Locality Preserving Clustering for Image Database

Xin Zheng[1]*, Deng Cai*, Xiaofei He[2]*, Wei-Ying Ma*, Xueyin Lin[1]

*Microsoft Research Asia
Beijing, China
cai_deng@yahoo.com
wyma@microsoft.com

[1]Key Lab of Pervasive Computing,
Tsinghua University
Beijing, China
zhengxin99@mails.tsinghua.edu.cn
lxy-dcs@mail.tsinghua.edu.cn

[2]Computer Science Dept.
University of Chicago
xiaofei@cs.uchicago.edu

ABSTRACT
It is important and challenging to make the growing image repositories easy to search and browse. Image clustering is a technique that helps in several ways, including image data preprocessing, user interface designing, and search result representation. Spectral clustering method has been one of the most promising clustering methods in the last few years, because it can cluster data with complex structure, and the (near) global optimum is guaranteed. However, existing spectral clustering algorithms, like Normalized Cut, are difficult to handle data points out of training set. In this paper, we propose a clustering algorithm named Locality Preserving Clustering (LPC), which shares many of the data representation properties of nonlinear spectral method. Yet LPC provides an explicit mapping function which is defined everywhere, both on training data points and testing points. Experimental results show that LPC is more accurate than both "direct Kmeans" and "PCA + Kmeans". We also show that LPC produces in general comparable results with Normalized Cut, yet is more efficient than Normalized Cut.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval –*Clustering*; I.4.m [**Image Processing and Computer Vision**]: Miscellaneous –*Image Clustering*.

General Terms
Algorithms, Performance, Experimentation, Theory

Keywords
Locality preserving clustering, Locality preserving projections, Spectral clustering, Image clustering

1. INTRODUCTION
Image repositories are growing rapidly nowadays. It is important and challenging to make the repositories easy to search and browse. Image clustering is a technique that helps in several ways. For example, image clustering can be used as a preprocessing step that improves the speed and performance of content-based image retrieval (CBIR) [7]. For users who want to browse image database, hierarchical clusters of images will be useful for designed a convenient user interface (UI) [13]. Even in powerful search engines, image clustering will help to make more meaningful representation of query results [3].

Image clustering is a technique that associates each image in database with a class label such that the images associated with the same label are similar to each other. Traditional clustering methods (such as Kmeans, Gaussian Mixture Model (GMM), etc) used in image clustering often get poor results in complex data, e.g. data points sampled from a non-linear manifold. Image database might be one of such examples because images with different concepts are generally not well-separated using traditional clustering methods. The reason why traditional methods failed is that the typical Gaussian distribution (or mixture of Gaussian distributions) is defined on the Euclidean space, and hence it can not always describe the data points sampled from a non-linear manifold. Note that, Parzen window [5] might be an exception, but it remains unclear how to apply it for clustering. In order to deal with such situations, spectral clustering method was proposed and has been successfully used in several applications. For example, Normalized Cut (NCut) [18] was used for image segmentation, video structuring [13], scene detection [14], video segmentation [16] and motion segmentation [17]. Ng et.al proposed a spectral clustering algorithm that extends classical Normalized Cut and gets better results [12]. Although Laplacian Eigenmaps [1] is focused on embedding of manifold data rather than clustering, it is easy to see that Eigenmaps is essentially equivalent to the embedding step of Normalized Cut in the case of certain kernel (e.g. Gaussian kernel).

In spite of the success of Normalized Cut and Eigenmaps on manifold data embedding and clustering, they can not provide us with an explicit mapping function. Actually, when dealing with new data points, similarities between the new points and all training data are needed [2]. The computation of the similarities can be very complicated due to the large size of training set. In order to solve this problem, Locality preserving projections (LPP) was proposed recently [8] and has bees used in document representation [1] and face recognition [10]. As a spectral embedding method, LPP shares many of the data representation properties of nonlinear methods such as Laplacian Eigenmaps. Yet LPP provides an explicit mapping function which is defined everywhere, either on training data points and testing points.

---

*This work was done during Xin Zheng's internship in MSRA

[1]This work is partially supported by National Key Basic Research and Development Program 2002CB312101.

MM'04, October 10–16, 2004, New York, New York, USA.

An interesting modification of spectral clustering is the out-of-sample extension [2]. Although the out-of-sample formula seems to be more useful for supervised learning than traditional clustering algorithm, it does help for spectral clustering. In fact, the state-of-the-art spectral clustering algorithms involve two steps, dimensionality reduction and traditional clustering like Kmeans. Out-of-sample extension is crucial for dimensionality reduction due to the saved computation. In detail, only the affinities between each data point and the points in a subset of the data, rather than the whole data set, are to be computed.

