reconstruct $\phi(x)$ by using the basis H.

Note that directly using original feature for image classification may cause overfitting [3]. To avoid this, and following the BoW model, we use v_x for image classification. We hope each $\phi(x)$ is only assigned several clusterings, so we add the sparse constraint in the objective of EMK:

$$\min_{H, v_x} \quad \|\phi(x) - Hv_x\|^2 + \lambda \|v_x\|_1 \tag{13}$$

This is the same as the objective of our kernel sparse representation. So EMK can be regarded as the special case of our KSRSPM at $\lambda = 0$. Compared with EMK, our KSRSPM uses the quantized feature indices for image classification, so it is more robust to the noise. What's more, by using maximum pooling, the robustness to intra-class and noise of our KSRSPM can be further strengthened.

3.4 Experiments

Parameters Setting SIFT [16] is widely used in image recognition due to its excellent performance. For a fair comparison and to be consistent with previous work [28, 13, 2], we use the SIFT features under the same feature extraction setting. Specifically, we use dense grid sampling strategy and fix the step size and patch size to 8 and 16 respectively. We also resize the maximum side(width/length) of each image to 300 pixels[1]. After obtaining the SIFT, we use $\ell 2$-norm to normalize the feature length to 1. For the codebook size, we set $k = 1024$ in k-means, and randomly select $(5.0 \sim 8.0) * 10^4$ features to generate codebook for each data set. Following the work [28], we set $\lambda = 0.30$ for all the datasets. As for the parameter γ in the Gaussian kernel, we set γ to $\frac{1}{64}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}$ on Scene 15, UIUC-Sports, Caltech 256 and Corel 10 respectively. For SPM, we use top 3 layers and the weight for each layer is the same. We use one-vs-all linear SVM due to its advantage in speed [28] and excellent performance in maximum feature pooling based image classification. All the results for each dataset are based on six independent experiments, and the training images are selected randomly.

Scene 15 Dataset Scene 15 [13] dataset is usually used for scene classification. It contains 4485 images, which are divided into 15 categories. Each category contains about 200 to 400 images. The image content is diverse, containing *suburb,*

[1] For UIUC-Sport dataset, we resize the maximum side to 400 due to the high resolution of original image.

coast, forest, highway, inside city, mountain, open country, street, tall building, office, bedroom, industrial, kitchen, living room and *store*. For fair comparison, we follow the same experimental setting [28, 13]: randomly select 100 images each category as training data and use the remaining images as test data. The results are listed in Table 1.

Table 1. Performance Comparison on Scene 15 Dataset(%)

Method	Average Classification Rate
KSPM[13]	81.40±0.50
EMK[2]	77.89±0.85
ScSPM[28]	80.28±0.93
KSRSPM	**83.68±0.61**

Caltech 256 Caltech 256[2] is a very challenging dataset in both image content and dataset scale. First of all, compared with Caltech 101, the objects in Caltech 256 contains larger intra-class variance, and the object locations are no longer in the center of the image. Second, Caltech 256 contains 29780 images, which are divided into 256 categories. More categories will inevitably increase the inter-class similarity, and increase the performance degradation. We evaluate the method under four different settings: selecting 15, 30, 45, 60 per category as training data respectively, and use the rest as test data. The results are listed in Table 2.

Table 2. Performance Comparison on Caltech 256 dataset(%) (KC: Kernel codebook;)

Trn No.	KSPM[8]	KC[6]	EMK[2]	ScSPM[28]	**KSRSPM**
15	NA	NA	23.2±0.6	27.73±0.51	**29.77±0.14**
30	34.10	27.17±0.46	30.5±0.4	34.02±0.35	**35.67±0.10**
45	NA	NA	34.4±0.4	37.46±0.55	**38.61±0.19**
60	NA	NA	37.6±0.5	40.14±0.91	**40.30±0.22**

UIUC-Sport Dataset UIUC-Sport [15] contains images collected from 8 kind of different sports: *badminton, bocce,croquet, polo, rock climbing, rowing, sailing* and *snow boarding*. There are 1792 images in all, and the number of images ranges from 137 to 250 per category. Following the work of Wu *et al.* [27], we randomly select 70 images from each category as training data, and randomly

[2] `www.vision.caltech.edu/Image_Datasets/Caltech256/`

select another 60 images from each category as test data. The results are listed in Table 3.

