Kernel Learning for Local Learning Based Clustering

Hong Zeng and Yiu-ming Cheung*

Department of Computer Science
Hong Kong Baptist University, Hong Kong SAR, China
{hzeng,ymc}@comp.hkbu.edu.hk

Abstract. For most kernel-based clustering algorithms, their performance will heavily hinge on the choice of kernel. In this paper, we propose a novel kernel learning algorithm within the framework of the Local Learning based Clustering (LLC) (Wu & Schölkopf 2006). Given multiple kernels, we associate a non-negative weight with each Hilbert space for the corresponding kernel, and then extend our previous work on feature selection (Zeng & Cheung 2009) to select the suitable Hilbert spaces for LLC. We show that it naturally renders a linear combination of kernels. Accordingly, the kernel weights are estimated iteratively with the local learning based clustering. The experimental results demonstrate the effectiveness of the proposed algorithm on the benchmark document datasets.

1 Introduction

In the past few decades, the kernel methods have been widely applied to various learning problems, where the data is implicitly mapped into a nonlinear high dimensional space by kernel function [3]. Unfortunately, it is known that the performance heavily hinges on the choice of kernel, and the most suitable kernel for a particular task is often unknown in advance. Thereby, learning an appropriate kernel, is critical to obtain an improved performance for the employed kernel-based inference method.

In this paper, we are particularly interested in the problem of kernel learning for clustering. In the literature, the kernel learning has been extensively studied for the supervised learning contexts. However, this issue remains less explored in unsupervised problems, due to the absence of ground truth class labels that could guide the learning for "ideal" kernels. Until very recently, several algorithms have been proposed to address this issue for clustering. Some approaches [4,5] directly learn the kernel parameters of some specific kernels. Though improvement is often achieved, extension of the learning method to other kernel functions is often nontrivial. A more effective framework, termed as the multiple kernel learning [6], learns a linear combination of base kernels with different weights, which will be estimated iteratively with the inference process [7,8]. This strategy may bring potential advantages over those which try to obtain a single best kernel, through exploiting the complementary information among different kernels. In [7], the algorithm tries to find a maximum margin hyperplane to

* Yiu-ming Cheung (ymc@comp.hkbu.edu.hk) is the corresponding author. This work was supported by the Faculty Research Grant of HKBU under Project: FRG/07-08/II-54, and the Research Grant Council of Hong Kong SAR under Grant: HKBU 210306.

C. Alippi et al. (Eds.): ICANN 2009, Part I, LNCS 5768, pp. 10–19, 2009.

cluster data (restricted to binary-class case), accompanied with learning a mixture of Laplacian matrices. In [8], clustering is phrased as a non-negative matrix factorization problem of a fused kernel matrices. Nevertheless, both approaches in [7,8] are global learning based. Their performance may be degraded when samples are less separable from a global view.

Under the circumstances, we therefore propose a novel multiple kernel learning method within the framework of the Local Learning based Clustering (LLC) [1], which aims at optimizing the local purity requirement of clustering assignment. It is expected that it will produce a more reliable intermediate clustering result when the samples are globally less separable. We associate a non-negative weight with each Hilbert space (or called the feature space interchangeably) for the corresponding kernel, and then extend our previous work on feature selection [2] to select the suitable Hilbert spaces for LLC. Such strategy naturally leads to learn a linear combination of all the available kernels at hand. Accordingly, an algorithm is developed in which the combination coefficients of kernels are estimated iteratively with the local learning based clustering.

The remainder of the paper is organized as follows: Section 2 gives an overview of local learning based clustering algorithm. We present the proposed method in Section 3. In Section 4, the experiments on several benchmark datasets are presented. We draw a conclusion in Section 5.

2 Overview of the Local Learning Based Clustering Algorithm

Let us first introduce the indicator matrix that will be used later. Suppose n data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n (\mathbf{x}_i \in \mathbb{R}^d)$ will be partitioned into C clusters. The clustering result can be represented by a *cluster assignment indicator matrix* $\mathbf{P} = [p_{ic}] \in \{0,1\}^{n \times C}$, such that $p_{ic} = 1$ if $\mathbf{x}_i$ belongs to the cth cluster, and $p_{ic} = 0$ otherwise. The *scaled cluster assignment indicator matrix* used in this paper is defined by: $\mathbf{Y} = \mathbf{P}(\mathbf{P}^T\mathbf{P})^{-\frac{1}{2}} = [\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^C]$, where $\mathbf{y}^c = [y_{1c}, \ldots, y_{nc}]^T \in \mathbb{R}^n (1 \le c \le C)$, is the c-th column of $\mathbf{Y} \in \mathbb{R}^{n \times C}$. $y_{ic} = p_{ic}/\sqrt{n_c}$ can be regarded as the confidence that $\mathbf{x}_i$ is assigned to the cth cluster, where n_c is the size of the cth cluster. It is easy to verify that $\mathbf{Y}^T\mathbf{Y} = \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix.