In this paper, we propose a clustering algorithm named Locality Preserving Clustering (LPC) which is a clustering method base on modified LPP. Technically, LPP solves a generalized eigenvalue problem, but the original LPP algorithm stated in [8] may mix true solutions (eigenvectors) with pseudo solutions and the trivial solution. We show that the pseudo solutions and the trivial solution carry no useful information and thus waste the dimensions of embedded results. In our version of LPP, these two kinds of useless solutions are well-handled, and a renormalization step is added to improve the robustness of the algorithm. Theoretic analysis is provided to show why modified LPP gives better and more robust results. In the algorithmic framework, we treat LPP as a feature selection method, and adopt a traditional clustering method (Kmeans in this paper) in the resultant space of LPP. We apply our methods on the problem of image clustering. Experimental results show that LPC is more accurate than both "direct Kmeans" and "PCA + Kmeans". We also show that LPC produces in general comparable results with Normalized Cut, yet is much faster than Normalized Cut.

It would be worthwhile to highlight several aspects of our proposed algorithm here:

- LPC is defined everywhere, but traditional spectral clustering are only defined on training data points. This implies that clustering on incremental data points using LPC is much straightforward and faster than using traditional spectral clustering.
- As a spectral method, the eigenvalue problem of LPC scales with the number of feature dimensions, while the eigenvalue problem of NCut scales with the number of data points. In most of image applications, the number of images in database is much larger than the number of feature dimensions. In this sense, LPC is much more efficient than NCut.
- Since LPC is designed for preserving local structure, it is likely that the neighboring points in the low dimensional space are also near to each other in the original high dimensional space. So the clustering results of such data are probably more reasonable than that of data generated by PCA.

The rest of this paper is organized as follows: Section 2 describes the proposed Locality Preserving Clustering (LPC) algorithm which is based on modified LPP. Theoretic analysis of the modification is discussed in Section 3. The experimental results are shown in Section 4, followed by the conclusion in Section 5.

2. LOCALITY PRESERVING CLUSTERING (LPC)

Locality preserving clustering (LPC) is fundamentally based on LPP and Kmeans clustering. The algorithmic procedure is stated below:

ALGORITHM 1. (LPC) Suppose we are given a set of M data points with N-dimensional features denoted by matrix $X_{M\times N} = \left[\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{M}\right]^T$.

1. **Constructing the adjacency graph** (same as LPP): Let ***G*** denote a graph with ***M*** nodes. We put an edge between nodes ***i*** and ***j*** if ***i*** is among ***N*** nearest neighbors of ***j*** or ***j*** is among ***N*** nearest neighbors of ***i***.

2. **Choosing the weights** (same as LPP): Affinity matrix ***W*** is a sparse symmetric matrix generated from Heat kernel: If nodes ***i*** and ***j*** are connected, put

$$W_{ij} = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{\sigma}\right) \tag{1}$$

3. **Full-rank** (additive): Calculate $\tilde{X}$, the orthonormal basis of column space of $\left[\mathbf{1}\ X\right]$ ($span\left(\left[\mathbf{1}\ X\right]\right)$) using singular value decomposition (SVD):

$$\left[\mathbf{1}\ X\right] = \tilde{X}\Lambda V^T \tag{2}$$

where $\mathbf{1}$ denotes a column vector consisted of all one and only the non-zero singular values are reserved in Λ. This step results in a linear mapping function:

$$\mathbf{x}_i \to \tilde{\mathbf{x}}_i = E_{SVD}^T\left[1\ \mathbf{x}_i^T\right]^T, E_{SVD}^T = \Lambda^{-1}V^T \tag{3}$$

4. **Embedding** (modified): Solve the following generalized eigenvalue problem:

$$\tilde{X}^T L\tilde{X}\mathbf{a} = \lambda\tilde{X}^T D\tilde{X}\mathbf{a} \tag{4}$$

Let the column vectors $\left\{\hat{\mathbf{a}}_1,\cdots,\hat{\mathbf{a}}_r,\mathbf{a}_{r+1},\cdots,\mathbf{a}_t\right\}$ be the top (smallest) solutions of the above equation, ordered according to their eigenvalues, $0 = \lambda_1 = \cdots = \lambda_r < \lambda_{r+1} \le \cdots \le \lambda_t$. Calculate the orthogonal basis starting from $\mathbf{1}$, i.e. $\left\{\mathbf{a}_1\left(=\mathbf{1}\right),\cdots,\mathbf{a}_r\right\}$ s.t. $span\left(\mathbf{a}_1,\cdots,\mathbf{a}_r\right) = span\left(\hat{\mathbf{a}}_1,\cdots,\hat{\mathbf{a}}_r\right)$. The embedding in this step is as follows:

$$\tilde{\mathbf{x}}_i \to \mathbf{y}_i = E_{LPC}^T\tilde{\mathbf{x}}_i, E_{LPC}^T = \left[\mathbf{a}_2,\mathbf{a}_3,\cdots,\mathbf{a}_t\right] \tag{5}$$

