Noname manuscript No.
(will be inserted by the editor)

Non-Parametric Kernel Learning with Robust Pairwise Constraints

Changyou Chen · Junping Zhang · Xuefang He · Zhi-Hua Zhou

Received: date / Accepted: date

Abstract For existing kernel learning based semi-supervised clustering algorithms, it is generally difficult to scale well with large scale datasets and robust pairwise constraints. In this paper, we proposed a new **N**on-**P**arametric **K**ernel **L**earning framework (`NPKL`) to deal with these problems. We generalized the graph embedding framework into kernel learning, by reforming it as a semi-definitive programming (`SDP`) problem, smoothing and avoiding over-smoothing the functional Hilbert space with Laplacian regularization. We proposed two algorithms to solve this problem. One is a straightforward algorithm using semidefinite programming (`SDP`) to solve the original kernel learning problem, dented as `TRA`nsductive `G`raph `E`mbedding `K`ernel learning (`TRAGEK`); the other is to relax the `SDP` problem and solve it with a constrained gradient descent algorithm. To accelerate the learning speed, we further divide the data into groups and used the `sub-kernels` of these groups to approximate the whole kernel matrix. This algorithm is denoted as **E**fficient **N**on-**PA**rametric **K**ernel **L**earning (`ENPAKL`). The advantages of the proposed `NPKL` framework are 1) supervised information in the form of pairwise constraints can be easily incorporated; 2) it is robust to the number of pairwise constraints, *i.e.*, the number of constraints does not affect the running time too much; 3) `ENPAKL` is efficient

Changyou Chen
Research School of Information Sciences and Engineering, The Australian National University, Canberra, Australia
E-mail: cchangyou@gmail.com
This work was done when I was at Fudan University, Shanghai, China

Junping Zhang
Shanghai Key Laboratory of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai, China
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: jpzhang@fudan.edu.cn

Xuefang He
School of Software and Information Engineering, Beihai College of Beihang University
E-mail: ahexuefang@gmail.com

Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: zhouzh@nju.edu.cn

to some extent compared to some related kernel learning algorithms since it is a constraint gradient descent based algorithm. Experiments for clustering based on the learned kernels show that the proposed framework scales well with the size of datasets and the number of pairwise constraints. Further experiments for image segmentation indicate the potential advantages of the proposed algorithms over the traditional k-means and N-cut clustering algorithms for image segmentation in term of segmentation accuracy.

Keywords Kernel learning · semi-definitive programming · graph embedding · pairwise constraint · semi-supervised learning

1 Introduction

Semi-supervised clustering based on kernel learning is a popular research topic in machine learning since one can incorporate the information of a limited number of labeled data or a set of pairwise constraints into the kernel learning framework [10]. The reason is that for clustering, the pairwise constraints provide useful information about which data pairs are in the same category and which ones are not. To learn such kinds of kernel matrices, Kulis *et al.* [18] proposed to construct a graph based kernel matrix which unifies the vector-based and graph-based semi-supervised clustering. A further refinement on learning kernel matrices for clustering was investigated by Li *et al.* [19]. In their approach, data are implicitly projected onto a feature space which is a unit hyperball, subjected to a collection of pairwise constraints. However, the above clustering algorithms via kernel matrices either can not scale well with the increasing number of pairwise constraints and the amount of data, or lacks theoretical guarantee for the positive semi-definite property of the kernel matrices. In another aspect, Yueng *et al.* [33] proposed an efficient kernel learning algorithm through low rank matrix approximation. However, in their algorithm, the form of kernel matrix is assumed to be linear combination of several base kernel matrices. Note that this might reduce the dimension of the hypothesis kernel space, we call such kinds of algorithms parametric kernel learning. In addition, Cortes *et al.* [34] proposed a kernel learning algorithm by taking the non-linear combinations of kernels, which is a generalization of the linear combination case but still lies in the framework of parametric kernel learning. Addressing these two limitations is the major purpose of this paper.

On the other hand, we note that many algorithms based on the graph embedding framework often achieve an enhanced discriminant ability by utilizing the marginal information, *e.g.*, making the dissimilarity data points near the margin as far as possible and meanwhile compacting the points in the same class [30,31]. It is therefore worthwhile to generalize the graph embedding framework into kernel learning [1].

Based on the aforementioned goals, in this paper we propose a new scalable kernel learning framework `NPKL` (**N**on-**P**arametric **K**ernel **L**earning with robust pairwise constraints), and apply it for semi-supervised clustering. First, we generalize the graph embedding framework on a feature space which is assumed to be a possibly infinite subspace of the l^2 Hilbert space with unit norm, which is similar to [19]. Then the unknown feature projection function ϕ is implicitly learned by transforming the criterion of the graph embedding (*i.e.*, maximizing the sum of distances

[1] Note that although we want to learn a kernel matrix from the aspect of graph embedding, it has little relationship with some algorithms using graph embedding framework such as marginal factor analysis (`MFA`) [30]. The reason is that such kinds of algorithms aim at supervised learning for classification, thus there is no need to compare the proposed algorithm with them.

of between-class data pairs while minimizing that of within-class data pairs) into an SDP problem. To get smoother solution of the predictive function, smoothing technique using some kind of Laplacian regularizer is introduced. By this, ideally, data from the same class would be projected into the same location in the feature space. Meanwhile, the distances between the locations of different classes should be as large as possible, as illustrated in Figure 1. We propose two algorithms to solve this problem. One is to optimize the objective function directly by solving an SDP problem, which we call TRAGEK (**TRA**nsductive **G**raph **E**mbedding **K**ernel learning), since the SDP problem is derived from a transductive graph embedding formulation. In TRAGEK, it is not necessary to explicitly specify which pair of data points should lie close, and the running time is much less sensitive to the number of pairwise constraints than Li *et al.*'s work [19]. However, the SDP problem in TRAGEK limits the application of the proposed algorithm to large scale datasets. To alleviate this problem, we propose to solve the SDP problem via a constrained gradient descent algorithm that iteratively projects the unconstrained solutions to the cone formed by the constraints. Furthermore, we divide the whole dataset into groups of sub-data sets, and the corresponding sub-kernels are learned for these sub-data separately. Finally, the global kernel matrix is obtained through the combination of these sub-kernels. In this way, not only is the positive semi-definite property of the kernel matrix well preserved, but also the computational complexity scales at most linearly with the size of the dataset, which is very efficient. We call this algorithm ENPAKL (**E**fficient **N**on-**PA**rametric **K**ernel **L**earning).

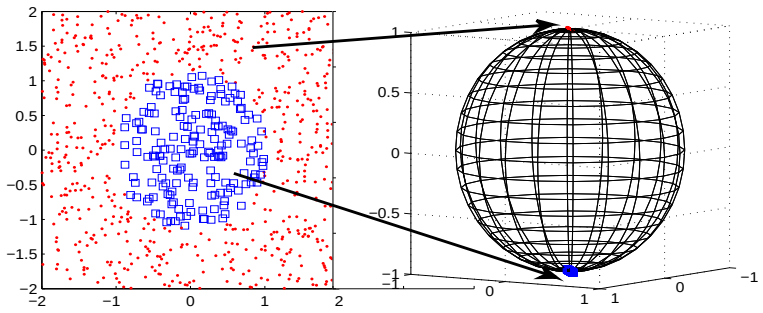


Fig. 1 With a feature projection function ϕ, data from two classes are projected from 2-dimensional data space into 3-dimensional feature space in which each point corresponds to one class.

The remaining of this paper is organized as follows. Section 2 reviews some related work for clustering using kernel learning. Section 3 formulates our problem and presents the TRAGEK algorithm. Section 4 elaborates the efficient algorithm ENPAKL for our problem. And experiment results are shown in Section 5. Finally, Section 6 concludes the paper.

2 Clustering and Kernel Learning

Learning with kernels [22] is a popular research topic in machine learning. In this section, however, rather than reviewing the theoretical aspects of kernel learning algorithms such as [5,6], and the online kernel learning such as [7], we put our emphasize on the applications of kernel learning algorithms. More specifically, we discuss the work of

kernel learning for semi-supervised clustering. To a certain extent, kernel learning for clustering can be viewed as metric learning, because the kernel matrix can be regarded as some specific distance between data points equipped with a specific metric.

