A Scalable Unsupervised Feature Merging Approach to Efficient Dimensionality Reduction of High-dimensional Visual Data

Lingqiao Liu
CECS, Australian National University
ACT 0200, Canberra, Australia
lingqiao.liu@cecs.anu.edu.au

Lei Wang
School of Computer Science & Software Engineering
University of Wollongong, NSW 2522, Australia
leiw@uow.edu.au

Abstract

To achieve a good trade-off between recognition accuracy and computational efficiency, it is often needed to reduce high-dimensional visual data to medium-dimensional ones. For this task, even applying a simple full-matrix-based linear projection causes significant computation and memory use. When the number of visual data is large, how to efficiently learn such a projection could even become a problem. The recent feature merging approach offers an efficient way to reduce the dimensionality, which only requires a single scan of features to perform reduction. However, existing merging algorithms do not scale well with high-dimensional data, especially in the unsupervised case.

To address this problem, we formulate unsupervised feature merging as a PCA problem imposed with a special structure constraint. By exploiting its connection with k-means, we transform this constrained PCA problem into a feature clustering problem. Moreover, we employ the hashing technique to improve its scalability. These produce a scalable feature merging algorithm for our dimensionality reduction task. In addition, we develop an extension of this method by leveraging the neighborhood structure in the data to further improve dimensionality reduction performance. In further, we explore the incorporation of bipolar merging – a variant of merging function which allows the subtraction operation – into our algorithms.

Through three applications in visual recognition, we demonstrate that our methods can not only achieve good dimensionality reduction performance with little computational cost but also help to create more powerful representation at both image level and local feature level.

1. Introduction

The recent advances in visual recognition introduce many high-dimensional representations, e.g. high-dimensional histograms or pooled coding vectors [2], Fisher vectors [15] and histogram of local binary patterns with a large-sized neighborhood [7]. Although higher-dimensional representations help achieving better recognition performance, it is often desirable to reduce the dimensionality for the sake of saving computational load and memory usage. To achieve a good trade-off between recognition performance and computational efficiency, we often need to reduce the high-dimensional data to medium-dimensional ones, e.g. from 65536 to 1024. However, this requirement makes traditional dimensionality reduction methods, even the simple full-matrix linear projection, inefficient in performing the reduction in terms of computational load and memory usage. For instance, the above example needs one million multiplication operations to project a *single* sample and incurs 500MB memory to save the double-precision projection matrix. As a result, the dimensionality reduction may become unaffordable for the applications requiring real-time performance, e.g. real-time detection and tracking, and in the systems with limited memory, e.g. embedding systems. Moreover, when both the number of feature dimensions and samples are large, how to efficiently learn the reduction function will become a problem.

The recent feature merging approach offers an efficient way to perform the dimensionality reduction. The idea of feature merging is to group features into clusters and use the sum of the features within each cluster as each reduced feature. As shown in the recent literature [6, 18, 13], this simple strategy can achieve good performance for histogram-alike data. Compared with traditional dimensionality reduction, merging is much more efficient in performing the reduction and incurs much lower memory usage. It only needs a scan over all D features to accomplish the reduction and a $1 \times D$ sized table to store the merging indexes.

However, existing feature merging, especially the unsupervised merging algorithms do not scale well with high-dimensional data. Their poor scalability is caused by two factors: (1) the clustering frameworks adopted in these methods incur super-linear computational load growth with

the feature dimensions, e.g. hierarchical clustering [6, 18, 13, 10] and spectral clustering [11]. (2) besides the scalability issue in their clustering frameworks, the existing unsupervised merging algorithms [10, 11] require the calculation of pairwise feature similarity by some measures whose computational cost grows linearly with the number of samples, e.g. point-wise mutual information. Thus these methods become computationally impractical in dealing with a large number of samples, say, one million samples.

Compared with supervised merging, unsupervised merging has much wider applications (e.g. dictionary learning, image retrieval). However, many of those applications involve a large number of samples and thus the scalability issue of existing unsupervised merge methods severely limits their applications.

To overcome this problem, in this paper we propose an unsupervised feature merging algorithm which scales well with both high feature dimensionality and large sample size. In this algorithm, we formulate unsupervised feature merging as a principal component analysis (PCA) problem imposed with a special structure constraint. By exploiting its connection with k-means, we transform this constrained PCA problem into a feature clustering problem. However, directly solving this clustering problem is time-consuming when the sample size is large. To handle this situation, we employ the hashing technique to improve its scalability. The combination of the above two techniques finally gives us a scalable feature merging algorithm. Also, inspired by kernel alignment which is commonly used in supervised learning, we further propose an extension of the above basic method. More specifically, we integrate the neighborhood structure in data to generate 'pseudo-supervised' information and utilize this information with kernel alignment to formulate a new merging function learning problem, which can be efficiently solved by our basic algorithm with a slight modification. In addition, we explore to incorporate a variant of merging function called bipolar merging which allows the subtraction operation in the merging process.

To show the significance of our methods, we introduce three applications – dimensionality reduction of the bag-of-features (BoF) based image representation, learning better compact local binary pattern and dimensionality reduction of high-dimensional local features. Through the comparison with other alternatives in each application, we demonstrate that our methods can not only achieve good dimensionality reduction performance with little computational cost but also help to create more powerful representation at both image level and local feature level.