The above two steps generate a linear embedding function:

$$\mathbf{x}_i \to \mathbf{y}_i = E_{LPC}^T E_{SVD}^T\left[1\ \mathbf{x}_i^T\right]^T \tag{6}$$

5. **Renormalization**: Project the embedding into unit sphere.

$$\mathbf{y}_i \to \mathbf{v}_i = \frac{\mathbf{y}_i}{\|\mathbf{y}_i\|} \tag{7}$$

6. **Clustering**: Perform Kmeans on renormalized embedded results $\{\mathbf{v}_i\}$.

3. THEORETICAL ANALYSIS

In this section we give the theoretical analysis of our algorithm.

3.1 Relation between LPP and Eigenmaps

LPP can be seen as generalized Laplacian Eigenmaps [1]. Laplacian Eigenmaps solves the generalized eigenvalue problem of

$$L\mathbf{y} = \lambda D\mathbf{y} \tag{8}$$

It is clear, according to definition of General Rayleigh Quotient, the eigenvector corresponding to the least eigenvalue of (8) is actually

$$\mathbf{y}_1 = \arg\min_{\mathbf{y}} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \tag{9}$$

and the *i*-th eigenvector is

$$\mathbf{y}_i = \arg\min_{\mathbf{y}^T D\mathbf{y}_j=0,\, j<i} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \tag{10}$$

Given $\mathbf{y}=\tilde{X}\mathbf{a}$, we found that the solution of (4) is

$$\mathbf{y}_1 = \arg\min_{\mathbf{y}\in span(\tilde{X})} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \tag{11}$$

and the *i*-th eigenvector is

$$\mathbf{y}_i = \arg\min_{\mathbf{y}\in span(\tilde{X}),\mathbf{y}^T D\mathbf{y}_j=0,\, j<i} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \tag{12}$$

Thus (4) is actually solving (8) in subspace $span(\tilde{X})$.

3.2 Motivation of modified LPP

The original LPP [8] solves the eigenvalue problem without preprocessing of data matrix:

$$X^T LX\mathbf{a} = \lambda X^T DX\mathbf{a} \tag{13}$$

It then constructs the embedding results by

$$\mathbf{x}_i \to \mathbf{y}_i = A^T\mathbf{x}_i, A = (\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_t) \tag{14}$$

This procedure probably causes two kinds of problems.

1. **Pseudo solution**: A regular condition of generalized symmetric eigenvalue problem of (13) is that $X^T DX$ should be positive definite. However this condition is not guaranteed when the column rank of X is not full ($rank(X)<N$). Technically, when solving the broken-conditioned problem using pseudo inverse of $X^T DX$, there are some $\mathbf{a}\neq\mathbf{0}$, s.t. $X\mathbf{a}=\mathbf{0}$, which appear to be zero-eigenvectors during calculation. However, they are pseudo ones and bring no embedding information, due to that the embedded coordinates of the dimensions corresponding to a are all zero.

2. **Near trivial solution**: It is shown in literature of spectral graph theory (e.g. [4]) that $\mathbf{1}$ is the trivial eigenvector of (8) associated with eigenvalue of 0, where $\mathbf{1}$ denotes the column vector of all one. However, $\mathbf{1}$ is meaningless for embedding, since the coordinates of all embedded data points are identical in the dimension corresponding to $\mathbf{1}$. The original LPP (13) solve eigenvalue problem in *span(X)*. If $\mathbf{1}$ is not perpendicular to *span(X)* and is not located in *span(X)*, the original LPP probably produces some eigenvector $\mathbf{a}$ that is *near trivial*, i.e. $X\mathbf{a}\approx\mathbf{1}$. This implies that the embedded coordinates along some dimensions are dominated by constant components and carry little useful information.

The modifications which we present in Algorithm 1 are designed to avoid the problems mentioned above.

3.3 Justification of modified LPC

The two major modifications of the algorithms are: a) $\mathbf{1}$ is explicitly introduced to feature space, i.e. $[\mathbf{1}\, X]$ is used instead of $[X]$; b) $\tilde{X}$ is enforced to be a column-full-rank column-orthogonal matrix. The column-full-rank property of $\tilde{X}$ is necessary because $\tilde{X}^T D\tilde{X}$ need to be positive definite. We can further explain the significance of the modifications using the following theorem.