The starting point of the LLC [1] is that the cluster assignments in the neighborhood of each point should be as pure as possible. Suppose there exists an arbitrary $\mathbf{Y}$ at first, for each $\mathbf{x}_i$, a regression model is built with the training data $\{(\mathbf{x}_j, y_{jc})\}_{\mathbf{x}_j \in \mathcal{N}_i} (1 \le c \le C, 1 \le i, j \le n)$, where $\mathcal{N}_i$ denotes the set of neighboring[1] points of $\mathbf{x}_i$ (not including $\mathbf{x}_i$ itself). The output of the local model is of the following form: $f_i^c(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\theta}_i^c, \forall \mathbf{x} \in \mathbb{R}^d$, where $\boldsymbol{\theta}_i^c \in \mathbb{R}^d$ is the local regression coefficients vector. Here, the bias term is ignored for simplicity, provided that one of the features is 1. In [1], $\boldsymbol{\theta}_i^c$ is solved by:

$$\min_{\boldsymbol{\theta}_i^c} \sum_{c=1}^{C} \sum_{i=1}^{n} \Big[\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta(y_{jc} - \mathbf{x}_j^T\boldsymbol{\theta}_i^c)^2 + \|\boldsymbol{\theta}_i^c\|^2 \Big], \tag{1}$$

[1] The k-mutual neighbors are adopted in order to well describe the local structure, i.e. $\mathbf{x}_j$ is considered as a neighbor of $\mathbf{x}_i$ only if $\mathbf{x}_i$ is also one of the k-nearest neighbors of $\mathbf{x}_j$.

where β is a trade-off parameter. Denote the solution to the linear ridge regression problem (1) as $\boldsymbol{\theta}_i^{c*}$, the predicted cluster assignment for the test data $\mathbf{x}_i$ can then be calculated by: $\widehat{y}_{ic} = f_i^c(\mathbf{x}_i) = \mathbf{x}_i^T \boldsymbol{\theta}_i^{c*} = \boldsymbol{\alpha}_i^T \mathbf{y}_i^c$, where

$$\boldsymbol{\alpha}_i^T = \beta \mathbf{x}_i^T (\beta \mathbf{X}_i \mathbf{X}_i^T + \mathbf{I})^{-1} \mathbf{X}_i, \tag{2}$$

$\mathbf{X}_i = [\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_{n_i}}]$ with $\mathbf{x}_{i_k}$ being the k-th neighbor of $\mathbf{x}_i$, n_i is the size of $\mathcal{N}_i$, and $\mathbf{y}_i^c = [y_{i_1 c}, y_{i_2 c}, \ldots, y_{i_{n_i} c}]^T$.

After all the local predictors have been constructed, LLC aims to find an optimal cluster indicator matrix $\mathbf{Y}$ via minimizing the overall prediction errors:

$$\sum_{c=1}^{C} \sum_{i=1}^{n} (y_{ic} - \widehat{y}_{ic})^2 = \sum_{c=1}^{C} \|\mathbf{y}^c - \mathbf{A}\mathbf{y}^c\|^2 = trace(\mathbf{Y}^T \mathbf{T} \mathbf{Y}), \tag{3}$$

where $\mathbf{T} = (\mathbf{I} - \mathbf{A})^T (\mathbf{I} - \mathbf{A})$, $\mathbf{A}$ is an $n \times n$ sparse matrix with its (i, j)-th entry a_{ij} being the corresponding element in $\boldsymbol{\alpha}_i$ by (2) if $\mathbf{x}_j \in \mathcal{N}_i$ and 0 otherwise.