For traditional clustering algorithms, the most frequently used ones include k-means and fuzzy k-means [2]. Recently, Yang *et al.* [32] proposed an improved fuzzy k-means algorithm to assign a value of 1 to data pairs with a defined cluster score. Moreover, Trappey *et al.* [24] presented a fuzzy ontology schema for hierarchically clustering of documents, which can solve the inconsistent and ineffective problem encountered by the traditional keyword-based methods. For the traditional k-means clustering algorithm, Xiong *et al.* [28] proposed to use the coefficient of variation (CV) as some criterion to analysis the performance of k-means algorithm under skewed data distribution.

To utilize label information to enhance the cluster performance, semi-supervised clustering approaches were proposed, which can be roughly categorized into constraint- and metric-based ones. The former utilizes either labeled data or pairwise constraints to improve the performance of clustering [4, 8], while the latter learns a more rational metric to fit the constraints by utilizing the provided label information [17, 27], thus the semi-supervised kernel learning is closely related to the clustering algorithms.

More specifically, Wagstaff *et al.* [25] modified the k-means algorithm by considering pairwise similarities and dissimilarities, known as the constrained k-means. To boost the performance of constrained k-means, Hong *et al.* [16] refined the assignment order of this algorithm by ranking all instances in the dataset according to their clustering uncertainty. Based on the Hidden Markov Random Field (`HMRF`), Basu *et al.* [4] used the provided pairwise constraints in the objective function for semi-supervised clustering, while Lu *et al.* [20] proposed a clustering algorithm with the must-link and cannot-link constraints using the Gaussian mixture model (`GMM`). Xing *et al.* [27] learned a distance matrix for clustering by explicitly minimizing the distances between similar samples and maximizing those between dissimilar ones. Bar-Hillel *et al.* [3] proposed a simpler but more efficient algorithm using the relevant component analysis (`RCA`). Furthermore, they proposed another metric learning algorithm [14] that learns a non-parametric distance function, but without the guarantee that the function is actually a metric.

Although kernel methods have been studied for decades, not much work has focused on learning a non-parametric kernel using only the training data. In contrary, most of the work focuses on learning the kernel from some predefined base kernels [29]. Recently, kernel learning for clustering has attracted more and more attentions because one can easily incorporate some useful information into the kernel learning framework [11, 18]. Earlier kernel learning algorithms mainly focus on linear or non-linear combination of some base kernels [9]. For example, Yeung *et al.* [33] proposed a scalable kernel learning algorithm in which some low rank kernel matrices obtained by the eigenvectors of the initial kernel matrix are used as base kernels, and a collection of optimal weights of the base kernels need to be learned. Cortes [34] proposed to learn the kernel matrix by defining some non-linear terms of the basis kernels and use a projection-based gradient descent algorithm to learn the weights. One disadvantage of this algorithm is that only the must-link constraint information is incorporated. To employ both must-link and cannot-link constraint information, Hoi *et al.* [15] proposed a kernel learning algorithm by formulating it into a semi-definite programming (`SDP`) problem. To the best of our knowledge, this is the first non-parametric kernel learning algorithm that does not need to explicitly

take the base kernels into consideration. Latter, an efficient algorithm for solving the SDP problem mentioned above is proposed by Zhuang *et al.* [26] by introducing an extra low rank constraint on the objective function. Furthermore, assuming that the feature space is a unit hyperball in which must-link data pairs are constrained to be one point and cannot-link pairs should be orthogonal with each other, Li *et al.* [19] proposed another SDP based algorithm for kernel learning. Note that one problem for the SDP related algorithms introduced above is the computational complexity, how to avoid this complexity is one of the major concerts of this paper.

3 TRAGEK: Transductive Non-Parametric Kernel Learning

Our non-parametric kernel learning problem assumes that the feature space of the data is a subspace of the l^2 Hilbert space with unit norm. More specifically, given a data point x, there exists a projection ϕ from the data space to the feature space endowed with a unit norm, *i.e.*, $\|\phi(x)\| = 1$. In this way, data are mapped to the surface of a unit hyperball in which data of different classes can be separated easily. To find such a mapping function ϕ, we tried to generalize the graph embedding [30] into the kernel space and transformed the problem into the semi-definite programming with pairwise constraints. We further introduced some regularization terms such as the Laplacian regularizer to smooth the prediction function. Finally, the kernel k-means clustering algorithm is performed in the learned kernel matrix for clustering.

3.1 Transductive Kernel Learning with Graph Embedding

One goal of the graph embedding is to learn a low-dimensional discriminant subspace in which the sum of within-class distances is minimized meanwhile that of between-class distances is maximized [30]. This idea is formulated in Eq. (1):

$$F_g = \min_{Y} \frac{tr\{YLY^T\}}{tr\{YL^pY^T\}}, \tag{1}$$

where $Y = (y_1, y_2, \cdots, y_M), y_i \in \Re^m$ is the data representation in the feature space, M is the number of samples. $L = D - W$ and $L^p = D^p - W^p$ are two graph Laplacian matrices with $D(i,i) = \sum_j W(i,j)$ and $D^p(i,i) = \sum_j W^p(i,j)$ representing two diagonal matrices. Two similarity matrices W and W^p represent the within-class relationship and between-class relationship of data points, respectively, which are defined according to the pairwise constraints as:

$$W(i,j) = \begin{cases} 1, \text{ } i \text{ and } j \text{ belong to the same class} \\ 0, \text{ otherwise.} \end{cases}, W^p(i,j) = \begin{cases} 1, \text{ } i \text{ and } j \text{ belong to different classes} \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

For better generalization, the two quadratic forms in Eq. (1) are lifted into linear forms in the kernel space as:

$$F_1 = tr\{YLY^T\} = tr\{LY^TY\} =< l, K_v >, \tag{3}$$

where $l = vec(L)$ is the vectorization of the matrix L, and K_v[2] is the vectorization of the kernel matrix K defined by $K(i,j) =< y_i, y_j >$.

[2] We use such kind of convention without declaration below.

In contrast to the graph embedding, the goal of TRAGEK is to cluster data in the feature space defined on the unit norm subspace of the l^2 Hilbert space. It is easy to get the inner product of two vector in the feature space using the re-producing property of the re-producing Hilbert space:

$$K(\phi(x), \phi(y)) = k_{\phi(y)}(\phi(x)) =< \phi(x), \phi(y) >, \tag{4}$$

where $k_{\phi(y)}(\cdot) \in \mathcal{H}$, x and y are two data points in the data space, ϕ is the mapping function to be learned. Furthermore, in our transductive inference, the matrices W and W^p are defined using the pairwise constraints as in Eq. (2). Based on this, the objective function of TRAGEK is to kernelize the criterion of graph embedding [30] as:

$$\begin{aligned} F_t = \min_K \frac{< l, K_v >}{< l^p, K_v >} \\ \text{s. t. } l^T K_v > 0, l^{pT} K_v > 0, \\ K(i,i) = 1, \text{for all } i, \\ K(i,j) \leqslant 1, \text{for all } i, j. \end{aligned} \tag{5}$$

The optimization function means that the learned kernel K should cluster within-class data as close as possible and between-class data as far as possible in the feature space. The first constraint is required according to the positive semi-definite property of the Laplacian matrix L, and the second and third constraints stem from the assumption that data in the feature space should lie on the surface of a hyperball with unit radius.

3.2 Conic Optimization Programming Relaxation

To transform this problem into a convex optimization problem which has a unique optimal solution and polynomial time computational complexity, we first prove Theorem 1 as follows.