Table 3. Performance Comparison on UIUC-Sport Dataset(%)

Method	Average Classification Rate
HIK+ocSVM[27]	83.54±1.13
EMK[2]	74.56±1.32
ScSPM[28]	82.74±1.46
KSRSPM	**84.92±0.78**

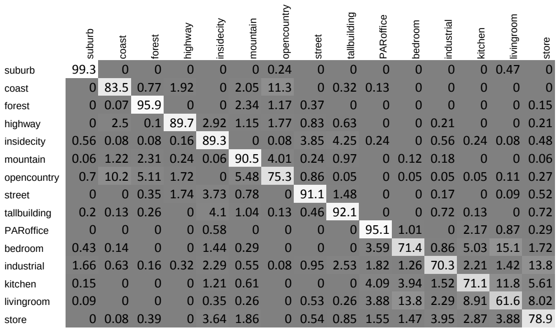
Corel10 Dataset Corel10 [18] contains 10 categories: *skiing, beach, buildings, tigers, owls, elephants, flowers, horses, mountains* and *food*. Each category contains 100 images. Following the work of Lu *et al.* [18], we randomly select 50 images as training data and use the rest as test data. The results are listed in Table 4.

Table 4. Performance Comparison on Corel10 Dataset(%) (SMK:Spatial Markov Model)

Method	Average Classification Rate
SMK [17]	77.9
EMK [2]	79.90±1.73
ScSPM [28]	86.2±1.01
KSRSPM	**89.43±1.27**

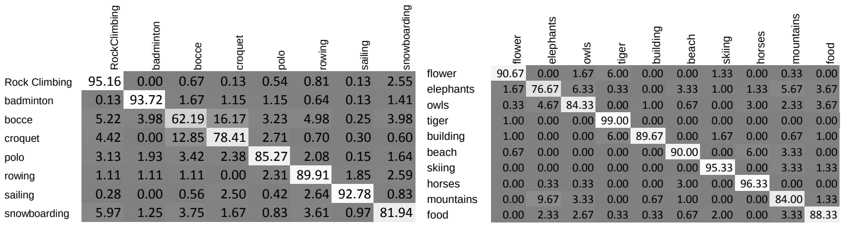
Results Analysis From Table 1-4, we can see that on Scene, UIUC-Sports, Corel10, KSRSPM outperforms EMK around $(5.7 \sim 10.4)\%$, and outperforms ScSPM around $(2.2 \sim 3.4)\%$. For Caltech 256, due to too many classes, the improvements are not very substantial, but still higher than EMK and ScSPM. We also list the confusion matrices of Scene, UIUC-Sports and Corel10 datasets in Figure 1 and Figure 2. The entry located in i^{th} row, j^{th} column in confusion matrix represents the percentage of class i being misclassified to class j. From the confusion matrices, we can see that some classes are easily be misclassified to some others.

Feature Quantization Error Define Average Quantization Error (AverQE) as: $AverQE = \frac{1}{N}\sum_{i=1}^{N} \|\phi(x_i) - \mathcal{U}v_i\|_F^2$. It can be used to evaluate the information



	suburb	coast	forest	highway	insidecity	mountain	opencountry	street	tallbuilding	PARoffice	bedroom	industrial	kitchen	livingroom	store
suburb	99.3	0	0	0	0	0	0.24	0	0	0	0	0	0	0.47	0
coast	0	83.5	0.77	1.92	0	2.05	11.3	0	0.32	0.13	0	0	0	0	0
forest	0	0.07	95.9	0	0	2.34	1.17	0.37	0	0	0	0	0	0	0.15
highway	0	2.5	0.1	89.7	2.92	1.15	1.77	0.83	0.63	0	0	0.21	0	0	0.21
insidecity	0.56	0.08	0.08	0.16	89.3	0	0.08	3.85	4.25	0.24	0	0.56	0.24	0.08	0.48
mountain	0.06	1.22	2.31	0.24	0.06	90.5	4.01	0.24	0.97	0	0.12	0.18	0	0	0.06
opencountry	0.7	10.2	5.11	1.72	0	5.48	75.3	0.86	0.05	0	0.05	0.05	0.05	0.11	0.27
street	0	0	0.35	1.74	3.73	0.78	0	91.1	1.48	0	0	0.17	0	0.09	0.52
tallbuilding	0.2	0.13	0.26	0	4.1	1.04	0.13	0.46	92.1	0	0	0.72	0.13	0	0.72
PARoffice	0	0	0	0	0.58	0	0	0	0	95.1	1.01	0	2.17	0.87	0.29
bedroom	0.43	0.14	0	0	1.44	0.29	0	0	0	3.59	71.4	0.86	5.03	15.1	1.72
industrial	1.66	0.63	0.16	0.32	2.29	0.55	0.08	0.95	2.53	1.82	1.26	70.3	2.21	1.42	13.8
kitchen	0.15	0	0	0	1.21	0.61	0	0	0	4.09	3.94	1.52	71.1	11.8	5.61
livingroom	0.09	0	0	0	0.35	0.26	0	0.53	0.26	3.88	13.8	2.29	8.91	61.6	8.02
store	0	0.08	0.39	0	3.64	1.86	0	0.54	0.85	1.55	1.47	3.95	2.87	3.88	78.9

Fig. 1. Confusion Matrix on Scene 15 dataset(%).