2. Related Work

Dimensionality reduction is a classic topic in machine learning and has many applications in computer vision. The related literature is just too many to review. Related to efficient dimensionality reduction, the existing methods [5, 1] mainly focus on the time and space complexity of learning the reduction function rather than that of performing the reduction. For example, the popular random projection method [1] almost has zero cost in learning the projection function but still has high storage and computational complexity in performing the reduction. One recent work on reducing the complexity in performing dimensionality reduction is Hashing [16]. It calculates the projection matrix via hash function and results in a algorithm which has both low time and space complexity in performing the reduction.

Feature merging can be categorized into supervised and unsupervised approaches. Examples of supervised merging algorithms include the methods in [19, 18, 6, 13]. The use of feature merging as an efficient dimensionality reduction is well demonstrated in [6], where the merging reduction is employed to accelerate the integral histogram calculation for object detection. Unsupervised merging algorithms have been also developed in [11, 10]. The method in [10] directly extends the supervised merging method in [6] by substituting the class-conditional probability with the probability of a word occurring in each image. This substitution makes the pairwise feature similarity calculation required in its hierarchical clustering framework grow linearly with the sample size. The method in [11] also needs to calculate the same feature similarity to construct the Laplacian matrix for their spectral-clustering-alike method. As discussed in the introduction section, this feature similarity evaluation together with the inefficient clustering framework significantly affect their scalability and limit their potential applications.

3. Our Methods

3.1. The Basic Method

Formulation: Our basic algorithm is inspired by PCA – the most popular unsupervised dimensionality reduction method. In a nutshell, we express the merging operation as a linear projection operator with a special structure constraint and try to incorporate this structure constraint into the PCA formulation.

Recall that PCA solves:

$$\max_{\mathbf{W}} \ \operatorname{trace}\left(\mathbf{W}\mathbf{X}\mathbf{X}^T\mathbf{W}^T\right) \quad s.t. \ \mathbf{W}\mathbf{W}^T = \mathbf{I} \qquad (1)$$

where $\mathbf{X} \in R^{D\times N}$ is the centralized data matrix consisting of D N-dimensional samples. $\mathbf{W} \in R^{d\times D}$ is the projection matrix, where d is the reduced dimensionality. We also denote the i^{th} column and the (j,i) entry of $\mathbf{X}$ as $\mathbf{x}_i$ and $x_{j,i}$ respectively. Similar notations are applied to other matrices.

Let $\mathbf{Y} \in R^{d\times N}$ represent the data obtained by applying the merging operator over $\mathbf{X}$. In conventional feature merging, $y_{j,i} = \sum_{k\in\mathcal{G}_j} x_{k,i}$, where $\mathcal{G}_j$ denotes a set of indexes assigned to the j^{th} group. This merging operation is

also equivalent to multiplying $\mathbf{X}$ with a special structured $\mathbf{W}$: (1) each entry of $\mathbf{W}$ is either '0' or '1'. (2) each column of $\mathbf{W}$ has at most one '1' entry. This implies that $\mathbf{W}\mathbf{W}^T = \mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is a diagonal matrix with its j^{th} diagonal component λ_j equal to the cardinality of the set $\mathcal{G}_j$ (denoted as $|\mathcal{G}_j|$). From (1), we can easily verify that $y_{j,i} = \mathbf{w}_{(j,:)}\mathbf{x}_i = \sum_{\{k|w_{j,k}=1\}} x_{k,i}$. So each row of $\mathbf{W}$, denoted by $\mathbf{w}_{(j,:)}$, corresponds a group and the entries with '1' indicate the features that should be merged into that group. The property (2) effectively requires that each feature can only be assigned to one group. However, if we replace the constraint in Eqn. (1) with the aforementioned constraints, it will result in a mixed-integer problem which is difficult to solve. Fortunately, if we slightly modify the merging function, we could arrive at a formulation which is equivalent to k-means clustering. Specifically, we propose to calculate $y_{j,i} = \frac{1}{\sqrt{|\mathcal{G}_j|}} \sum_{k\in\mathcal{G}_j} x_{k,i}$ instead of simply summing the feature values in one group. Translating this merging function into the structure constraint on $\mathbf{W}$, we obtain:

$$\mathbf{W} \in \Omega = \{\mathbf{W} | w_{i,j} \in \{0, \frac{1}{\sqrt{\lambda_i}}\} \ \forall i, \ \ \mathbf{W}\mathbf{W}^T = \mathbf{I}\} \quad (2)$$
$$\lambda_i = \|\mathbf{w}_{(i,:)}\|_0 = |\mathcal{G}_j|,$$

where $\|\cdot\|_0$ denotes the zero norm (number of nonzero entries). Incorporating this constraint into PCA, we obtain:

$$\max_{\mathbf{W}\in\Omega} \ \text{trace}\left(\mathbf{W}\mathbf{X}\mathbf{X}^T\mathbf{W}^T\right). \quad (3)$$

For the purpose of finding a better solution for k-means, the work in [23, 4] show that if we treat $\mathbf{W}$ as the cluster membership indicator, the following problem is equivalent to the k-means formulation with data matrix $\mathbf{A}$.

$$\max_{\mathbf{W}\in\Omega} \ \text{trace}\left(\mathbf{W}\mathbf{A}^T\mathbf{A}\mathbf{W}^T\right). \quad (4)$$

To apply their conclusion, we can define $\mathbf{A} = \mathbf{X}^T$ which in effect treats each row of $\mathbf{X}$, a vector indicating the feature values in each sample, as a sample. Thus, there are D N-dimensional samples to be clustered. Once the clustering is accomplished, we can readily translate the cluster membership to $\mathbf{W}$ according to the relationship discussed above.