Theorem 1 *The nonlinear optimization problem defined in Eq. (5) can be relaxed to a second order cone programming as:*

$$\begin{aligned} \min \ : \ \{t\} \\ s.t. \ t + < l^p, K_v > \geqslant \left\| \begin{matrix} < l, K_v > \\ t - < l^p, K_v > \end{matrix} \right\|, \\ l^T K_v > 0, l^{pT} K_v > 0, \\ K(i,i) = 1, for \ all \ i, \\ K(i,j) \leqslant 1, for \ all \ i, j. \end{aligned} \tag{6}$$

Proof we first relax the F_t in Eq. (5) as:

$$F_t^{'} = \min_f \frac{(< l, K_v >)^2}{4 \cdot < l^p, K_v >}. \tag{7}$$

This does not bring a significant reduction to the optimization formula since both $< l, K_v >$ and $(< l, K_v >)^2$ are monotonously increasing when $l^T K_v > 0$. Now let's introduce an extra variable t such that $t \geqslant \frac{(< l, K_v >)^2}{4 \cdot < l^p, K_v >}$. As a

consequence, the optimization equation in Eq. (7) is equivalent to:

$$\begin{aligned} &\min\ :\ \{t\} \\ &\text{s. t. } t \geqslant \frac{(<l, K_v>)^2}{4\cdot <l^p, K_v>}. \end{aligned} \tag{8}$$

Furthermore, the constraint in Eq. (8) can be further transformed as:

$$\begin{aligned} 4t \cdot (<l^p, K_v>) &\geqslant (<l, K_v>)^2 \\ \Leftrightarrow (t+ <l^p, K_v>)^2 &\geqslant (<l, K_v>)^2 + (t- <l^p, K_v>)^2, \end{aligned}$$

which is a second order cone constraint:

$$t+ <l^p, K_v> \geqslant \left\| \begin{array}{c} <l, K_v> \\ t- <l^p, K_v> \end{array} \right\|. \tag{9}$$

Combining Eqs. (7), (8) and (9) we get the conclusion in Theorem 1.

To further simplify Eq. (5), we here introduce Theorem 2.

Theorem 2 *If K is a positive semi-definite matrix of size $n \times n$, then*

$$K(i,j) \leqslant \max\{K(k,k)\}, 1 \leqslant i,j,k \leqslant n. \tag{10}$$

Proof Since matrix K is positive semi-definite, we can decompose K as:

$$K = (k_1, k_2, \cdots, k_n)^T (k_1, k_2, \cdots, k_n), \tag{11}$$

where k_i is a vector of arbitrary dimension. So we have:

$$\begin{aligned} K(i,j) &= <k_i, k_j> \\ &\leqslant \sqrt{<k_i, k_i> \cdot <k_j, k_j>} \\ &\leqslant \max\{K(i,i), K(j,j)\} \\ &\leqslant \max\{K(k,k)\}, 1 \leqslant k \leqslant n. \end{aligned} \tag{12}$$

By Theorem 2, we can introduce an extra positive semi-definite constraint to replace the last n^2 inequality constraints in Eq. (6), resulting in:

$$\begin{aligned} &\min\ :\ \{t\} \\ &\text{s.t. } t+ <l^p, K_v> \geqslant \left\| \begin{array}{c} <l, K_v> \\ t- <l^p, K_v> \end{array} \right\|, \\ &K \succcurlyeq 0, \\ &K(i,i) = 1, \text{for all } i, \\ &l^T K_v > 0, l^{pT} K_v > 0. \end{aligned} \tag{13}$$

3.3 Smoothness Controlling

While the number of constraints has been greatly reduced, it is necessary to refine Eq. (13) since the current prediction function is not smooth enough in the Hilbert space. Note that if all the elements of K in Eq. (13) were the same, it would lead to $< l, K >= 0$ and $< l^p, K >= 0$. Consequently, the optimization problem would be infeasible since there were no interior points in the second order cone. We regard this case as over-smoothness. Therefore, we propose two strategies to smooth the predicted function and avoid over-smoothness.

Note that the "smoothness" of the manifold, which is measured by the `Laplace-Beltrami` operator on Riemannian manifolds, can be substituted by a discrete analogue operator defined as the graph Laplacian on the graph [35]. Therefore, we can employ Laplacian regularizer, denoted as S, to smooth the prediction function in its Hilbert space. Similar to Li *et al.*'s work [19], we here introduce a refined regularizer by incorporating a global normalized graph Laplacian[3] into our optimization framework, which is defined as:

$$\begin{aligned} S &= \sum_{i,j=1}^{n} W^{'}(i,j) \| \frac{\phi(x_i)}{\sqrt{D'(i,i)}} - \frac{\phi(x_j)}{\sqrt{D'(j,j)}} \|^2 \\ &= tr\left\{ (I - (D^{'})^{-\frac{1}{2}} W^{'} (D^{'})^{-\frac{1}{2}}) K \right\} \\ &= < \bar{l}, K_v >, \end{aligned} \tag{14}$$

where K, K_v are defined as the same as the previous ones, I is the identity matrix, $W^{'}$ is defined as:

$$W^{'}(i,j) = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}, & i \neq j \\ 0, & i = j, \end{cases} \tag{15}$$

where σ is a scale factor, $(I - (D^{'})^{-\frac{1}{2}} W^{'} (D^{'})^{-\frac{1}{2}})$ is a normalized Laplacian matrix corresponding to $W^{'}$, and $\bar{l} = vec(I - (D^{'})^{-\frac{1}{2}} W^{'} (D^{'})^{-\frac{1}{2}})$ is the vectorization of the Laplacian matrix. Obviously, this formulation enforces smoothness over all the data globally. Finally, adding this term as an regularizer of the optimization problem in Eq. (13) results in the following optimization problem:

$$\begin{aligned} \min \ : \ & \{t + \lambda S\} \\ \text{s.t. } t + < l^p, K_v > & \geqslant \left\| \begin{array}{c} < l, K_v > \\ t - < l^p, K_v > \end{array} \right\|, \\ & K \succcurlyeq 0, \\ & K(i,i) = 1, \text{for all } i \\ & < l, K_v > > 0, < l^p, K_v >> 0, \end{aligned} \tag{16}$$

where λ is a parameter controlling the degree of smoothness on the predicted function.

As we stated before, over-smoothness is likely to happen. In addition, dropping the last two constraints $l^T K_v > 0, l^{pT} K_v > 0$ is prone to numerical problem[4]. We solve this by adding two extra terms into the optimization framework.

[3] More satisfactory results might be attained if employing more sophisticated regularizers.

[4] Note that for two positive semi-definite matrices A and B, $tr\{AB\} \geqslant 0$ holds, but not always > 0. We thus can not drop the last two constraints directly. Otherwise it is easy to run into numerical problem. Because the constraint still holds if $< l^p, K_v >$ is equal to a small enough positive constant, but this is far from the goal that $< l^p, K_v >$ should be as large as possible.

Algorithm 1 The TRAGEK Algorithm

1: **Input:** data $X = (x_1, x_2, \cdots, x_M)$, must-link constrains $\mathcal{M}$, cannot-link constrains $\mathcal{C}$, cluster number k.
2: **Output:** k clusters.
3: Construct the two Laplacian matrices in Eq. (5) as in graph embedding framework [30] using the provided must-link constraints $\mathcal{M}$ and cannot-link constraints $\mathcal{C}$, respectively.
4: Construct the global smoothness Laplacian matrix defined in Eq. (14).
5: Solve the conic optimization programming defined in Eq. (19).
6: Run the kernel k-means algorithm on the learnt kernel matrix K to form k clusters.

Remember that our goal is to make within-class samples as close as possible, and between-class samples as far as possible, this can be formulated in the following two formulas:

$$f_1 = \min tr\{YLY^T\} \equiv \min\{- < w, K_v >\}, \tag{17}$$

$$f_2 = \max tr\{YL^pY^T\} \equiv \min\{< w^p, K_v >\}, \tag{18}$$

where w, w^p are the vectorization forms of the two similarity matrices W and W^p defined in Eq. (2). Denote these two regularizers as $S_1 = - < w, K_v >$ and $S_2 =< w^p, K_v >$, we can incorporate them into the objective function to get the final optimization formula as.

$$\begin{aligned} \min \ : \ & \{t + \lambda S + \lambda_1 S_1 + \lambda_2 S_2\} \\ \text{s.t. } t+ < l^p, K_v > \geqslant & \left\| \begin{array}{c} < l, K_v > \\ t- < l^p, K_v > \end{array} \right\|, \\ K \succcurlyeq & \ 0, \\ K(i,i) = 1, & \text{for all } i, \end{aligned} \tag{19}$$

where the two parameters λ_1 and λ_2 control the weights of the two graphs. As the two regularization terms are introduced, the two terms $< l, K_v >$ and $< l^p, K_v >$ will be enforced larger than zero. Therefore, we can drop the last two constraints in Eq. (16) in practice without causing any problem. Furthermore, a remarkable advantage of Eq. (19) is that it is a conic optimization programming (also SDP problem) which can be solved using the popular conic optimization software such as SeDuMi, which is of polynomial time complexity and has a theoretically proven $O(\sqrt{n}\log(\frac{1}{\epsilon}))$ worst-case iteration bound [23]. TRAGEK is illustrated in Algorithm 1.