	RockClimbing	badminton	bocce	croquet	polo	rowing	sailing	snowboarding
Rock Climbing	95.16	0.00	0.67	0.13	0.54	0.81	0.13	2.55
badminton	0.13	93.72	1.67	1.15	1.15	0.64	0.13	1.41
bocce	5.22	3.98	62.19	16.17	3.23	4.98	0.25	3.98
croquet	4.42	0.00	12.85	78.41	2.71	0.70	0.30	0.60
polo	3.13	1.93	3.42	2.38	85.27	2.08	0.15	1.64
rowing	1.11	1.11	1.11	0.00	2.31	89.91	1.85	2.59
sailing	0.28	0.00	0.56	2.50	0.42	2.64	92.78	0.83
snowboarding	5.97	1.25	3.75	1.67	0.83	3.61	0.97	81.94

	flower	elephants	owls	tiger	building	beach	skiing	horses	mountains	food
flower	90.67	0.00	1.67	6.00	0.00	0.00	1.33	0.00	0.33	0.00
elephants	1.67	76.67	6.33	0.33	0.00	3.33	1.00	1.33	5.67	3.67
owls	0.33	4.67	84.33	0.00	1.00	0.67	0.00	3.00	2.33	3.67
tiger	1.00	0.00	0.00	99.00	0.00	0.00	0.00	0.00	0.00	0.00
building	1.00	0.00	0.00	6.00	89.67	0.00	1.67	0.00	0.67	1.00
beach	0.67	0.00	0.00	0.00	0.00	90.00	0.00	6.00	3.33	0.00
skiing	0.00	0.00	0.00	0.00	0.00	0.00	95.33	0.00	3.33	1.33
horses	0.00	0.33	0.33	0.00	0.00	3.00	0.00	96.33	0.00	0.00
mountains	0.00	9.67	3.33	0.00	0.67	1.00	0.00	0.00	84.00	1.33
food	0.00	2.33	2.67	0.33	0.33	0.67	2.00	0.00	3.33	88.33

Fig. 2. Confusion Matrices on UIUC-Sports and Corel10(%).

loss in the feature quantization process. To retain more information, we hope the feature quantization error can be reduced. We compute the AverQE of our kernel sparse representation (KSR) and Sparse coding (Sc) on all the features used for codebook generation, and list them in Table 5. From results we can see that kernel sparse representation can greatly decrease the feature quantization error. This may be the reason that our KSRSPM outperforms ScSPM. The results also agree with our assumption that sparse coding in high dimensional space can reduce the feature quantization error.

Table 5. Average Feature Quantization Error on Different datasets.

	Scene	Caltech 256	Sport	Corel
Sc	0.8681	0.9164	0.8864	0.9295
KSR	**9.63E-02**	**5.72E-02**	**9.40E-02**	**4.13E-02**

4 Application II: Kernel Sparse Representation for Face Recognition

4.1 Sparse Coding for Face Recognition

For face recognition, "If sufficient training samples are available from each class, it would be possible to represent the test samples as a linear combination of those training samples from the same class [26]".

Suppose there are N classes in all, and the training instances for class i are $A_i = [a_{i,1}, \ldots, a_{i,n_i}] \in \mathbb{R}^{d \times n_i}$, in which each column corresponds to one instance. Let $A = [A_1, \ldots, A_N] \in \mathbb{R}^{d \times \sum_{i=1}^{N} n_i}$ be the training set, and $y \in \mathbb{R}^{d \times 1}$ be the test sample. When noise e exists, the problem for face recognition [26] can be formulated as follows:

$$min \quad \|x_0\|_1 \qquad s.t. \qquad y = Ax^T + e = [A \; I][x^T \; e^T]^T = A_0 x_0 \tag{14}$$

sparse coding based image recognition aims at selecting only a few images from all the training instances to reconstruct the images to be tested. Let $\alpha_i = [\alpha_{i,1}, \ldots, \alpha_{i,n_i}] (1 \leq i \leq N)$ be the coefficients corresponds to A_i in x_0. The reconstruction error by using the instances from class i can be computed as: $r_i(y) = \|y - A_i \alpha_i\|_2$. Then the test image is assigned to the category that minimizes the reconstruction error: $identity(y) = arg \; min_i \; \{r_1(y), \ldots, r_N(y)\}$.