Remark: (1) The clustering tends to group the features whose values in each sample are similar. Thus, we can infer that features within one group are positively correlated.
(2) Applying the proposed normalization factor $\frac{1}{\sqrt{|\mathcal{G}_j|}}$ on the merging function is actually beneficial in the sense of preserving the pairwise data distance which is desirable for dimensionality reduction. To see this let's consider a toy example, suppose each sample has n duplicate features $\{x^k, k = 1, \cdots, n\}$, then according to (1) they will be merged together. Before the merging, the expectation of the distance (ED) between two different samples $\text{E}\left(\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) = \sum_{k=1}^{n} \text{E}\left((x_i^k - x_j^k)^2\right) = 2n\text{Var}\{x^k\}$, where E and Var denote expectation and variance respectively. If we apply the traditional merging function, the above ED after merging becomes $2\text{Var}\{nx^k\} = 2n^2\text{Var}\{x^k\}$, that is, the ED is enlarged by the group size n. However, if we apply our modified merging function, the ED after merging will be $2\text{Var}\{\frac{nx^k}{\sqrt{n}}\} = 2n\text{Var}\{x^k\}$ which is identical to the one before merging. Thus, we can see that for the traditional merging function the distance between the merged data will be distorted by the different group sizes but the proposed normalization factor can alleviate this issue. (3) Running a k-means algorithm on a $N \times D$ data matrix can still be difficult if both N and D are large. It is even impossible to load such a large data matrix into memory in some cases.

Algorithm 1 Our basic merging algorithm

1: Set $\mathbf{S} \leftarrow \mathbf{0}$, where $\mathbf{0}$ is a $d_s \times D$ all '0' matrix.
2: **for** $i = 1$ to N **do**
3: **for** seed index $s = 1$ to M **do**
4: $k \leftarrow h_s(i, d_s)$, $\alpha \leftarrow 2h_s(i, 2) - 3$.
5: $\mathbf{s}_{(k,:)} \leftarrow \mathbf{s}_{(k,:)} + \alpha\mathbf{x}_i$
6: **end for**
7: **end for**
8: Run k-means on $\mathbf{S}$ to obtain the merging indexes.

Handle large N with hashing: To handle the issue in remark (3), we propose to adopt hashing [16] to reduce the dimensionality of $\mathbf{X}^T$. Hashing can be viewed as a special form of random projection. Its advantage is that the projection matrix can be analytically worked out so we do not need to allocate memory for storing the projection matrix. In our case, we need a projection matrix $\mathbf{H} \in \mathbf{R}^{d_s \times N}$ to project $\mathbf{X}^T$ from $N \times D$ to $d_s \times D$. Using hash function, we can calculate $\mathbf{H}$ via:

$$h_{k,i} = \begin{cases} 2h_s(i,2) - 3, & h_s(i, d_s) = k, \ \forall s \in \{1 \cdots M\} \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

where $h_s(i, d_s)$ is a hash function which maps i to an integer between 1 and d_s. We use the hashing function provided by the work in [16]. s is the seed of hashing function and different seeds define different hashing functions. To ensure good performance, multiple (M) seeds are often used. In our implementation, we use 30 seeds.

Applying $\mathbf{H}$ to reduce the dimensionality of $\mathbf{X}^T$, we obtained a matrix $\mathbf{S} = \mathbf{H}\mathbf{X}^T \in \mathbf{R}^{d_s \times D}$. We call it 'signature matrix' and use it to approximate $\mathbf{X}^T$ in the clustering step. That is, we conduct the clustering directly on the signature matrix. In general, larger d_s produces better approximation [1]. In our implementation, d_s is set to 200-350 and we find that this setting is sufficient to obtain good performance.

Online algorithm to calculate signature matrix: In [16], the authors propose an efficient way to calculate $\mathbf{HX}^T$. However, directly applying their method to compute $\mathbf{S} = \mathbf{HX}^T$ requires loading $\mathbf{X}$ into memory. This will be impractical for large sized $\mathbf{X}$. To address this issue, we propose the following online algorithm which allows the data to be processed in a sequential manner. Note that the k^{th} row of $\mathbf{S}$ can be calculated via:

$$\mathbf{s}_{(k,:)} = \mathbf{h}_{(k,:)}\mathbf{X}^T = \sum_{i=1}^{N} h_{k,i}\mathbf{x}_i \tag{6}$$

where $\mathbf{h}_{(k,:)}$ denotes the k^{th} row of $\mathbf{H}$. According to the definition of $\mathbf{H}$ in Eqn. (5), there are at most M nonzero values in $\{h_{k,i}, k = 1 \cdots N\}$. Thus we can utilize the following update rule to calculate $\mathbf{S}$: once a $\mathbf{x}_i$ is loaded we update those rows of $\mathbf{S}$ that satisfy $\{k|h_{k,i} \neq 0\}$ via $\mathbf{s}_{(k,:)} \leftarrow \mathbf{s}_{(k,:)} + h_{k,i}\mathbf{x}_i$. The detailed algorithm of our basic merging method is shown in Algorithm 1.