THEOREM 1. $\exists$ the unique eigenvector $\hat{\mathbf{a}}$ of eigenvalue problem (4) associated with eigenvalue of 0, s.t. $\tilde{X}\hat{\mathbf{a}}=\mathbf{1}$.

PROOF. See Appendix.

Theorem 1 implies that the employment of $[\mathbf{1}\, X]$ extends the space of optimization (see (11)) and brings $\mathbf{1}$ to the eigenspace of (4) associated with eigenvalue of 0. In the Embedding step, we reorganize the 0-eigen-space s.t. $\mathbf{1}$ is the first eigenvector. Therefore the meaningless eigenvector $\mathbf{1}$ is explicitly removed.

Note that LPP can be modified in different ways. For instance, He et.al [10] applied Principle Component Analysis (PCA) on *X* before conducting LPP. In their strategy, by subtracting the mean from *X*, $\mathbf{1}$ is orthogonal to *span(X)*. That avoids the "near trivial solution" problem. Further more, by preserving only a majority of information, say 98%, the remains is full-ranked. That avoids the "pseudo solution" problem. Nevertheless, it should be mentioned that such strategy does not guarantee optimal embedding. This is explained as follows.

THEOREM 2. Given data matrix *X*, suppose $\hat{X}$ is the results of PCA on *X* that subtracts the mean and keeps 100% information. $\tilde{X}$ is set as (2). Denote $span(\tilde{X}^{\mathbf{1}\perp}) = \mathbf{1}^{\perp} \cap span(\tilde{X})$. The solution of LPP on $\hat{X}$ is

$$\hat{\mathbf{y}}_1 = \arg\min_{\mathbf{y}\in span\left(\tilde{X}^{\perp\perp}\right)} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \qquad (15)$$

and

$$\hat{\mathbf{y}}_i = \arg\min_{\mathbf{y}\in span\left(\tilde{X}^{\perp\perp}\right), \mathbf{y}^T D\hat{\mathbf{y}}_j = 0,\, j<i} \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \qquad (16)$$

PROOF. It is easy to prove from the definition of General Rayleigh Quotient.

Comparing $\mathbf{y}_{i+1}$ and $\hat{\mathbf{y}}_i$ from Equation (11-12) and (15-16), it is clear that they are optimizing different criterion. They become the same if and only if $D=cI$ with I the identity matrix.

The justification of orthogonalization (2) is to improve numerical precision. If X is far from column-orthogonal, the generalized eigenvalue problem will be ill-posed. That will increase the time of convergence or/and reduce the precision [5].

The last but not least modification is to perform renormalization before further clustering. It is shown by Andrew Y. Ng, et.al that renormalization of embedded results helps when the intra-cluster connection degree varies across clusters [12]. Experimental results show that it also works on LPC.

4. EXPERIMENTS

We have designed three kinds of experiments and the results show the effectiveness of our proposed algorithm. Some other methods are implemented for comparison, i.e. Normalized Cut, PCA + Kmeans and direct Kmeans.

4.1 Data Corpora

We test the algorithms on a general-purpose image database, in which 79 categories of COREL are included. The image number included in each category is between 100 and 300, and the total number of image is 10,000. Each COREL category was treated as a human-labeled cluster and is used as groundtruth for our clustering task, while the multiple labels assigned with each images in the original CDs are ignored. Some sample images are shown in Figure 1.

Image feature used in our experiments is the union of color histogram and Color Texture Moment (CTM) proposed by Yu et.al [19]. CTM is a 48-dimensional feature that integrates the color and texture characteristics of an image in a compact form. The color histogram is calculated using 4*4*4 bins in HSI space. Thus an 112-dimensional feature vector is used for each image. The feature vector is normalized s.t. each image has a feature vector of norm 1.

4.2 Evaluation metric

We test the algorithms on several different subsets of the database. Each subset is a mixture of k randomly selected categories. For each experiment, mixed images together with the cluster number k are provided to the clustering algorithms, and the performance is evaluated by comparing the cluster label of each image given by algorithm with the groundtruth. Two metric, the accuracy (AC)