As in the spectral clustering [9,10], $\mathbf{Y}$ is relaxed into the continuous domain while keeping the property $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$ for (3). LLC then solves:

$$\min_{\mathbf{Y} \in \mathbb{R}^{n \times C}} \quad trace(\mathbf{Y}^T \mathbf{T} \mathbf{Y}) \;\; s.t. \; \mathbf{Y}^T \mathbf{Y} = \mathbf{I} \tag{4}$$

A solution to $\mathbf{Y}$ is given by the first C eigenvectors of the matrix $\mathbf{T}$, corresponding to the first C smallest eigenvalues. The final partition result is obtained by discretizing $\mathbf{Y}$ via the method in [10] or by k-means as in [9].

3 Multiple Kernel Learning for Local Learning Based Clustering

The LLC algorithm can be easily kernelized as in [1], by replacing the linear ridge regression with the kernel ridge regression. Under the circumstances, selecting a suitable kernel function will be a crucial issue. We extend our previous work of feature selection for LLC [2] to learn a proper linear combination of several pre-computed kernel matrices.

In the kernel methods, the symmetric positive semi-definite kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, implicitly maps the raw input features into a high-dimensional (possibly infinite) *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{H}$, which is equipped with the inner product $< \cdot, \cdot >_{\mathcal{H}}$ via a nonlinear mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$, i.e., $\mathcal{K}(x, z) =< \phi(x), \phi(z) >_{\mathcal{H}}$. Suppose there are L different available kernel functions $\{\mathcal{K}^{(l)}\}_{l=1}^{L}$. Accordingly, there are L different associated feature spaces $\{\mathcal{H}^{(l)}\}_{l=1}^{L}$. Since it is unknown which feature space should be used, an intuitive way is to use them all by concatenating all feature spaces into an augmented Hilbert space: $\widetilde{\mathcal{H}} = \bigoplus_{l=1}^{L} \mathcal{H}^{(l)}$, and associate each feature space with a relevance weight τ_l ($\sum_{l=1}^{L} \tau_l = 1, \tau_l \geq 0, \forall l$), or equivalently the importance factor for kernel function $\mathcal{K}^{(l)}$. Later, we will show that performing LLC in such feature space is equivalent to employing a combined kernel function: $\mathcal{K}^{\tau}(x, z) = \sum_{l=1}^{L} \tau_l \mathcal{K}^{(l)}(x, z)$ for LLC. A zero weight τ_l will correspond to *blend out* the feature space associated with the corresponding kernel similar to the feature selection in [2]. Our task is to learn the coefficients $\{\tau_l\}_{l=1}^{L}$ which can lead to a more accurate and robust performance. Subsequently, an algorithm that iteratively performs clustering and estimates the kernel weight is developed.

3.1 Update Y for a Given $\boldsymbol{\tau}$

First of all, given a $\boldsymbol{\tau}$, the nearest neighbors $\mathcal{N}_i$ for LLC algorithm will be re-found by the $\boldsymbol{\tau}$-weighted squared Euclidean distance in $\widetilde{\mathcal{H}}$, i.e.:

$$d_\tau(\mathbf{x}_1, \mathbf{x}_2) = \|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_\tau^2 = \mathcal{K}^\tau(\mathbf{x}_1, \mathbf{x}_1) + \mathcal{K}^\tau(\mathbf{x}_2, \mathbf{x}_2) - 2\mathcal{K}^\tau(\mathbf{x}_1, \mathbf{x}_2). \quad (5)$$

Then the local discriminant function in the $\widetilde{\mathcal{H}}$ can be written as follows:

$$f_i^c(\phi(\mathbf{x})) = \phi(\mathbf{x})^T \mathbf{w}_i^c + b_i^c, \quad (6)$$

where $\phi(\mathbf{x}) = [\phi_1(\mathbf{x})\ \phi_2(\mathbf{x})\ \cdots\ \phi_L(\mathbf{x})]^T \in \mathbb{R}^D$, $\phi_l(\mathbf{x}) \in \mathbb{R}^{D_l}$ is the sample mapped by the lth kernel function, $\sum_{l=1}^L D_l = D$, D and D_l are the dimensionalities of $\widetilde{\mathcal{H}}$ and $\mathcal{H}^{(l)}$, respectively. Taking the relevance of each feature space for clustering into account, the regression coefficient $\mathbf{w}_i^c \in \mathbb{R}^D$ and the bias $b_i^c \in \mathbb{R}$ now will be solved via the following weighted l_2 norm regularized least square problem:

$$\min_{\mathbf{w}_i^c, b_i^c} \sum_{c=1}^{C} \sum_{i=1}^{n} \Big[\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta(y_{jc} - \phi(\mathbf{x}_j)^T \mathbf{w}_i^c - b_i^c)^2 + \mathbf{w}_i^{cT} \boldsymbol{\Lambda}_\tau^{-1} \mathbf{w}_i^c \Big], \quad (7)$$

where $\boldsymbol{\Lambda}_\tau$ is a diagonal matrix with the vector $\widetilde{\boldsymbol{\tau}} = (\underbrace{\tau_1, \ldots, \tau_1}_{D_1}, \ldots, \underbrace{\tau_L, \ldots, \tau_L}_{D_L})^T$ in the diagonal, and $\sum_{l=1}^L \tau_l = 1, \tau_l \geq 0\ \forall l$. Similar to [2], the weighted l_2 norm (i.e., the second term in the square bracket of (7)) with $\boldsymbol{\tau}$ defined on the standard simplex is able to provide adaptive regularization: a large penalty will be imposed on the elements of $\mathbf{w}_i^c$ corresponding to the feature spaces associated with irrelevant kernels. Thus, an improved clustering result can be expected because the vanishing elements in $\mathbf{w}_i^c$ will eliminate the feature spaces with irrelevant kernels from the prediction (c.f. (6)).

After removing the bias term by plugging its optimal solution

$$b_i^c = \frac{1}{n_i} \mathbf{e}_i^T (\mathbf{y}_i^c - \phi(\mathbf{X}_i)^T \mathbf{w}_i^c), \quad (8)$$

into (7), where $\mathbf{e}_i = [1\ 1 \cdots 1]^T \in \mathbb{R}^{n_i}$, we can reformulate the primal problem (7) as follows:

$$\min_{\mathbf{w}_i^c} \sum_{c=1}^{C} \sum_{i=1}^{n} \Big[\beta \|\boldsymbol{\Pi}_i \mathbf{y}_i^c - (\phi(\mathbf{X}_i)\boldsymbol{\Pi}_i)^T \mathbf{w}_i^c\|^2 + \mathbf{w}_i^{cT} \boldsymbol{\Lambda}_\tau^{-1} \mathbf{w}_i^c \Big], \quad (9)$$

where $\boldsymbol{\Pi}_i = \mathbf{I}_{n_i} - \frac{1}{n_i} \mathbf{e}_i \mathbf{e}_i^T$ and $\mathbf{I}_{n_i} \in \mathbb{R}^{n_i \times n_i}$ is a unit matrix, $\boldsymbol{\Pi}_i \boldsymbol{\Pi}_i = \boldsymbol{\Pi}_i$. Then we consider the dual formulation of the (9) in terms of $\mathbf{w}_i^c$. Denote

$$\boldsymbol{\zeta}_i^c = (\phi(\mathbf{X}_i)\boldsymbol{\Pi}_i)^T \mathbf{w}_i^c - \boldsymbol{\Pi}_i \mathbf{y}_i^c, \quad (10)$$

then the Lagrangian for problem (9) is

$$\mathcal{L}(\{\boldsymbol{\zeta}_i^c, \mathbf{w}_i^c, \boldsymbol{\gamma}_i^c\}) = \sum_{c=1}^{C} \sum_{i=1}^{n} \Big(\beta \|\boldsymbol{\zeta}_i^c\|^2 + \mathbf{w}_i^{cT} \boldsymbol{\Lambda}_\tau^{-1} \mathbf{w}_i^c \Big) \quad (11)$$

$$- \sum_{c=1}^{C} \sum_{i=1}^{n} \boldsymbol{\gamma}_i^{cT} \Big((\phi(\mathbf{X}_i)\boldsymbol{\Pi}_i)^T \mathbf{w}_i^c - \boldsymbol{\Pi}_i \mathbf{y}_i^c - \boldsymbol{\zeta}_i^c \Big),$$

where $\boldsymbol{\gamma}_i^c$'s with $\boldsymbol{\gamma}_i^c \in \mathbb{R}^{n_i}$ are the vectors of Lagrangian dual variables. Taking the derivatives of $\mathcal{L}$ w.r.t. the primal variables $\boldsymbol{\zeta}_i^c$ and $\mathbf{w}_i^c$, and setting them equal to zero, we obtain:

$$\boldsymbol{\zeta}_i^c = -\frac{\boldsymbol{\gamma}_i^c}{2\beta},\ \mathbf{w}_i^c = \frac{\boldsymbol{\Lambda}_\tau \phi(\mathbf{X}_i)\boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^c}{2}, \tag{12}$$