3.2. Pseudo-supervised Kernel Alignment (PKA)

In the following, we make a further extension to the basic algorithm by leveraging the neighborhood structure of data. The key hypothesis employed here is the posteriori smoothness assumption, that is, *the posterior probability $P(l|\mathbf{x})$ varies smoothly over the data manifold. So the k nearest neighbors of a sample tend to share the same class label with it,* where l is the class label. Before elaborating this algorithm, we first introduce a kernel alignment formulation which inspires this extension.

Assuming that the class label is available for each sample, we can design a merging criterion such that after merging the features, the linear kernel matrix of the reduced data is maximally aligned with the ideal kernel $\mathbf{L}^T\mathbf{L}$, where $L \in \mathrm{R}^{C\times N}$ (C is the number of classes) is the class label indicator matrix. Each column is a binary vector with one and only one '1' entry, whose position indicates the class label of the sample. To evaluate this alignment, we choose unnormalized centralized kernel alignment [3] as the objective:

$$\begin{aligned}&\max_{\mathbf{W}\in\Omega} \langle \mathbf{X}^T\mathbf{W}^T\mathbf{WX}, \mathbf{L}^T\mathbf{L}\rangle \\ &= \max_{\mathbf{W}\in\Omega} \mathrm{trace}(\mathbf{WXL}^T\mathbf{LX}^T\mathbf{W}^T)\end{aligned} \tag{7}$$

where $\langle \mathbf{A}, \mathbf{B}\rangle = \mathrm{trace}(\mathbf{AB}^T)$ is the inner product between two kernel matrices. We assume that $\mathbf{X}$ is centralized. Hence, the dimensionality-reduced data $\mathbf{WX}$ and its resultant kernel matrix $\mathbf{X}^T\mathbf{W}^T\mathbf{WX}$ are centralized. As seen, Eqn. (7) takes a very similar form as Eqn. (3). Thus we can leverage its relationship with the k-means algorithm to develop an efficient solution. Note that although in the literature normalized kernel alignment is more commonly used, it cannot be related to the k-means alike algorithm. Thus we choose the unnormalized centralized kernel alignment which has also shown good performance in [3].

In the unsupervised case, the class label is unknown. However we can still leverage the posterior smoothness assumption to generate 'pseudo-supervised information'. More specifically, we assume that there are N classes for N samples. For the i^{th} sample, besides being assigned to the i^{th} class, it is also assigned to the classes of its k nearest neighbors. Formally, we introduce a matrix $\hat{\mathbf{L}} \in \mathrm{R}^{N\times N}$ and set $\hat{\mathbf{L}}_{i,j} = 1$ if $j \in \mathcal{N}(i,k)$ and otherwise 0, where $\mathcal{N}(i,k)$ denotes a set of indexes which are the $k+1$ nearest neighbors of sample i (including sample i itself). Substituting $\hat{\mathbf{L}}$ into Eqn. (7), we obtain the following optimization problem (we call it Pseudo-supervised Kernel Alignment (PKA)):

$$\max_{\mathbf{W}\in\Omega} \ \mathrm{trace}\left(\mathbf{WX}\hat{\mathbf{L}}^T\hat{\mathbf{L}}\mathbf{X}^T\mathbf{W}^T\right) \tag{8}$$

Defining $\mathbf{Z} = \mathbf{X}\hat{\mathbf{L}}^T$, we can see that $\mathbf{z}_{(:,i)} = \sum_{j\in\mathcal{N}(i,k)} \mathbf{x}_j$. Thus, once the knn graph $\mathcal{N}$ is known, we can efficiently calculate $\mathbf{Z}$ and solve Eqn. (8) by using our basic method.

For high-dimensional data, it may be computationally expensive to calculate the knn graph. To handle this problem, we can firstly apply the basic method to obtain intermediate lower-dimensional data and then construct the knn graph from it. After that, we can run the PKA algorithm to attain the final reduction function. Certainly, this additional knn graph construction step will hurt the scalability of this algorithm in the case of large sample size. However, it may be a suitable choice for the medium-sized dataset since its performance is usually better than the basic method.

3.3. Exploring Bipolar Merging Function

Traditional merging functions only allow the addition operation. In this section, we show that subtraction operation can be also incorporated via our method. The motivation of this extension is to handle negatively correlated features. For example, let's assume that the p^{th} feature is the flipped version of the q^{th} feature, that is, $\mathbf{x}_{i,p} = -\mathbf{x}_{i,q}\ \forall i$. Obviously one of these two features is redundant. However, if we merge (add) them, their values will be canceled out. However, if we subtract them the result will be $\mathbf{x}_{i,p} - (-\mathbf{x}_{i,q}) = 2\mathbf{x}_{i,p}$. In this way, their information will be preserved by one dimension and we can obtain a more compact representation. Note that this extension is equivalent to allowing the entries of $\mathbf{W}$ to be negative. We call this scheme of feature merging 'Bipolar Merging'. Applying this idea to Eqn. (3), we obtain a problem:

$$\begin{aligned}&\max_{\mathbf{W}} \ \mathrm{trace}\left(\mathbf{WXX}^T\mathbf{W}^T\right) \\ &s.t.\ \ |\mathbf{W}||\mathbf{W}^T| = \mathbf{I},\ \ w_{i,j} \in \{0, 1/\sqrt{\lambda_i}, -1/\sqrt{\lambda_i}\} \\ &\qquad \lambda_i = \|\mathbf{w}_{(i,:)}\|_0\ \ \forall i\end{aligned} \tag{9}$$

where, $|\cdot|$ computes the element-wise absolute value.