4 ENPAKL: An Efficient Kernel Learning Algorithm

Although TRAGEK is a convex optimization problem which means that there exists a global optimal solution, the computational cost is often relatively high and it often results in unstable solutions for large datasets, furthermore, it is sensitive to the parameter settings such as the chooses of $\lambda, \lambda_1, \lambda_2$, etc. In this section, we propose to resolve these problems by two strategies. Firstly, we propose a constrained gradient descent based algorithm to make the learning procedure much stable and efficient. Secondly, we reduce the large-scale kernel learning problem into sub-kernel learning and combine these sub-kernels to approximate the global kernel matrix, this trick makes the computational

complexity rely linearly on the number of data points. Experimental results in Section 5 show that the proposed strategies can approximate the true kernels well.

4.1 Constrained Gradient Descent

Note that the original optimization problem of Eq. (19) is not efficient enough, by taking the advantage of the iterative projection algorithm [27], we propose a constrained gradient descent based algorithm for training. The algorithm iteratively projects the solution obtained by gradient descent to the cones formed by the constraints.

Specifically, we want to avoid the `SDP` formulation above to reduce the computational complexity, thus we reformulate the original kernel learning problem of Eq. (19) in the following form:

$$\begin{aligned} F^{'} &= \min_{K} \frac{< l, K_v >}{< l^p, K_v >} \\ \text{s. t.} \quad & K \succcurlyeq 0, \\ & K(i,i) = 1, \text{for all } i. \end{aligned} \tag{20}$$

Taking the logarithm of $F^{'}$, and using the Laplacian smoother S in Eq. (19) as a regularization term, the objective function of `ENPAKL` is:

$$\begin{aligned} F = \min\nolimits_K \{ & \log(< l, K_v >) - \log(< l^p, K_v >) + \lambda \log(< \bar{l}, K_v >) \} \\ \text{s. t.} \quad & K \succcurlyeq 0, \\ & K(i,i) = 1, \text{for all } i. \end{aligned} \tag{21}$$

It is straightforward to derive the gradient of F in Eq. (21):

$$\frac{\partial}{\partial K_v} F = \frac{l}{< l, K_v >} - \frac{l^p}{< l^p, K_v >} + \lambda \frac{\bar{l}}{< \bar{l}, K_v >}. \tag{22}$$

Thus, we can update the kernel matrix K_v by constrained gradient descent as:

$$\begin{aligned} K_v^t &= K_v^{t-1} - \omega \cdot \frac{\partial}{\partial K_v^{t-1}} F \\ \text{s. t.} \quad & K^t \succcurlyeq 0, \\ & K^t(i,i) = 1, \text{for all } i, \end{aligned} \tag{23}$$

where ω is the step size for the current update, and t is the iteration index. Actually, we can regard Eq. (23) as the optimization problem on manifolds [36], however, instead of solving this problem directly on manifolds, we use the projection method by iteratively projecting the updated values into the cones formed by the constraints until converged. The constrained gradient descent algorithm is described in Algorithm 2.

Note that in Algorithm 2, the solution of Eq. (24) and Eq. (25) can be calculated based on the following theorems:

Theorem 3 *The solution to Eq. (24) is to set all the negative eigenvalues of K_v to 0, that is, $K = V \max(D, 0) V^T$, where V, D are eigenvectors and eigenvalues of K_v.*

Algorithm 2 Constraint Gradient Descent Algorithm

1: **Input:** Initial kernel matrix K_v^0, learning rate ω.
2: **Output:** Kernel matrix K.
3: Calculate for the first update: $K_v^1 = K_v^0 - \omega \cdot \frac{\partial}{\partial K_v^0} F$.
4: Set $t = 1$.
5: **while** Not Converged **do**
6: **while** Not Found rational $K^{''}$ satisfying the constraints **do**
7: Solve:

$$K^{'} = \arg\min_K \|K - K^t\|^2$$
$$\text{s. t. } K \succcurlyeq 0, \tag{24}$$
$$K^{''} = \arg\min_K \|K - K^{'}\|^2,$$
$$\text{s. t. } K(i,i) = 1, \text{for all } i. \tag{25}$$

8: **end while**
9: $t = t + 1$.
10: $K_v^t = K_v^{''} - \omega \cdot \frac{\partial}{\partial K_v^{t-1}} F$.
11: **end while**

Theorem 4 *The solution to Eq. (25) is equal to $K^{'}$, except that all the diagonal elements of $K^{'}$ are set to 1.*

Similar problem and proof for Theorem 3 can be found in [13]. Here we prove Theorem 4.

Proof (Proof of Theorem 4) Unfolding the norm and omitting the terms independent of $K^{'}$, we can rewrite Eq. (25) as:

$$K^{''} = \arg\min_K tr\left\{KK^T - 2KK^{'T}\right\}$$
$$\text{s. t.} \quad tr\{KE_i\} = 1. \tag{26}$$

where E_i is a matrix of the same size with K, and with all elements being 0 except the i-th element of the diagonal being 1. Then the corresponding Lagrangian function with the corresponding Lagrangian multipliers λ_i's is:

$$g(K,\lambda) = tr\left\{KK^T - 2KK^{'T} - \sum_i \lambda_i \left(tr\{KE_i\} - 1\right)\right\}. \tag{27}$$

Taking the derivative of $g(K,\lambda)$ with respect to K, we have:

$$\begin{aligned}\frac{\partial}{\partial K} g(K,\lambda) &= 2K - 2K^{'} - \sum_i \lambda_i E_i \\ &= 2K - 2K^{'} - diag(\lambda).\end{aligned} \tag{28}$$

Setting Eq. (28) to zero leads to:

$$K = K^{'} + \frac{1}{2} diag(\lambda). \tag{29}$$

Remember the constraint is $diag(K) = 1$, where 1 is a vector with all elements being 1, and note the form of the solution of K in Eq. (28), we can get the solution K by setting the diagonal elements of $K^{'}$ to 1. This completes the proof.

4.2 Learning the Global Kernel from `Sub-kernels`

In this section, the second strategy to improve the efficiency of the proposed algorithm is presented. First note that it is not scalable and efficient enough for large datasets using Algorithm 2 directly, since we need to perform an eigen-decomposition for the current kernel matrix to solve Eq. (24), which is time consuming when the number of data points is large. Therefore, we propose to approximate the global kernel matrix using local kernel matrices (or `sub-kernel` matrices) formed by a subset of data points.

Suppose we start with a small subset of data (namely, m data points) denoted as $D = \{x_1, x_2, \cdots, x_m\}$, and the corresponding `sub-kernel` matrix K_D has been learned using the constrained gradient descent algorithm described in Algorithm 2. The idea is to approximate the other elements of the global kernel matrix using this sub-kernel matrix. Note that because data of the same class in the feature space $\mathcal{H}$ is assume to be flat (they are clustered into one point ideally in the feature space), it is reasonable to approximate all other data points $\phi(x_i)$ using the linear combination of this subset of data $\phi(D)$, that is: $\phi(x_i) = \sum_j w_{ij}\phi(x_j)$, where w_{ij} are the weights to be learned. There are two situations for $x_i \notin D$:

1). If x_i has at least one link constraint with some points x_j in D, according to our assumption, this means in the feature space, $\phi(x_i) = \phi(x_j), x_j \in D$. Taking all such points into consideration, we relax $\phi(x_i)$ to be the linear combination of other points in the feature space, then we get $\phi(x_i) = \sum_j \xi\phi(x_j)$, where $w_{ij} = \xi$ is equal for all x_j.