and finally we obtain the dual problem:

$$\begin{aligned}\max_{\boldsymbol{\gamma}_i^c} \sum_{c=1}^{C}\sum_{i=1}^{n} -\frac{1}{4\beta}\boldsymbol{\gamma}_i^{cT}\boldsymbol{\gamma}_i^c - \frac{1}{4}\boldsymbol{\gamma}_i^{cT}\boldsymbol{\Pi}_i\phi(\mathbf{X}_i)^T\boldsymbol{\Lambda}_\tau\phi(\mathbf{X}_i)\boldsymbol{\Pi}_i\boldsymbol{\gamma}_i^c + \boldsymbol{\gamma}_i^{cT}\boldsymbol{\Pi}_i\mathbf{y}_i^c = \\ \max_{\boldsymbol{\gamma}_i^c} \sum_{c=1}^{C}\sum_{i=1}^{n} -\frac{1}{4\beta}\boldsymbol{\gamma}_i^{cT}\boldsymbol{\gamma}_i^c - \frac{1}{4}\boldsymbol{\gamma}_i^{cT}\boldsymbol{\Pi}_i\mathbf{K}_i^\tau\boldsymbol{\Pi}_i\boldsymbol{\gamma}_i^c + \boldsymbol{\gamma}_i^{cT}\boldsymbol{\Pi}_i\mathbf{y}_i^c.\end{aligned} \tag{13}$$

with $\phi(\mathbf{X}_i)^T\boldsymbol{\Lambda}_\tau\phi(\mathbf{X}_i) = \sum_{l=1}^{L}\tau_l\phi_l(\mathbf{X}_i)^T\phi_l(\mathbf{X}_i) = \sum_{l=1}^{L}\tau_l\mathbf{K}_i^{(l)} = \mathbf{K}_i^\tau$, where $\mathbf{K}_i^{(l)}, \mathbf{K}_i^\tau \in \mathbb{R}^{n_i\times n_i}$ are the base and combined kernel matrices over $\mathbf{x}_j \in \mathcal{N}_i$, respectively, i.e., $\mathbf{K}_i^{(l)} = [\mathcal{K}^{(l)}(\mathbf{x}_u, \mathbf{x}_v)]$ and $\mathbf{K}_i^\tau = [\mathcal{K}^\tau(\mathbf{x}_u, \mathbf{x}_v)]$, for $\mathbf{x}_u, \mathbf{x}_v \in \mathcal{N}_i$. For fixed $\boldsymbol{\tau}$ constrained on the simplex, the convex combination of the positive semi-definite kernel matrices : $\mathbf{K}_i^\tau = \sum_{l=1}^{L}\tau_l\mathbf{K}_i^{(l)}$ is still a positive semi-definite kernel matrix. Therefore, the problem in (13) is an unconstrained concave quadratic program whose unique optimal solution can be obtained analytically:

$$\boldsymbol{\gamma}_i^{c*} = 2\beta(\mathbf{I}_i + \beta\boldsymbol{\Pi}_i\mathbf{K}_i^\tau\boldsymbol{\Pi}_i)^{-1}\boldsymbol{\Pi}_i\mathbf{y}_i^c. \tag{14}$$

Then altogether with (8), (12) and (14), the predicted indicator value at point $\mathbf{x}_i$ for the cth ($c = 1, \ldots, C$) cluster can be calculated by (6): $\widehat{y}_{ic} = f_i^c(\phi(\mathbf{x}_i)) = \phi(\mathbf{x}_i)^T\mathbf{w}_i^c + b_i^c = \boldsymbol{\alpha}_i^T\mathbf{y}_i^c$, with

$$\boldsymbol{\alpha}_i^T = \beta(\mathbf{k}_i^\tau - \frac{1}{n_i}\mathbf{e}_i^T\mathbf{K}_i^\tau)\boldsymbol{\Pi}_i\Big[\mathbf{I}_i - (\beta^{-1}\mathbf{I} + \boldsymbol{\Pi}_i\mathbf{K}_i^\tau\boldsymbol{\Pi}_i)^{-1}\boldsymbol{\Pi}_i\mathbf{K}_i^\tau\boldsymbol{\Pi}_i\Big] + \frac{1}{n_i}\mathbf{e}_i^T, \tag{15}$$

where $\mathbf{k}_i^\tau \in \mathbb{R}^{n_i}$ denotes the vector $[\mathcal{K}^\tau(\mathbf{x}_i, \mathbf{x}_j)]^T$ for $\mathbf{x}_j \in \mathcal{N}_i$.