At the first glance, this problem is difficult to solve. However, the following Theorem suggests that it can be reduced to Eqn. (3) with only one additional constraint on $\mathbf{W}$ (Please refer to the supplemental material for the proof).

Theorem 1 *Let* $\mathbf{W}^+$ *and* $\mathbf{W}^-$ *denote the positive and negative part of* $\mathbf{W}$ *respectively, that is,* $\mathbf{W} = \mathbf{W}^+ - \mathbf{W}^-$, $\mathbf{W}^+_{ij} \geq 0$, $\mathbf{W}^-_{ij} \geq 0$. *Define* $\widetilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ -\mathbf{X} \end{pmatrix}$ *and* $\widetilde{\mathbf{W}} = \begin{pmatrix} \mathbf{W}^+ & \mathbf{W}^- \\ \mathbf{W}^- & \mathbf{W}^+ \end{pmatrix}$. *Then the problem in Eqn. (9) is equivalent to the following problem:*

$$\max_{\widetilde{\mathbf{W}} \in \Omega \cap \mathcal{S}} \quad \mathrm{trace}\left(\widetilde{\mathbf{W}}\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T\widetilde{\mathbf{W}}^T\right), \tag{10}$$

where Ω is the structure constraint imposed in Eqn. (3) and $\mathcal{S}$ is the underlying symmetry structure constraint introduced by the definition of $\widetilde{\mathbf{W}}$. We prove that (see supplemental material) Eqn. (10) can be solved by treating $\widetilde{\mathbf{X}}^T = \left(\mathbf{X}^T, -\mathbf{X}^T\right) \in \mathrm{R}^{N \times 2D}$ as a data matrix with $2D$ N-dimensional samples. Reducing the dimensionality of $\mathbf{x}_i$ from D to d is equivalent to clustering the $2D$ samples into $2d$ clusters in the following way:

- Randomly initialize the cluster assignment of the first d clusters.
- Initialize the cluster assignment of the $(i+d)^{th}$ $(\forall i \leq d)$ cluster from the assignment in the i^{th} cluster by following the rule: if the i^{th} $(\forall i \leq d)$ cluster contains the data indexes $p \leq D, q > D$, then the $(i+d)^{th}$ cluster must contain the data indexes $p + D$ and $q - D$.
- Run standard k-means on $\widetilde{\mathbf{X}}^T$. Translate the clustering result into $\widetilde{\mathbf{W}}$ and the merging function.

Note that hashing can be applied here too, that is, we can replace $\widetilde{\mathbf{X}}^T$ with the signature matrix $(\mathbf{S}, -\mathbf{S})$. Also, this bipolar merging scheme can be readily applied to PKA.

4. Application and Experiment

To show the significance of the proposed methods, we introduce three applications and compare our methods with other alternative algorithms for each application.