2). If x_i has no link constraints with the points in D, then we approximate $\phi(x_i)$ using the weighted combination of $\phi(D)$ in the feature space. We assume these weights should be approximately the same with those learned by minimizing the reconstruction error in the original data space. This makes our approximation different from the one proposed by Yueng *et al.* [33]. While the objective function for w_{ij} is similar to local linear embedding (LLE) [21], the definition of neighborhood is different. The objective function for w_{ij} is:

$$E = \min_{w_{ij}} \sum_i \|x_i - \sum_{j \in \mathcal{N}(i)} w_{ij} x_j\|^2, \tag{30}$$

where $\mathcal{N}(i)$ is defined to be the k nearest data points of x_i except for those having cannot link constraints with x_i.

To sum up, the weights w_{ij}'s are defined as:

$$w_{ij} = \begin{cases} 1 & x_i, x_j \in D \text{ and } i = j, \\ \frac{1}{T} & x_i \notin D, x_j \in D \text{ has a linked constraint}, \\ \text{Eq. (30)} & x_i \notin D, x_j \in D \text{ is } x_i\text{'s neighboring point, ,} \\ & \text{but has no cannot linked constraint}, \\ 0 & \text{otherwise}, \end{cases}$$

where T is the number of link constraints for $x_i \notin D$ and $x_j \notin D$. Thus, the whole dataset in the feature space can now be written in the matrix form as:

$$\phi(X) = \phi(D)W^T, \tag{31}$$

Algorithm 3 Sub-data sets picking schema

1: Sort the data points by the degrees of themselves.

2: Choose the first B data points that have the largest degrees as `basic landmarks`.

3: For the rest of the data points with degrees larger than 0, each time choose R data points in descending order of degrees, then combine them with the `basic landmarks` to get one `sub-data set`, loop until all data points have been chosen.

where X is the whole dataset, $W = (w_1^T, w_2^T, \cdots, w_n^T)^T, w_i = (w_{i1}, w_{i2}, \cdots, w_{iM})^T$. Then, the whole kernel matrix can be approximated using Eq. (31) as[5]:

$$K_X = \phi(X)^T \phi(X) = W K_D W^T. \tag{32}$$

From Eq. (32) it can be seen that the kernel matrix K_X of the whole dataset can be approximated by a `sub-kernel` matrix K_D. However, given arbitrary must-link and cannot-link constraints, only one `sub-kernel` matrix might not approximate the whole kernel matrix well because all the pairwise constraints might not be included in one sub dataset. To solve this problem, we propose a `sub-data set picking schema` that scales at most linearly with the size of the dataset to partition the whole dataset into several sub-data sets, then we use the corresponding sub-kernels to approximate the whole kernel. It can be proved that the computational complexity for this strategy is at most $O(n)$ times larger than that of using only one sub-kernel.

In this schema, we use the number of constraints (`degrees` of nodes in the graph) in the sub-data as the measure of the prior information this sub-data contains. The larger `degree` of one data point, the more prior information it has, and thus the higher probability the data point should be used to learn the sub-kernel. This `sub-data` set picking schema is described in Algorithm 3.

We can see from Algorithm 3 that the number of `sub-data sets` scales at most linearly with the number of the whole data points, thus is very efficient. Suppose at last we divide the whole dataset into L `sub-data sets`, and for each of such `sub-data set` a `sub-kernel` is learned by some kernel learning algorithms such as Algorithm 2, also we denote $K_1, K_2, \cdots, K_L$ as the approximated kernel matrices calculating using Eq. (32), then the final kernel matrix for the whole dataset is set to be:

$$K = \sum_i \alpha_i K_i, \text{ s t. } \sum_i \alpha_i = 1, \tag{33}$$

where α_i is the weight for the i-th kernel. In the experiments, we set α_i proportional to the total degree of their data points.

Note that this algorithm is efficient because it solves the original `SDP` problem of `TRAGEK` using a constraint gradient descend based algorithm, we will compare these two algorithms with respect to their efficiency and accuracy for clustering in the experiments.

[5] There are two points to be declared here. One is that it is easy to prove that K_X in Eq. (32) is a positive semi-definite matrix if K_D is positive semi-definite, this property makes K_X of the whole dataset still be a kernel matrix, which does not violate our objective. The second point is that for unknown points, in order to constrain the feature space be a hyperball, we need to normalize the weights calculated in Eq. (30) by dividing the weights by a normalized scaler $w_i^T K_D w_i$, that is, $w_i = \frac{w_i}{w_i^T K_D w_i}$.

Table 1 Twelve datasets used in our experiments. The first two databases are artificial datasets, the rest ones are from the UCI machine learning repository.

Data set	#Classes	Dimension	#Samples
Chessboard	2	2	100
Double-Spiral	2	3	100
Glass	7	9	214
Heart	2	13	270
Iris	3	4	150
Protein	6	20	116
Sonar	2	60	208
Soybean	4	35	47
Wine	3	13	178
Wisconsin	2	30	720
Digital04	5	6	1000
Waveform	3	21	1800

5 Experiments

To test the proposed kernel learning algorithms TRAGEK and ENPAKL, we employed them for clustering and also used ENPAKL for image segmentation. We carried out the evaluations on two simulated datasets and ten datasets from the UCI machine learning repository [1]. The details of these datasets are tabulated in Table 1, where the first nine datasets have been often used in evaluating the performance of semi-supervised clustering algorithms [15, 19]. We compared ENPAKL with the PCP algorithm [19] and the SSKK algorithm [18] as well as the traditional k-means algorithm. We also investigated the influence of the number of pairwise constraints to the clustering performance for ENPAKL. To measure the clustering performance, we adopted the metric defined in [15]:

$$acc = \sum_{i>j} \frac{2 \cdot \mathcal{I}(\mathcal{I}(c_i, c_j), \mathcal{I}(\hat{c}_i, \hat{c}_j))}{n(n-1)}, \tag{34}$$

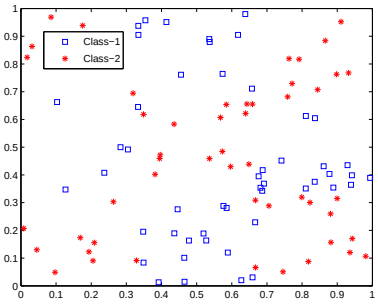
where $\mathcal{I}(a, b)$ is an indicator function returning 1 if $a = b$, and 0 otherwise, c denotes the true cluster membership and $\hat{c}$ denotes predicted cluster membership, and n is the number of samples. Without loss of generality, moreover, we set parameters λ, λ_1 and λ_2 in Eq. (19) to 1, and the scale factor σ in Eq. (15) to the average pairwise distance of the data set.

Other than this, we also applied the proposed ENPAKL together with the k-means and N-cut algorithms [41] on the MSRC Object Category Image Database(v2) [39] for image segmentation. Details are described in Section 5.5.

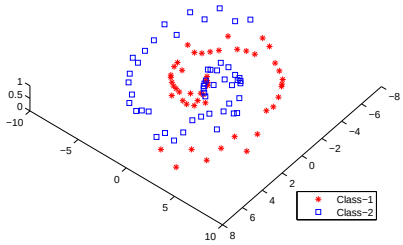
5.1 An Illustrative Experiment

To show that the proposed algorithms can propagate label information through the datasets, so that data in different classes can be separated as far as possible and those in the same class are clustered as close as possible, we run TRAGEK on the two synthetic datasets as used in [15], i.e., the Chessboard and Double-Spiral datasets in Figure 2. For better

illustration, we rearranged the order of the data such that the first part of the data matrix belongs to one class and the last part to the other. It can be seen in Figure 2 that the two classes are well separated. More specifically, we observed that the elements in the learned kernel K corresponding to the same class tend to 1 (black), while those corresponding to different classes tend to -1 (white). This means that the data from different classes are projected onto the opposite points on the hyperball of the feature space.