To obtain $\mathbf{Y}$, we will first build the matrix $\mathbf{T}$ by (3) with $\boldsymbol{\alpha}_i$ defined in (15), using the combined kernel $\mathcal{K}^\tau(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{L}\tau_l\mathcal{K}^{(l)}(\mathbf{x}_i, \mathbf{x}_j)$. Then $\mathbf{Y}$ is given by the first C eigenvectors of $\mathbf{T}$ corresponding to the C smallest eigenvalues.

3.2 Update $\boldsymbol{\tau}$ for a Given $\mathbf{Y}$

Subsequently, the L kernel combination coefficients $\{\tau_l\}_{l=1}^L$ will be recomputed based on the current estimation for $\mathbf{Y}$. We propose to estimate $\boldsymbol{\tau}$ using the *projected gradient descent* method as in [11,12].

With fixed $\mathbf{Y}$ and neighborhood determined at each point, an optimal τ is expected to minimize:

$$\mathcal{P}(\boldsymbol{\tau}),\ s.t.\ \sum_{l=1}^{L}\tau_l = 1, \tau_l \geq 0, \forall l, \tag{16}$$

where $\mathcal{P}(\boldsymbol{\tau}) = \min_{\mathbf{w}_i^c} \sum_{c=1}^{C} \sum_{i=1}^{n} \left[\beta \|\boldsymbol{\Pi}_i \mathbf{y}_{ic} - (\phi(\mathbf{X}_i)\boldsymbol{\Pi}_i)^T \mathbf{w}_i^c\|^2 + \mathbf{w}_i^{cT} \boldsymbol{\Lambda}_\tau^{-1} \mathbf{w}_i^c\right]$. In general, it can be solved by the projected gradient descent method through the update equation $\boldsymbol{\tau}^{(new)} = \boldsymbol{\tau}^{(old)} - \eta \boldsymbol{\nabla} \mathcal{P}$, as given $\mathbf{Y}$ and $\mathcal{N}_i$ such that $\mathcal{P}(\boldsymbol{\tau}^{(new)}) \leq \mathcal{P}(\boldsymbol{\tau}^{(old)})$, where η is the step size, and $\boldsymbol{\nabla} \mathcal{P}$ is the projected gradient. It is expected that the local regression model derived from $\boldsymbol{\tau}^{(new)}$ should be better than the one derived from $\boldsymbol{\tau}^{(old)}$. Nevertheless, since both $\mathbf{Y}$ and $\mathcal{N}_i$ depend on $\boldsymbol{\tau}$ as shown in Section 3.1, they need to be recomputed as in Section 3.1 once $\boldsymbol{\tau}$ is updated.

Then the key issue is to obtain the derivatives of $\mathcal{P}(\boldsymbol{\tau})$ in analytic forms. In order to do so, we resort to the dual of $\mathcal{P}(\boldsymbol{\tau})$ which has been investigated in Sub-section 3.1, and is rewritten below:

$$\mathcal{D}(\boldsymbol{\tau}) = \max_{\boldsymbol{\gamma}_i^c} \sum_{c=1}^{C} \sum_{i=1}^{n} -\frac{1}{4\beta} \boldsymbol{\gamma}_i^{cT} \boldsymbol{\gamma}_i^c - \frac{1}{4} \boldsymbol{\gamma}_i^{cT} \boldsymbol{\Pi}_i \mathbf{K}_i^\tau \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^c + \boldsymbol{\gamma}_i^{cT} \boldsymbol{\Pi}_i \mathbf{y}_i^c. \tag{17}$$