4.1. Application I: Dimensionality Reduction for BoF Based Image Representation

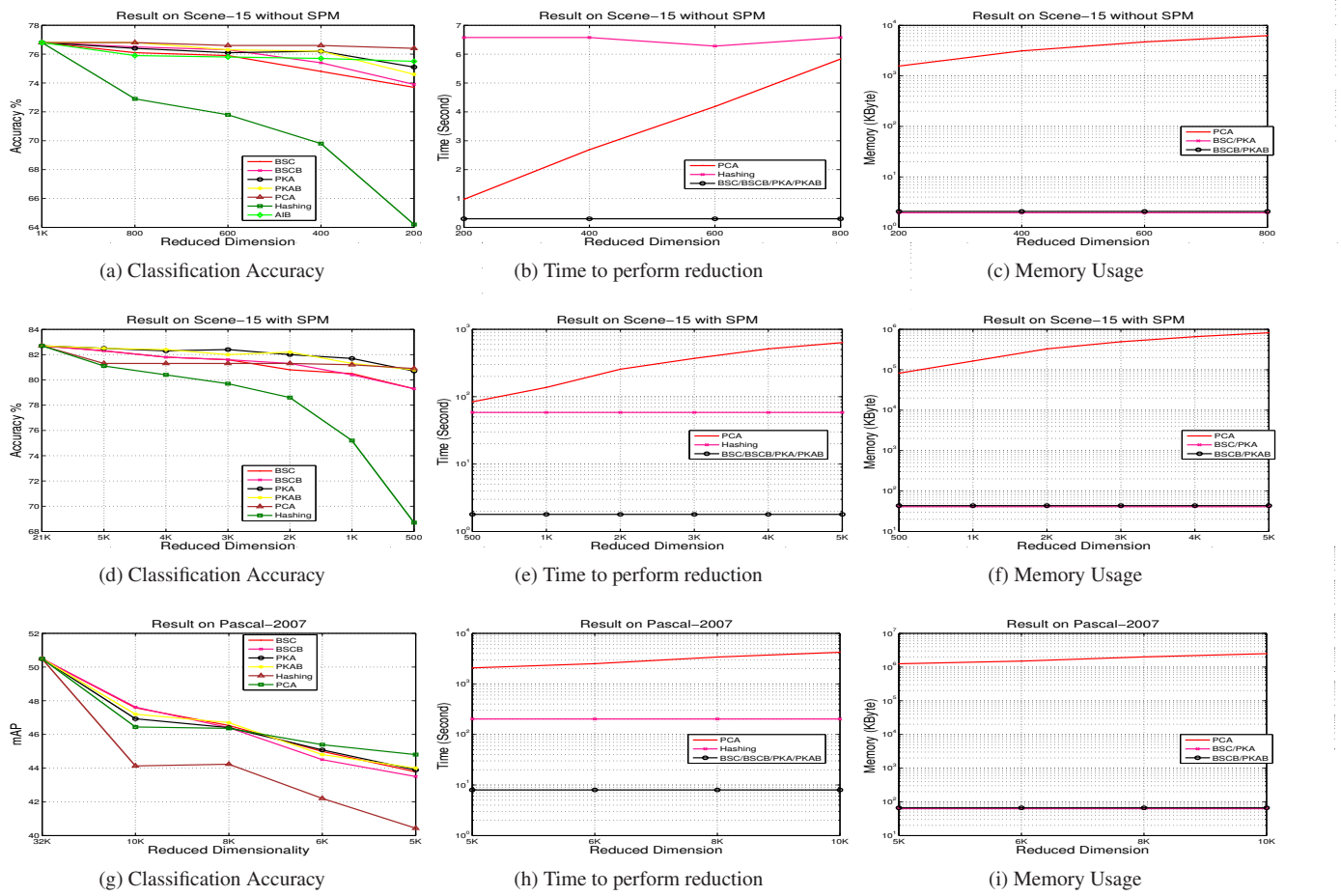
Problem Introduction: Large-sized codebooks and the use of Spatial Pyramid (SPM) often make the dimensionality of BoF image representation very high. In this application, we evaluate various reduction methods via the classification accuracy after reduction, time of performing reduction on whole dataset and the memory usage. For the reason mentioned in Introduction, we mainly focus on reducing high-dimensional data to medium-dimensional ones.

Experimental Setting: Two datasets, Scene-15 and Pascal-2007, are used. For Scene-15, we follow the setting in [12] to extract 21000-D image representation (1000-word dictionary with a 21-grid SPM). For Pascal-2007, we follow the setting in [17] to extract 32000-D representation (4000-word dictionary with a 8-grid SPM). To compare with the unsupervised AIB algorithm in [10] which does not scale well, we also evaluate on Scene-15 by avoiding applying SPM. Linear SVM is used as the classifier.

Seven methods are compared, including PCA, Hashing [16], AIB [10], the proposed basic merging algorithm (BSC in short), PKA and their bipolar variants (BSCB, PKAB in short). For PKA and PKAB, we firstly create the intermediate representation by using our BSC algorithm to reduce the dimensionality (to 200 for Scene 15 and 4000 for Pascal). Then we build the knn graph based on the intermediate features and re-run the dimensionality reduction with PKA and PKAB. We set the neighborhood size $k = 10$ for all the experiments.

Result Analysis: The results are shown in Fig. 1. As seen: (1) Merging-based reduction requires much less computational time than Hashing and PCA. For merging and Hashing, their computational time is independent of the reduced dimensionality while the time used by PCA linearly increases with it. (2) Merging-based reduction incurs less memory usage, e.g. it needs less than 100K in all settings. In contrast, the memory cost of PCA can be as large as over 1GB! (3) In terms of classification performance, PCA tends to perform slightly better when the reduced dimensionality is low while our methods outperform PCA at higher dimensions. This result is probably because when targeted dimension is high, PCA needs to estimate too many parameters $(d \times D)$ and this could become unreliable. Merging, however, only needs to infer $1 \times D$ parameters. (4) Our methods significantly outperform Hashing which is the most competitive method in terms of computational efficiency. This is because hashing cannot leverage any data-dependent information for dimensionality reduction as our methods do. (5) As seen in the experiment on Scene-15, our PKA makes further improvement over its basic counterpart BSC. Particularly, in Scene-15 with SPM, the PKA and PKAB achieve the overall best performance. PKA shows similar performance as BSC on Pascal-2007. This is probably because Pascal-2007 is a challenging dataset and the neighborhood of a sample obtained via low-dimensional intermediate features is less likely to share the same class label. As a result, the posteriori smoothness assumption taken in PKA cannot be well satisfied. However, we do not observe any adverse impact by applying PKA on this dataset. [1]

[1] Actually, we observe that the performance of PKA is comparable to



(a) Classification Accuracy (b) Time to perform reduction (c) Memory Usage

(d) Classification Accuracy (e) Time to perform reduction (f) Memory Usage

(g) Classification Accuracy (h) Time to perform reduction (i) Memory Usage

Figure 1: Reduction performance comparison on Scene-15 and Pascal-2007. Top row: comparison on Scene-15 without employing SPM. Middle row: comparison on Scene-15 with SPM. Bottom row: comparison on Pascal-2007.