(a) Chessboard



(b) Double-Spiral

(c) Chessboard

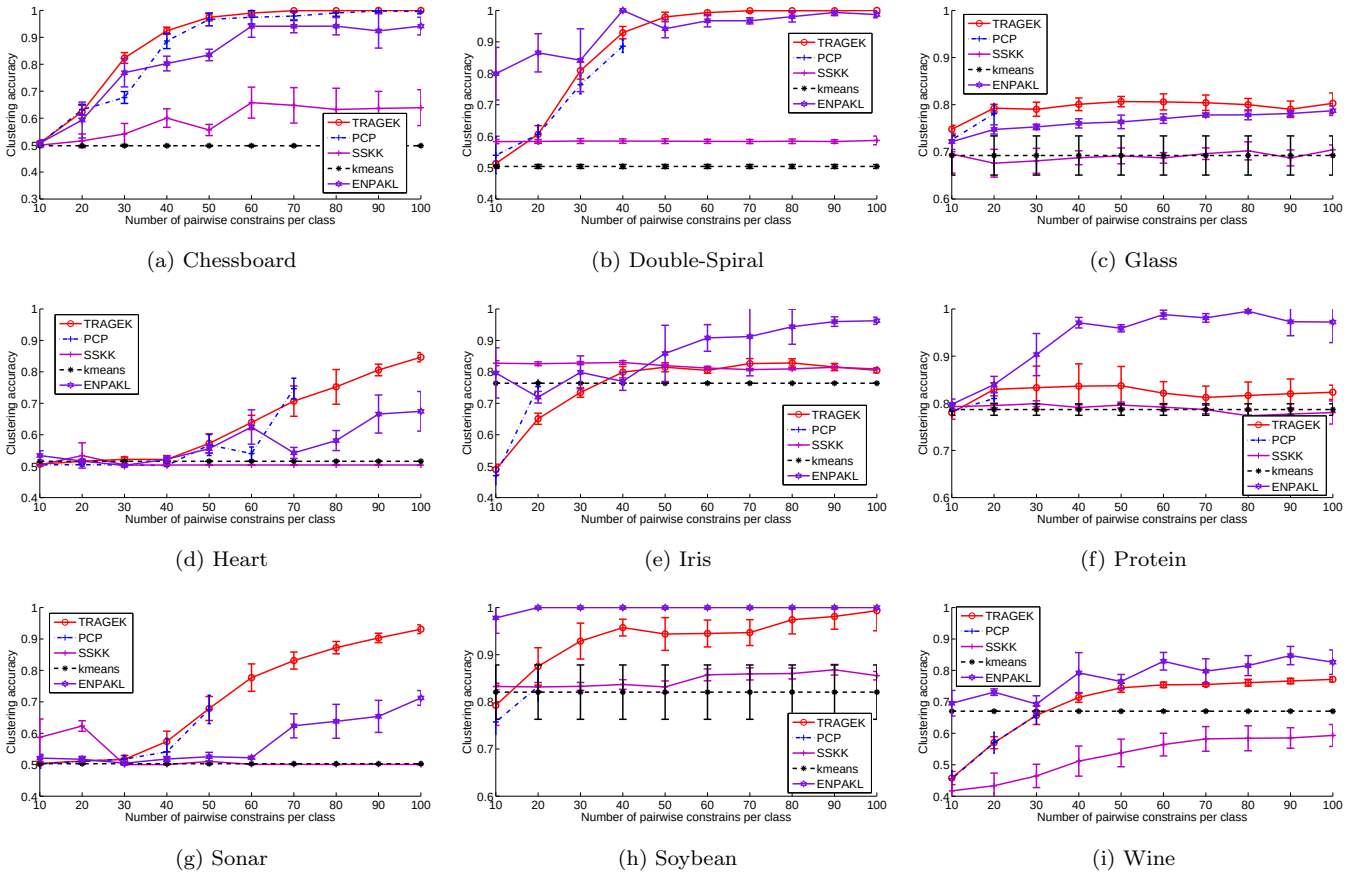
(d) Double-Spiral

Fig. 2 Clustering results on Chessboard and Double-Spiral datasets. In (c) and (d), black color means the corresponding values in the kernel matrix is 1, and white color means -1.

5.2 On Small Datasets

To compare the proposed kernel learning algorithms with some related algorithms for clustering, we tested them on nine small-scale data sets in Table 1 ranging from Glass to Wine. For `ENPAKL`, we set $B = 20, R = 10$ in Algorithm 3. Note that pairwise constraints are required for `TRAGEK`, `ENPAKL`, `PCP`, and `SSKK`, so we randomly generated k must-link constraints in each class and k cannot-link constraints between each two classes, where k ranges from 10 to 100 with an interval of 10. We thus have a total of $\frac{c(c+3c)k}{2}$ pairwise constraints for each experiment with a dataset of c classes. For each k, we randomly generated 20 different pairwise constrains, resulting in 20 different realizations of the pairwise constraints. The reported results were the average of the 20 different realizations together with 10 repetitions in the kernel k-means clustering step in Algorithm 1 for each pairwise constraint realization. The results are illustrated in Figures 3.

We can observe from Figure 3 that:



(a) Chessboard (b) Double-Spiral (c) Glass

(d) Heart (e) Iris (f) Protein

(g) Sonar (h) Soybean (i) Wine

Fig. 3 Clustering performance on nine small UCI datasets.

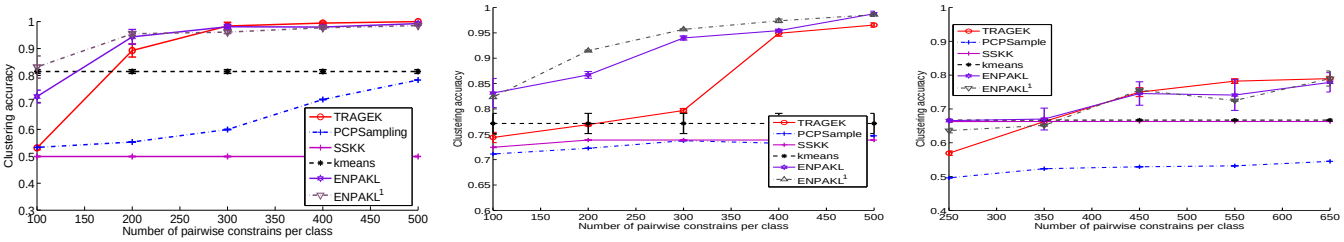
– `TRAGEK` outperforms the other three algorithms in all datasets except for the `Iris` and and `Wine` datasets when the number of pairwise constraints is less than 30.
– `PCP` is worse than `TRAGEK` in clustering accuracy in most cases and it runs into numerical problems when the number of pairwise constraints is large or when some noisy constraints (constraints that are wrongly labeled) are added. Furthermore, we observed in the experiments that the running time of `TRAGEK` fluctuated little when varying the number of pairwise constraints, which can be seen in Section 5.4.
– `ENPAKL` approximates amazingly well to the original kernel learning problem `TRAGEK`, sometimes even gets better performance. Another merit of `ENPAKL` is that it is much faster than `TRAGEK` and `PCP`. We will give some examples below.

5.3 On Larger Datasets

Note that the datasets used in Section 5.2 are small, though often used in evaluating semi-supervised clustering algorithms [15,19]. `TRAGEK` and other algorithms such as `PCP` can not scale well with large datasets and robust pairwise constraints. To evaluate the scalability of the proposed `ENPAKL` algorithm, we performed the experiments on three larger datasets described in the last three rows of Table 1. The `optdigits` dataset is a subset of a large digital dataset,

and it contains the digits from 0 to 4 with each class containing 200 instances (`Digital04`); `Waveform` also comes from a large dataset and it has 1800 instances.

The intrinsic disadvantage of `PCP` prevents it from being applied on such kind of large data with robust constraints. In order to enable it to work, we reduced the number of constraints by sampling. Specifically, the number of sampled constraints was set to the final number of constraints after the reduction in `TRAGEK`. We repeated the experiments for ten times with random sampling for the `PCP` algorithm, and picked up the best result and reported as the performance of `PCP`, denoted as `PCPSample`. In this experiment, we varied the number of pairwise constraints relative to the number of total data samples, and the parameters in Algorithm 3 were set to $B = 100, R = 100$. Also `ENPAKL`^{1} is a variant of `ENPAKL` by replacing the base `sub-kernel` learning algorithm in Algorithm 2 with `PCP`. The results for these algorithms are shown in Figure 4.



(j) Wisconsin (k) Digital04 (l) Waveform

Fig. 4 Clustering performance on three large UCI datasets.