4.2 Kernel Sparse Representation for Face Recognition

Kernel method can make the features belonging to the same category closer to each other [22]. Thus we apply kernel sparse representation in face recognition. Firstly, the $\ell 1$ norm on reconstruction error is replaced by using $\ell 2$ norm(We assume that the noise may not be sparsely reconstructed by using the training samples). By mapping features to a high dimensional space: $y \rightarrow \phi(y)$, $A = [a_{1,1}, \ldots, a_{N,n_N}] \rightarrow \mathcal{A} = [\phi(a_{1,1}), \ldots, \phi(a_{N,n_N})]$, we obtain the objective of kernel sparse representation for face recognition:

$$min \quad \lambda \|x\|_1 + \|\phi(y) - \mathcal{A}x\|_2^2 \tag{15}$$

In which the parameter λ is used to balance the weight between the sparsity and the reconstruction error. Following the work of John Wright *et al.*, the test image is assigned to the category which minimizes the reconstruction error in the high dimensional feature space.

4.3 Evaluation on Extended Yale B Database

We evaluate our method on Extended Yale B Database [7], which contains 38 categories, 2414 frontal-face images. The cropped image size is 192 × 168. Following the work [26], we randomly select a half as training images in each category, and use the rest as test. The following five features are used for evaluation:

RandomFace [26], LaplacianFace [10], EigenFace [24],FisherFace [1] and Downsample [26], and each feature is normalized to unit length by using $\ell 2$ norm. Gaussian kernel is used in our experiments: $\kappa(x_1, x_2) = exp(-\gamma \|x_1 - x_2\|^2)$. For Eigenfaces, Laplacianfaces, Downsample and Fisherfaces, we set $\gamma = 1/d$ where d is the feature dimension. For Randomfaces, $\gamma = 1/32d$.

The Effect of λ We firstly evaluate λ by using 56D Downsample Feature. We list the results based on different λ in Table 6. When $\lambda \neq 0$, as λ decreases, the performance increases, and the proportion of non-zero elements in coefficients increases. But computational time also increases. When $\lambda = 0$, it happens to be the objective of Efficient Match Kernel, but the performance is not good as that in the case of $\lambda \neq 0$. This can show the effectiveness of the sparse term.

Table 6. The Effect of Sparsity Parameter: 56D Downsample Feature (Here sparsity is percentage of non-zeros elements in sparse codes).

λ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	0
sparsity(%)	0.58	0.75	0.88	2.13	4.66	8.35	16.69	-
reconstruction error	0.2399	0.1763	0.1651	0.1113	0.0893	0.0671	0.0462	-
time(sec)	0.0270	0.0280	0.0299	0.0477	0.2445	0.9926	6.2990	-
accuracy(%)	76.92	84.12	85.19	90.32	91.65	93.30	93.47	84.37

Result Comparison Considering both the computational cost and the accuracy in Table 6, we set $\lambda = 10^{-5}$. The experimental results are listed in Table 7. All the results are based on 10 times independent experiments. Experimental results show that kernel sparse representation can outperform sparse coding in face recognition.

Table 7. Performance of Sparse Coding for Face Recognition(%)

	Feature Dimension	30	56	120	504
Eigen	Sc [26]	86.5	91.63	93.95	96.77
	KSR	**89.01**	**94.42**	**97.49**	**99.16**
Laplacian	Sc [26]	87.49	91.72	93.95	96.52
	KSR	**88.86**	**94.24**	**97.11**	**98.12**
Random	Sc [26]	82.6	91.47	95.53	98.09
	KSR	**85.46**	**92.36**	**96.14**	**98.37**
Downsample	Sc [26]	74.57	86.16	92.13	97.1
	KSR	**83.57**	**91.65**	**95.31**	**97.8**
Fisher	Sc [26]	86.91	NA	NA	NA
	KSR	**88.93**	NA	NA	NA

To further illustrate the performance of KSR, we calculate the similarity between the sparse codes of KSR and Sc in three classes(each classes contains 32 images). We list the results in Figure 3, in which the entry in (i, j) is the sparse codes similarity (normalized correlation) between image i and j. We know that a good sparse coding method can make the sparse codes belonging to same class more similar, therefore, the sparse codes similarity should be block-wise. From Figure 3 we can see that our KSR can get more discriminative sparse codes than sparse coding, which facilitates the better performance of the face recognition.