4.2. Application II: Learning Better Compact Local Binary Pattern

Problem Introduction: Local Binary Pattern [14] is a very efficient image representation for texture classification and scene classification [20]. It compares the binary relationship (larger or smaller) of a center pixel intensity with its 8-neighborhood to produce a binary code for the center pixel. Then an image representation can be obtained by calculating the histogram of all possible binary patterns.

One improvement over LBP is to consider a larger neighborhood, e.g. the neighborhood shown on the right of Fig. 2 (we call it LBP-D5). However, a larger neighborhood will exponentially increase the number of possible binary patterns and result in very high-dimensional histograms (the number of possible patterns for LBP-D5 is $2^{16} = 65536$). To handle this issue, in [7] the authors use a vector quantization (VQ) based method to cluster different binary patterns. After the clustering, the binary patterns in the same clusters are viewed as being equivalent and the total number of possible patterns is then reduced. This is essentially a merging-based reduction. However, the clustering used in their method is purely appearance-based and it largely ignores the semantic correlation between the patterns having different appearances in a given task.

By implementing their VQ method, we find that its performance is good for texture classification but less satisfying for scene classification. This is probably because in scene images many visually dissimilar patterns are semantically correlated. For example, the patterns in 'sky' and 'cloud' are not visually similar but they are frequently co-occurred. This motivates us to apply our merging methods to this problem since they can exploit the co-occurrence information between features.

Experimental Setting: We compare the performance of our methods with VQ, PCA and Hashing in reducing the dimensionality of LBP-D5 histogram. The performance

BSC even if we randomly select the k nearest neighbors for each sample.

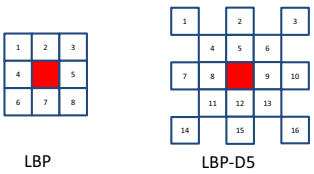


Figure 2: The neighborhood structure of LBP and LBP-D5.

of LBP is also included as the baseline. In particular, to test their generalization performance we learn the reduction function (PCA projection and merging indexes) on a separate dataset and evaluate the classification performance on Scene-15. For the separate dataset, we use Holiday dataset [8] – a popular image retrieval dataset which contains around 1600 trip photos.

Result Analysis: The result is shown in Table 1. As seen, the reduced LBP-D5 histogram obtained via our methods achieves much better performance than the LBP baseline and its advantage is more pronounced with the increase of the targeted dimensions. VQ-based reduction, however, does not bring much improvement over LBP. In fact, its performance is even worse than LBP when $d = 256$. As expected, the performance of hashing is not good enough due to its data-independent nature. Interestingly, PCA does not perform well for this task either. This is probably due to the same reason discussed in section 4.1 (too many parameters to be estimated). For the two proposed methods, the use of PKA (PKAB) again shows its advantage over BSC (BSCB), especially when the reduced dimension is low.

4.3. Application III: Dimensionality Reduction for High-dimensional Local Feature

Problem Introduction: High-dimensional image representation e.g. a pooled coding vector + SPM or a histogram of LBP-D5, often has better descriptive power than the low-dimensional ones, e.g. SIFT and LBP. Thus better classification performance is expected if we apply them to describe a local patch/region. However, directly applying them will generate very high-dimensional local features (HDLF in short) which will cause many problems in implementing the BoF model. For example, it will be difficult to load a large number of HDLF into memory to learn a dictionary (20K LBP-D5 feature will need 9.7G RAM). Thus, dimensionality reduction on HDLF is necessary. Moreover, since each image generates a large number of HDLF, the efficiency of performing dimensionality reduction becomes very important. Thus, the proposed merging methods will have significant advantages. However, what is performance of the dimensionality-reduced HDLF? How to choose the dimensionality reduction methods for this task? In the following parts, we answer these questions by conducting two experiments with two types of HDLF.

Experimental Setting: In the first experiment, we create the HDLF by using the pooled coding vector plus SPM (we call it HDLF-I for short) and evaluate its performance on the 14-class subset of Caltech-USD birds-200-2011 dataset [22]. We make such a choice because we find that HDLF-I achieves better performance for fine-grained image classification. To extract HDLF-I, We firstly follow [22] to extract color-SIFT feature and encode them by LSC coding [12] with a 1000-word dictionary. Then we crop a 40×40 subregion around each densely sampled point as a local patch and describe it by pooling the coding vectors within. A simple $1 + 2 \times 2$ SPM is also applied to encode the local geometry which gives us a 5000-D HDLF. We then reduce its dimensionality to 512 and encode it with sparse coding [21] since we find that it significantly outperforms other coding methods in encoding HDLF-I. As for the lower-dimensional local feature baselines, we use the color-SIFT and encode it by LSC (Base-I) and sparse coding (Base-II).

Table 2: Classification performance of the color-SIFT baselines and the methods using HDLF-I with different dimensionality reduction approaches on Caltech-USD Birds-200-2011 dataset. Evaluated by mean average precision (mAP).

Methods	Base-I	Base-II	Hashing	BSC	BSCB
mAP	46.5	42.8	44.6	**52.6**	**52.6**

Table 3: Classification performance of the LBP baselines and the methods using HDLF-II with different dimensionality reduction approaches on Scene-15 dataset.