It is found in the experiments that:

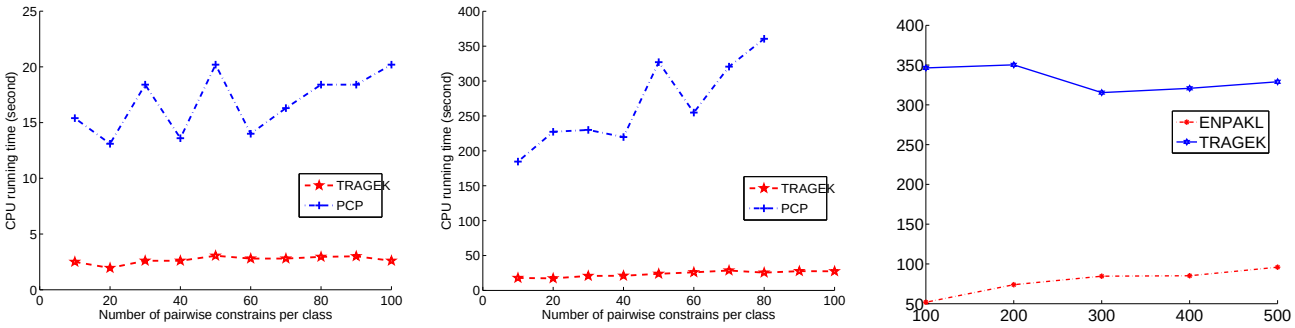
- `ENPAKL` is a little faster than `ENPAKL`^{1}, meanwhile both of them are much faster than `PCP` and `SSKK`.
- `TRAGEK` is apparently superior to `PCP` and `SSKK`, where these two algorithms even fail to compete with the traditional k-means algorithm.
- The performances of `ENPAKL` and `ENPAKL`^{1} are competitive, and represent the best algorithms in terms of effectiveness and efficiency.

5.4 Running Time

This section shows the running time of several related algorithms and we claim that the running time of `TRAGEK` is not sensitive to the the number of pairwise constraints. To test this, we perform experiments on the Heart dataset and the Chessboard dataset with increasing number of pairwise constraints from 10 to 100 with an interval of 10. The results are shown in Figure 5(a) and (b). We can see from the figures that as the number of pairwise constraints increases, the running time of `TRAGEK` varies little, whereas that of `PCP` increases dramatically.

Next we examined the efficiency of `ENPAKL`. We used the `Wisconsin` dataset and recorded the corresponding running time. Note that `PCP` is too time consuming when the constraints are large, thus we do not show its running time here.

We compared ENPAKL with TRAGEK, the results are shown in Figure 5(c). Obviously, ENPAKL is much more efficient than TRAGEK in term of computational complexity[6].



(a) Chessboard (b) Heart (c) Wisconsin

Fig. 5 Running time comparison. The x-axis is the number of constraints, the y-axis represents the running time in seconds.

5.5 Image Segmentation

In this section we applied ENPAKL for image segmentation by doing clustering on images. We tested our algorithm on the MSRC Object Category Image Database (v2) [39], which contains 791 images of size approximately 320×210, and includes different scenes such as grasses, forests, streets, *etc.*. In this experiment, we do not care about what feature we used. Instead, we want to test the effectiveness and robustness of the proposed algorithm against other popular clustering algorithms such as k-means, N-cut, and *etc.*. As a result we simply used the histogram features in the experiments (richer features for image segmentation would be our future work). Specifically, we divided each image into 5×5 patches, and exacted the color histograms of each patch as its features, and finally used these features to do the segmentation. We set the number of clusters to the ground truth, for ENPAKL, we randomly generated 50 must-link and cannot-link constraints for each cluster in the images. For simplicity, we compared the proposed ENPAKL algorithm with the k-means and N-cut algorithms[7] which are popularly used in image segmentation, and also because the above experiments have shown the superior of the k-means algorithm over PCP and SSKK. Some examples of the images and their segmentation results are shown in Figure 6. From these results we can see the superior of ENPAKL over the k-means and N-cut algorithms in term of segmentation accuracy, though it runs much slower, which is a typical problem for kernel based algorithms[8]. We used Eq. (34) as the segmentation accuracy criterion, and the corresponding accuracies are also shown in the figure. We see from the figure that ENPAKL performs best while N-cut and k-means are comparable. Also note that for some images, the segmentations learned by ENPAKL are very close to the Ground Truth, while those learned by the k-means and the N-cut are much worse, this indicates that supervisory information

[6] This experiment was run on an Intel Core 2 Duo CPU T6400 2.00GHZ with 2GB of DDR2 memory

[7] We used an efficient implementation of the N-cut algorithm in [40]

[8] The k-means algorithm takes about 1 second for one image, the N-cut algorithm takes about 2 seconds, while ENPAKL needs about 5 minutes, and PCP cannot run in this experiment because the corresponding data is too large. How to accelerate the speed of the proposed algorithm further is our future work.

could help image segmentation a lot, and it is encouraged to use such kind of information to boost the segmentation accuracy. We believe better segmentation results can be obtained by choosing the constraints carefully, by using other kinds of features such as the sift features [37] and rich textual features [38], and also by taking the spatial information into consideration.

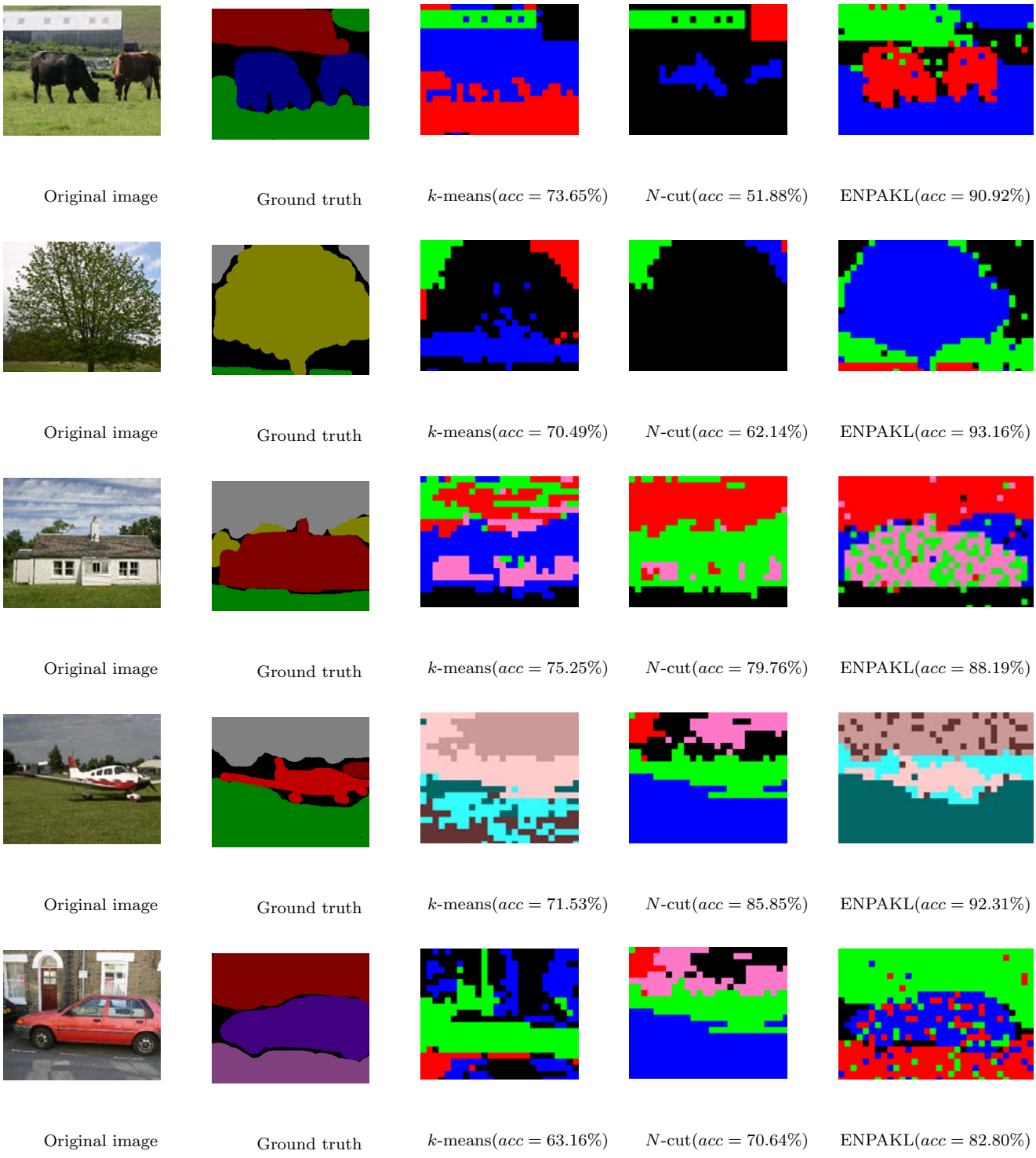


Fig. 6 Image segmentation using `ENPAKL`, k-means and N-cut. Here acc means segmentation accuracy evaluating using Eq. (34).