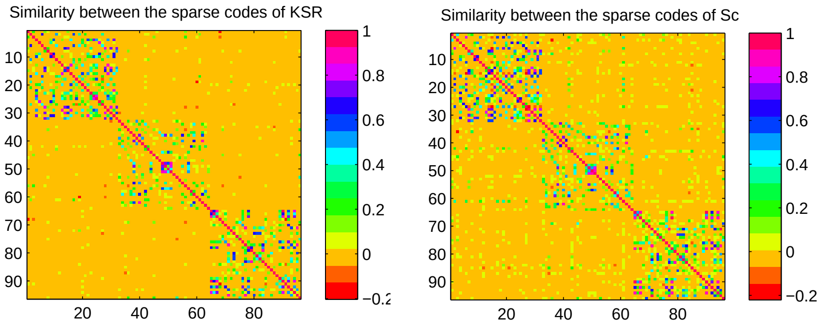


Fig. 3. Similarity between the sparse codes of KSR and Sc.

5 Conclusion

In this paper, we propose a new technique: Kernel Sparse Representation, which is the sparse coding technique in a high dimensional feature space mapped by implicit feature mapping feature. We apply KSR to image classification and face recognition. For image classification, our proposed KSRSPM can both be regarded as an extension of ScSPM and an generalization of EMK. For face recognition, KSR can learn more discriminative sparse codes for face category identification. Experimental results on several publicly available datasets show that our KSR outperforms both ScSPM and EMK, and achieves state-of-the-art performance.

References

1. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. TPAMI 19(7), 711–720 (1997)
2. Bo, L., Sminchisescu, C.: Efficient match kernels between sets of features for visual recognition. In: NIPS (2009)
3. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: CVPR (2008)

4. Boureau, Y., Bach, F., LeCun, Y., Ponce, J.: Learnning mid-level features for recognition (2010)
5. Duda, R.O., Hart, P.E., Stock, D.G.: Pattern Classification. John Wisley & Sons, 2 edn. (2001)
6. van Gemert, J.C., Geusebroek, J.M., Veenman, C.J., Smeulders, A.W.M.: Kernel codebooks for scene categorization. In: ECCV (October 2008)
7. Georghiades, A., Belhumeur, P., Kriegman, D.: From few to many: Illumination cone models for face recognition under variable lighting and pose. TPAMI 23(6), 643–660 (2001)
8. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. In: Technical Report (2007)
9. Haussler, D.: Convolution kernels on discrete structure. In: Technical Report (1999)
10. He, X., Yan, S., Hu, Y., Niyogi, P., Zhang, H.: Face recognition using laplacianfaces. TPAMI 27(3), 328–340 (2005)
11. Ho, J., Yang, M.H., Lim, J., Lee, K.C., Kriegman, D.J.: Clustering appearances of objects under varying illumination conditions. In: CVPR (2003)
12. Hyvärinen, A.: The fixed-point algorithm and maximum likelihood estimation for independent component analysis. Neural Process. Lett. 10(1) (1999)
13. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. pp. 2169–2178 (2006)
14. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. pp. 801–808 (2006)
15. Li, L.J., Fei-Fei, L.: What, where and who? classifying events by scene and object recognition. In: ICCV (2007)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (2004)
17. Lu, Z., Ip, H.H.: Image categorization by learning with context and consistency. In: CVPR (2009)
18. Lu, Z., Ip, H.H.: Image categorization with spatial mismatch kernels. In: CVPR (2009)
19. Lyu, S.: Mercer kernels for object recognition with local features. In: CVPR. pp. 223–229 (2005)
20. Marial, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Non-local sparse models for image restoration. In: ICCV (2009)
21. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
22. Schölkopf, B., Smola, A.J., Müller, K.R.: Kernel principal component analysis. In: International Conference on Artificial Neural Networks. pp. 583–588 (1997)
23. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV. pp. 1470–1477 (2003)
24. Turk, M., Pentland, A.: Eigenfaces for recognition. In: CVPR (1991)
25. Wang, C., Yan, S., Zhang, L., Zhang, H.J.: Multi-label sparse coding for automatic image annotation. In: CVPR (2009)
26. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. TPAMI 31(2), 210–227 (2009)
27. Wu, J., Rehg, J.M.: Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In: ICCV (2003)
28. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)