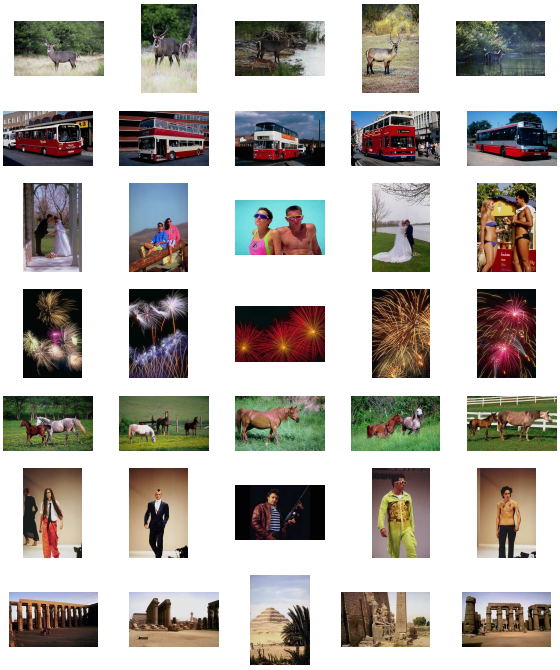


Figure 1. Sample images of some categories from data corpora. One row for each category. Category names are (from top to bottom): "Antelope", "Bus", "Couples", "Firework", "Horse", "Men", "Pyramid".

and the normalized mutual information ($\overline{MI}$), are used for evaluation. They are defined as follows. Suppose that r_i is the clustering result of a given image I_i and g_i is groundtruth, AC is defined by:

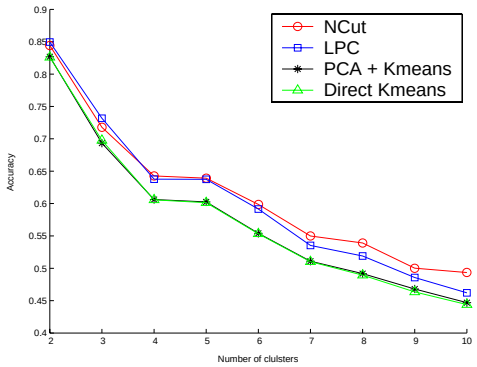
$$AC = \frac{\sum_{i=1}^{n} \delta\left(g_i, map\left(r_i\right)\right)}{n} \qquad (15)$$

where n denotes the total number of images in this experiment, $\delta(x,y)$ is the delta function that equals 1 if and only if $x=y$. $map(r_i)$ is the best mapping function that permute clustering labels to match the labels given by groundtruth. The Kuhn-Munkres algorithm can be used to obtain best mapping [11].

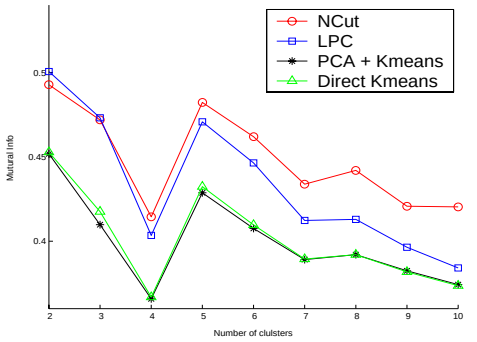
On the other hand, given clustering result $R = r_i$ with the groundtruth $G = g_j$, and denote $C_R = Range(R)$, $C_G = Range(G)$, the mutual information between them is defined by:

$$MI(R,G) = \sum_{s\in C_R, t\in C_G} p(s,t)\cdot\log_2 \frac{p(s,t)}{p(s)\,p(t)} \qquad (16)$$

where p(s), p(t) denote the probabilities that an arbitrary image in the subset belongs to the clusters s (in cluster result) or t (in groundtruth), respectively. p(s,t) denotes the joint probability that this image belongs to the clusters s and t at the same time. Suppose $H(R)$ and $H(G)$ denote the entropies of $p(s)$ and $p(t)$. $MI(R;G)$ varies between 0 and max($H(R)$;$H(G)$). We use normalized mutual information $\overline{MI}$ as the second metric.

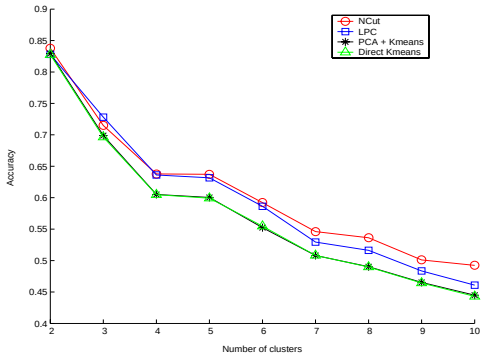


(a) Accuracy

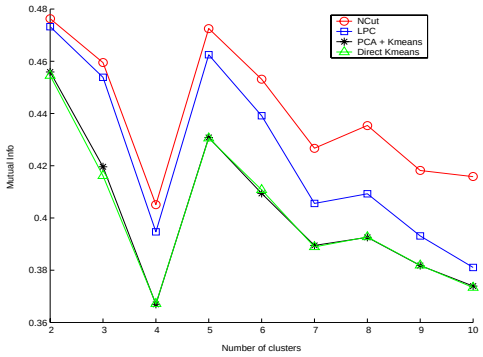


(b) Normalized Mutual Information

Figure 2. Clustering results comparison among NCut, LPC, PCA + Kmeans, and direct Kmeans. X-axis – number of clusters; Y-axis – Accuracy(a) or Normalized mutual information(b).