Methods	LBP	VQ	Hashing	BSC	BSCB
Accuracy	82.2	82.5	80.5	**83.63**	83.50

For the second experiment, we extract the LBP-D5 histogram on 16×16 pixel-sized local patches and this produces 65536-dimensional HDLF (we call it HDLF-II for short). Again, its dimensionality will be reduced to 512 for later processing. We apply the recently developed VLAD [9] coding to create the image representation. We use linear SVM and conduct the evaluation on Scene-15. We apply the same coding and classifier on LBP as our baseline.

These two problems involve both high feature dimensions and large sample size (the total number of local features in the training set is over one million.) Thus PCA and PKA are not suitable for this task and we only compare Hashing, BSC and VQ (only in HDLF-II) here.

Result Analysis: The result for the first experiment is shown in Table 2. As seen, the use of HDLF-I significantly outperforms the baseline algorithms (color-SIFT with LSC and sparse coding). It achieves around 6 percent improvement over the best baseline (color-SIFT with LSC). We emphasize that this achievement relies on the right choice of dimensionality reduction method. For example, if we simply apply Hashing instead of our methods, the performance

Table 1: Reduction performance of PCA, VQ, Hashing and proposed methods on Scene-15 dataset with LBP-D5 Feature.

Dimension	LBP	PCA	VQ	Hashing	BSC	PKA	BSCB	PKAB
256	76.1	72.7	72.6	63.0	77.4	**78.1**	76.8	**78.0**
512	-	72.9	75.5	68.9	77.9	78.5	78.6	**79.1**
1024	-	73.3	76.8	72.9	**79.4**	**79.5**	**79.6**	**79.7**

of the reduced HDLF is even inferior to the best baseline.

The result for the second experiment is shown in Table 3. As seen, the reduced LBP-D5 feature can outperform LBP if BSC/BSCB is applied. In contrast, the other three dimensionality reduction methods (PCA, VQ and Hashing) fail to maintain the good descriptive power of LBP-D5 after the reduction.

In sum, we conclude that applying the proposed methods on HDLF can produce better local feature than the traditional lower-dimensional local feature.

4.4. Discussion

The bipolar merging is designed to further group the negatively correlated features together and thus 'save more dimensions' for the informative features. However, in the above experiments, we do not observe significant improvement over the BSC and PKA by using their bipolar variants. This is probably due to two reasons: (1) Our data do not have many negatively correlated features. (2) We do not reduce the feature to very low dimensions and thus the advantage of 'saving more dimensions' is not pronounced. To verify our explanation, we create a synthesized data set by concatenating the original feature $\mathbf{x}_i$ with their flipped version $-\mathbf{x}_i$ on Scene-15 (with SPM). Thus, the dimensionality of each sample becomes 42000. We then reduce the feature dimension to 50 by using BSC and BSCB. Examining their classification performance, we do observe clear advantage of BSCB – BSCB achieves the accuracy 72.49% while BSC only attains 70.8%. This result and the comparable performance of BSC and BSCB in the previous experiments also suggest that utilizing bipolar merging function is a safe choice: it will not decrease the performance and may obtain additional improvement in certain circumstances.

5. Conclusion

In this paper, we propose a scalable unsupervised merging algorithm and one extension to achieve the efficient dimensionality reduction. Through three applications, we demonstrate their superior performance in creating more powerful image representation.

6. Acknowledgement

The authors would like to thank the support of Australian Research Council (ARC) Linkage Grant LP0991757.

References

[1] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD*, 2001. 2, 3

[2] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 1

[3] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *JMLR*, 2012. 4

[4] C. H. Q. Ding and X. He. *K*-means clustering via principal component analysis. In *ICML*, 2004. 3

[5] T. T. Do, L. Gan, Y. Chen, N. Nguyen, and T. D. Tran. Fast and efficient dimensionality reduction using structurally random matrices. In *ICASSP*, pages 1821–1824, 2009. 2

[6] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *ECCV*, pages 179–192, 2008. 1, 2

[7] S. Hussain and B. Triggs. Visual recognition using local quantized patterns. In *ECCV (2)*, pages 716–729, 2012. 1, 6

[8] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008. 7

[9] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 7

[10] J. Liu and M. Shah. Learning human actions via information maximization. *CVPR*, 2008. 2, 5

[11] J. Liu, Y. Yang, and M. Shah. Learning semantic visual vocabularies using diffusion distance. In *CVPR*, pages 461–468, 2009. 2

[12] L. Liu, L. Wang, and X. Liu. In defence of soft-assignment coding. In *CVPR*, 2012. 5, 7

[13] L. Liu, L. Wang, and C. Shen. A generalized probabilistic framework for compact codebook creation. In *CVPR*, 2011. 1, 2

[14] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, Jan. 1996. 6

[15] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 1

[16] Q. Shi, H. Li, and C. Shen. Rapid face recognition using hashing. In *CVPR*, pages 2753–2760, 2010. 2, 3, 4, 5

[17] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR 2010*. 5

[18] L. Wang, L. Zhou, and C. Shen. A fast algorithm for creating a compact and discriminative visual codebook. In *ECCV*, 2008. 1, 2

[19] J. M. Winn, A. Criminisi, and T. P. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005. 2

[20] J. Wu and J. M. Rehg. Centrist: A visual descriptor for scene categorization. *IEEE TPAMI.*, 33(8):1489–1501, 2011. 6

[21] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 7

[22] B. Yao, G. R. Bradski, and F.-F. Li. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, pages 3466–3473, 2012. 7

[23] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001. 3