6 Conclusion

In this paper, we proposed a non-parametric kernel learning framework. It generalizes the graph embedding framework [30] into kernel space and is reformed as a conic optimization programming. A global Laplacian regularizer is used to smooth the functional space. Two algorithms are proposed for the corresponding kernel learning problem, one is to solve the original optimization problem through semi-definite programming. The other is to relax the `SDP` problem and solve with a constrained gradient descent based algorithm. To further reduce the computational complexity, the whole data is proposed to be divided into groups, and `sub-kernels` for these groups are learned separately, then the global kernel is constructed by combining these `sub-kernels`. Experiments are performed on nine datasets for clustering and one image dataset for image segmentation. Experimental results show that the proposed `ENPAKL` algorithm is superior to the recently developed algorithms [18,19] in terms of computational effectiveness and clustering accuracy, and often achieves better image segmentation.

We will study the parameters setting problem in the future. For example, the regularizer S in Eq. (19) may be replaced by a more sophisticated regularizer such as the s-weighted Laplacian operator [12]. The algorithms should also be evaluated with different settings of B and R in Algorithm 3, the k in the graph construction, *etc.*. Furthermore, we can incorporate `ENPAKL` into other kernel methods such as kernelization of some dimensional reduction algorithms. In addition, we will apply the proposed algorithm to more real applications, and explore more efficient algorithms for this problem since the current methods is not fast enough for large scale datasets.

References

1. A. Asuncion and D. J. Newman, "UCI machine learning repository," in *[http://www.ics.uci.edu/∼mlearn/MLRepository.html]*. University of California, Irvine, School of Information and Computer Sciences, 2007.
2. A. Baraldi, and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition–part II," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 29, no. 6, pp. 786–801, 1999.
3. A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints," *Journal of Machine Learning Research*, vol. 6, pp. 937–965, 2005.
4. S. Basu, M. Bilenko, and R. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 59–68, 2004.
5. C. Cortes, M. Mohri and A. Rostamizadeh, "Two-Stage Learning Kernel Algorithms," in *Proceedings of the 27st International Confernece on Machine Learning*, 2010.
6. C. Cortes, M. Mohri and A. Rostamizadeh, "Generalization Bounds for Learning Kernels," in *Proceedings of the 27st International Confernece on Machine Learning*, 2010.
7. R. Jin, S. C.H. Hoi, and T. Yang, "Online Multiple Kernel Learning: Algorithms and Mistake Bounds," in *Proceedings of the 21st International Conference Algorithmic Learning Theory*, pp. 390–404, 2010.
8. M. Bilenko, S. Basu, and R. Mooney, "Integrating constraints and metric learning in semi-supervised culstering," in *Proceedings of the 21st International Confernece on Machine Learning*, pp. 81–89, 2004.
9. O. Bousquet and D. Herrmann, "On the complexity of learning the kernel matrix," in *Advances in Neural Information Processing Systems 15*, pp. 399–406, 2003.
10. O. Chapelle, B. Schölkopf, and A. Zien, "Semi-supervised learning," *MIT Press, Cambridge, Massachusetts*, 2006.
11. I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556, 2004.

12. O. Duchenne, J-Y. Audibert, R. Keriven, J. Ponce, and F. Segonne, "Segmentation by transduction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
13. G. H. Golub and C. F. V. Loan, "Matrix Computation," *Johns Hopkins Univ. Press*, 1996.
14. T. Hertz, A. Bar-Hillel, and D. Weinshall, "Boosting margin based distance function for clustering," in *Proceedings of the 21st International Confernece on Machine Learning*, pp. 393–400, 2004.
15. S. C. H. Hoi, R. Jin, and M. R. Lyu, "Learning nonparametric kernel matrices from pairwise constraints," in *Proceedings of the 24th International Confernece on Machine Learning*, pp. 361–368, 2007.
16. Y. Hong, and S. Kwong, "Learning Assignment Order of Instances for the Constrained K-Means Clustering Algorithm," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 39, no. 2, pp. 568–574, 2009.
17. S. D. Kamvar, D. Klein, and C. Manning, "Spectral learning," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 561–566, 2003.
18. B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: A kernel approach," in *Proceedings of the 22nd International Confernece on Machine Learning*, pp. 457–464, 2005.
19. Z. Li, J. Liu, and X. Tang, "Pairwise constraint propagation by semidefinite programming for semi-supervised classification," in *Proceedings of the 25th International Confernece on Machine Learning*, pp. 576–583, 2008.
20. Z. Lu and T. K. Leen, "Semi-supervised learning with pernalized probabilistic clustering," in *Advances in Neural Information Processing Systems 17*, pp. 849–856, 2005.
21. S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
22. B. Schölkopf, and A. J. Smola, "Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," *MIT Press, Cambridge, Massachusetts*, 2001.
23. J. F. Sturm, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," *Optimization Methods & Software*, vol. 11, no. 2, pp. 625–653, 1999.
24. A. J. C. Trappey, C. V. Trappey, F.-C. Hsu, and D. W. Hsiao, "A Fuzzy Ontological Knowledge Document Clustering Methodology," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 39, no. 3, pp. 123–131, 2009.
25. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Confernece on Machine Learning*, pp. 798–803, 2001.
26. J. Zhuang, Ivor W. Tsang and S. C. H. Hoi, "SimpleNPKL: simple non-parametric kernel learning," in *Proceedings of the 26th International Confernece on Machine Learning*, pp. 1273–1280, 2009.
27. E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems 15*, 2003.
28. H. Xiong, J. Wu, and J. Chen, "K-Means Clustering Versus Validation Measures: A Data-Distribution Perspective," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 39, no. 2, pp. 318–331, 2009.
29. Z. Xu, M. Dai, and D. Meng, "Fast and Efficient Strategies for Model Selection of Gaussian Support Vector Machine," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 39, no. 5, pp. 1292–1307, 2009.
30. S. C. Yan, D. Xu, B. Y. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.
31. J. Yang, S. C. Yan, Y. Fu, X. L. Li, and T. S. Huang, "Non-negative graph embedding," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
32. M. S. Yang, K. L. Wu, J. N. Hsieh, and J. Yu, "Alpha-Cut Implemented Fuzzy Clustering Algorithms and Switching Regressions," *IEEE Transactions on Systems, Man, Cybernetics, part B*, vol. 38, no. 3, pp. 904–915, 2008.
33. D-Y. Yeung, H. Chang, and G. Dai, "A scalable kernel-based algorithm for semi-supervised metric learning," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1138–1143, 2007.
34. C. Cortes, M. Mohri and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Advances in Neural Information Processing Systems 21*, 2009.
35. D. Y. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proceedings of the 22nd International Confernece on Machine Learning*, pp. 1036–1043, 2005.

36. R. L. Adler, J. P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub, "Newton's method on Riemannian manifolds and a geometric model for the human spine," *IMA Journal of Numerical Analysis*, vol. 22, no. 3, pp. 359–390, 2002.

37. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

38. B. Tan, J. Zhang and L. Wang, "Semi-Supervised Elastic Net for Pedestrian Counting," *Pattern Recognition*, vol. 44, issues 10–11, pp. 2297–2304, 2011.

39. A. Criminisi, "MSRC Category Image Database(v2)," in *[http://research.microsoft.com/en-us/um/people/antcrim/data_objrec/msrc_objcategimagedatabase_v2.zip]*. MSRC.

40. J. Shi, "MATLAB Normalized Cuts Segmentation Code," in *[http://www.cis.upenn.edu/ jshi/software/]*.

41. J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.