(a) Accuracy



(b) Normalized Mutual Information

Figure 3. Generalization capability comparison among NCut, LPC, PCA + Kmeans, and direct Kmeans. 70% data are used for training; all 100% data points are used for clustering; the precision of test data (the rest 30%) are shown in the figures. X-axis – number of clusters; Y-axis – Accuracy(a) or Normalized mutual information(b).

$$\overline{MI}(R,G) = \frac{MI(R,G)}{\max\left(H(R), H(G)\right)} \qquad (17)$$

It is obvious that the normalized mutual information $\overline{MI}$ takes values in [0; 1]. It reaches 1 if the clustering result is identical with the groundtruth, and becomes 0 if the clustering result is independent with the groundtruth. Unlike *AC*, $\overline{MI}$ is invariant with the permutation of labels. That is to say, $\overline{MI}$ does not need matching the clustering result and the groundtruth in advance.

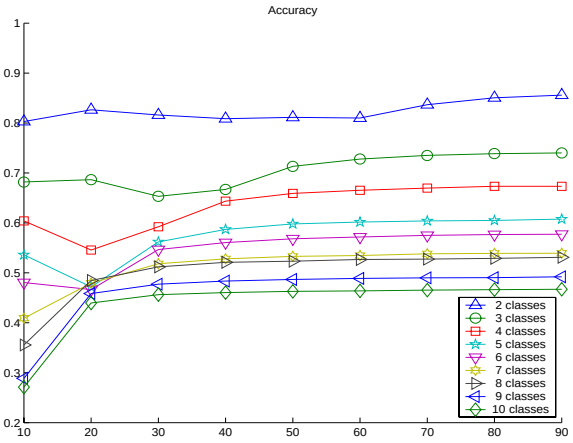
4.3 Image Clustering

The first experiment is designed to compare the performance of four algorithms:

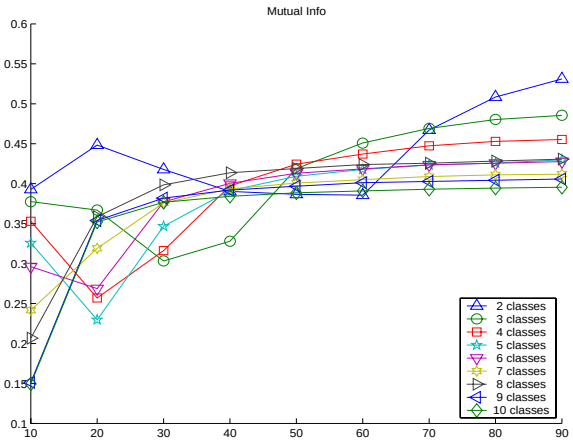
1. **Direct Kmeans**: Kmeans is performed directly on the 112-dimensional feature vectors.
2. **PCA + Kmeans**: PCA is firstly performed on the feature vectors. The reduced dimension of PCA is set to the minimal number that preserves at least 95% of the information. This number is about 40 for the 112- dimensional features. Then Kmeans is performed on the embedded data.
3. **LPC**: As described in Algorithm 1. The affinity matrix W is constructed using N nearest neighbors with N=10. We empirically choose the reduced dimension equal to (cluster number - 1).
4. **Normalized Cut**: Ng's version of Normalized Cut is used [12]. The affinity matrix W is defined exactly in the same way as LPC. The reduced dimension is equal to cluster number.

The cluster number k varies between 2 and 10, and is provided along with the data to all four algorithms, . For each cluster number k, 100 subsets are randomly selected from the corpora. All four algorithms are performed on each subset and their average of AC and $\overline{MI}$ are calculated over the 100 subsets. The comparison of performance is shown in Figure 2, and one example of clustering result of k=7 is shown in Figure 5.

It can be seen from Figure 2 that LPC outperforms "PCA + Kmeans" and "direct Kmeans". At the same time, we note that the reduced dimension of LPC (= 2; ⋯; 10) is greatly less than that of PCA (≈ 40) and direct Kmeans (= 112). Kmeans for LPC is thus much faster than Kmeans for PCA or direct Kmeans. The performance of LPC approaches that of Kmeans when the number of clusters increases. This is because it is more and more unlikely to be linearly separable with the increases of cluster number, which indicates the limitation of linear projection.



(a) Accuracy



(b) Normalized Mutual Information

Figure 4. Clustering Results improvement with the increasing percentage of training data. X-axis – percentage of training data; Y-axis – accuracy(a) or normalized mutual information(b).

As can be seen from Figure 2 that LPC has lower performance than Normalized Cut. Since LPC is LPP + Kmeans and Normalized Cut is Eigenmaps + Kmeans, this result is reasonable because LPP optimizes the same object function with Eigenmaps, yet within a smaller space (linear space of feature). However, as mentioned above, LPP offers explicit linear embedding operator on feature space. This property enables LPP to perform a fast embedding for out-of-sample data points, thus, enable LPC to cluster out-of-sample data. This is discussed in the next experiment in Section 4.4.

We currently do not conduct significance test of the difference of the performance between the four algorithms and other variations of LPP (e.g. that of He et.al [10]), and this should be part of a more thorough evaluation.

4.4 Generalization Capability

We also test generalization capability of LPC using the out-of-sample clustering procedure. The procedure is stated as follows: Given a training set *A* and a test set *B* both drawn from the same

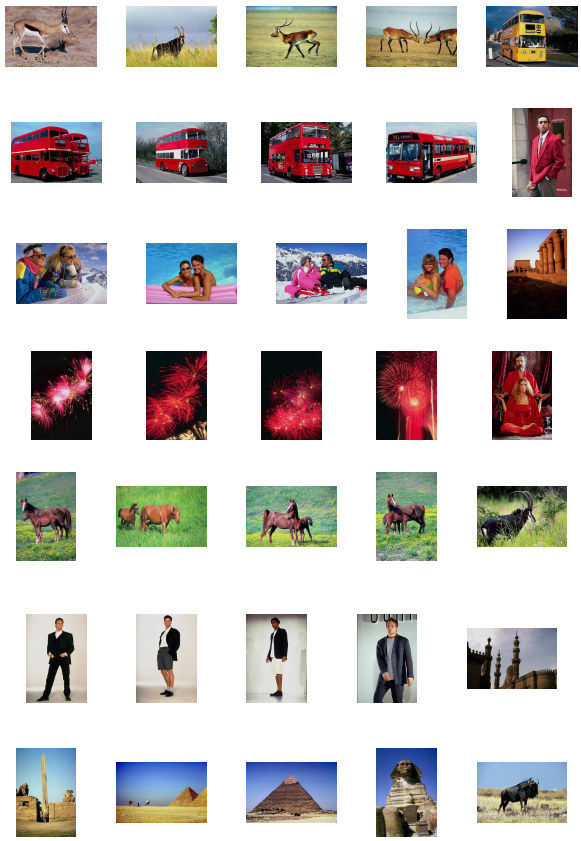


Figure 5. Results of LPC on COREL database. 7 categories. One row for each category. The rightmost position of each row shows one error example. Category names are (from top to bottom): "Antelope", "Bus", "Couples", "Firework", "Horse", "Men", "Pyramid".

distribution, learn an embedding function from *A* using PCA, LPP, and Eigenmaps, respectively. Then perform the embedding function to the test set *B* and re-cluster (using Kmeans) the union of embedded *A* and embedded *B*. Finally the precision of cluster label of *B* are calculated and compared among different algorithms.

We build PCA-based, LPP-based (LPC) and Eigenmaps-based (NCut) out-of-sample clustering algorithms, where PCA, Eigenmaps, LPP are regarded as three different choices of embedding methods.

All three algorithms are performed on each subset described in Section 4.3. The different thing is, for each of the subset, 70% data points are used as training data *A* and the rest 30% are used as newcomer data *B*. For comparison, direct Kmeans is also performed on $A \cup B0$, without learning of course, and the precision of cluster label of *B* is also calculated. The experiment results are shown in Figure 3.

It can be seen that Eigenmaps-based and LPP-based methods outperforms PCA-based method. Although Eigenmaps generates slightly better results than LPP, it is much slower than LPP . The

reason is that before Eigenmaps can perform embedding function on B the similarities between all test points in *B* and all training points in *A* should be calculated in advance [2]. Suppose there are N_A points in *A* and N_B points in *B*. Note *N* the dimension number of feature space and *k* the cluster number. For Eigenmaps, similarity computation needs *NA*NB*N* times multiplications and the embedding needs N_A*N_B*k times multiplications. So the total complexity of out-of-sample embedding in Eigenmaps is $O\left(N_A \times N_B \times (N+k)\right)$. While what LPP does is doing matrix multiplication between embedding function (matrix) and data matrix of B. So the complexity is $O(k \times N_B \times N)$. Typically, $N_A \gg N$ and $N_A \gg k$. This explains why LPC is much more efficient than Eigenmaps-based method.

4.5 Improvement with the Number of Training Samples

Since the training data set *A* and the newcomer data set *B* share the same distribution. It is a reasonable demand that the performance is improved with the number of training samples. We designed a third experiment to exam if the performance of LPP improves with the number of training samples, and empirically answer the question of how many percentage is enough to model the distribution for the data sets. At this time, 1*;* 000 subsets are selected for each category number *k*. The sample rate of training data (i.e. $|A|/(|A|+|B|)$) varies from 10%, 20% to 90%. After embedding operators are learnt from *A*, the whole data set are embedded and then clustered. The average accuracy and normalized mutual information are shown in Figure 4. It is clear that the performance improves with the number of training samples. And for this data corpora, 40% is enough for satisfying data model and embedding operator.

5. CONCLUSION

In this paper we proposed a Locality Preserving Clustering (LPC) algorithm. LPC shares many of the data representation properties of nonlinear spectral method yet LPC provides an explicit mapping function which is defined everywhere, both on training data points and testing points. So LPC shows powerful capability for data representation and computational efficiency at the same time.

6. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[2] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-Sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[3] Y. Chen, J. Z. Wang, and R. Krovetz. Content-based image retrieval by clustering. *In Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 193–200. ACM Press, 2003.

[4] F. R. Chung. *Spectral Graph Theory*. American Mathematical Society, Rhode Island, 1997.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.

[6] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[7] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. *In Proceedings of ICCV*, 2003.

[8] X. He and P. Niyogi. Locality preserving projections. *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[9] Xiaofei He, Deng Cai, Haifeng Liu and Wei-Ying Ma, Locality Preserving Indexing for Document Representation, *The 27th Annual International ACM SIGIR Conference (SIGIR'2004)* , July 2004.

[10] X. He, S. Yan, Y. Hu, and H.-J. Zhang. Learning a Locality Preserving Subspace for Visual Recognition, *IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[11] L. Lovasz and M. Plummer. *Matching Theory*. Akadémiai Kiadó, North Holland, Budapest, 1986.

[12] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[13] J-M. Odobez, D. Gatica-Perez, and M. Guillemot, Spectral Structuring of Home Videos. *International Conference on Image and Video Retrieval (CIVR'03)*, 2003.

[14] Z. Rasheed and M. Shah, A Graph Theoretic Approach for Scene Detection in Produced Videos. *Multimedia Information Retrieval Workshop*, 2003.

[15] K. Rodden, W. Basalaj, D. Sinclair, and K. R. Wood. Does organisation by similarity assist image browsing? *In Proceedings of CHI*, pages 190–197, 2001.

[16] J. Shi, S. Belongie, T. Leung, and J. Malik. Image and video segmentation: The normalized cut framework. *IEEE International Conference on Image Processing (ICIP)*, October, 1998.

[17] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. *In ICCV*, pages 1154–1160, 1998.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 2000.

[19] H. Yu, M. Li, H.-J. Zhang, and J. Feng. Color texture moments for content-based image retrieval. *IEEE International Conference on Image Processing*, pages 24–28, 2002.

Appendix A PROOF OF THEOREM 1

In this appendix, we prove Theorem 1 in Section 3.3.

PROOF. $span(\tilde{X}) = span([\mathbf{1}\ X])$. Since $\tilde{X}$ is the union of independent column vectors, $\exists$ the unique vector $\hat{\mathbf{a}}$ s.t. $\tilde{X}\mathbf{a}=\mathbf{1}$. It is clear that $\mathbf{1}$ is an eigenvector of (8) associated with eigenvalue of 0 because

$$\mathbf{1}^T L\mathbf{1} = \sum_{ij} D_{ij} - \sum_{ij} W_{ij} = 0 \qquad (18)$$

and

$$\mathbf{1}^T D\mathbf{1} = \sum_{ij} D_{ij} > 0 \tag{19}$$

So

$$\frac{\hat{\mathbf{a}}^T \tilde{X}^T L \tilde{X} \hat{\mathbf{a}}}{\hat{\mathbf{a}}^T \tilde{X}^T D \tilde{X} \hat{\mathbf{a}}} = 0 \tag{20}$$

According to definition of generalized Rayleigh quotient, $\hat{\mathbf{a}}$ is an eigenvector of (4) associated with eigenvalue of 0.□