Note (9) is convex with respect to $\mathbf{w}_i^c$. By the principle of strong duality, we have $\mathcal{P}(\boldsymbol{\tau}) = \mathcal{D}(\boldsymbol{\tau})$. Furthermore, as $\{\boldsymbol{\gamma}_i^{c*}\}$ in (14) maximizes $\mathcal{D}$, according to [13], $\mathcal{D}(\boldsymbol{\tau})$ is differentiable if $\{\boldsymbol{\gamma}_i^{c*}\}$'s are unique. Fortunately, this unicity is guaranteed by the unconstrained concave quadratic program in (13). Moreover, as proved in Lemma 2 of [14], $\mathcal{D}(\boldsymbol{\tau})$ can be differentiated with respect to $\boldsymbol{\tau}$ as if $\{\boldsymbol{\gamma}_i^{c*}\}$ did not depend on $\boldsymbol{\tau}$. Finally, we have:

$$\frac{\partial \mathcal{P}}{\partial \tau_l} = \frac{\partial \mathcal{D}}{\partial \tau_l} = -\frac{1}{4} \sum_{c=1}^{C} \sum_{i=1}^{n} \boldsymbol{\gamma}_i^{c*T} \boldsymbol{\Pi}_i \mathbf{K}_i^{(l)} \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^{c*} = -\frac{1}{4} \sum_{i=1}^{n} trace(\boldsymbol{\gamma}_i^{*T} \boldsymbol{\Pi}_i \mathbf{K}_i^{(l)} \boldsymbol{\Pi}_i \boldsymbol{\gamma}_i^*), \tag{18}$$

where $\boldsymbol{\gamma}_i^* = [\boldsymbol{\gamma}_i^{1*}, \ldots, \boldsymbol{\gamma}_i^{C*}] \in \mathbb{R}^{n_i \times C}$.

Note the equality and non-negative constraints over the $\boldsymbol{\tau}$ have to be kept inviolate when updating $\boldsymbol{\tau}$ along the descent gradient direction. We use the same strategy as in [12] by first projecting the gradient to enforce the equality, and then ensuring that the descent direction does not lead to negative τ_l. That is, each element of the reduced gradient $\boldsymbol{\nabla} \mathcal{P}$ is designed as follows:

$$(\boldsymbol{\nabla} \mathcal{P})_l = \begin{cases} \frac{\partial \mathcal{P}}{\partial \tau_l} - \frac{\partial \mathcal{P}}{\partial \tau_m}, & \text{if } l \neq m \text{ and } \tau_l > 0; \\ \sum_{\mu \neq m, \tau_\mu > 0} \left(\frac{\partial \mathcal{P}}{\partial \tau_m} - \frac{\partial \mathcal{P}}{\partial \tau_\mu} \right), & \text{if } l = m; \\ 0, & \text{if } \tau_l = 0 \text{ and } \frac{\partial \mathcal{P}}{\partial \tau_l} - \frac{\partial \mathcal{P}}{\partial \tau_m} > 0, \end{cases} \tag{19}$$

where $m = \arg\max_l \tau_l$. When updating $\boldsymbol{\tau}$ by $\boldsymbol{\tau}^{(new)} = \boldsymbol{\tau}^{(old)} - \eta \boldsymbol{\nabla} \mathcal{P}$, we first try η with the maximal admissible step size η_{max} which sets τ_ν to zero, where

$$\nu = \arg \min_{\{l | (\boldsymbol{\nabla} \mathcal{P})_l > 0\}} \frac{\tau_l^{(old)}}{(\boldsymbol{\nabla} \mathcal{P})_l}, \eta_{max} = \frac{\tau_\nu}{(\boldsymbol{\nabla} \mathcal{P})_\nu}. \tag{20}$$

If $\mathcal{D}(\boldsymbol{\tau}^{(trial)}) \leq \mathcal{D}(\boldsymbol{\tau}^{(old)})$, where $\boldsymbol{\tau}^{(trial)} = \boldsymbol{\tau}^{(old)} - \eta_{max} \boldsymbol{\nabla} \mathcal{P}$, $\boldsymbol{\tau}$ gets updated; otherwise, a one-dimensional line search for $\eta \in [0, \eta_{max}]$ is applied. Algorithm 1 describes the steps to update $\boldsymbol{\tau}$.

Algorithm 1. Update kernel weight vector $\boldsymbol{\tau}$ with the current $\mathbf{Y}$ and $\mathcal{N}_i$

Compute the projected gradient $\nabla\mathcal{P}$ by (19);
Compute the maximal admissible step size η_{max} by (20);
$\boldsymbol{\tau}^{(trial)} = \boldsymbol{\tau}^{(old)} - \eta_{max}\nabla\mathcal{P}$;
Compute $\mathcal{D}(\boldsymbol{\tau}^{(trial)})$ with $\{\boldsymbol{\gamma}_i^*\}$ calculated from $\mathbf{K}^{\tau^{(trial)}} = \sum_{l=1}^{L} \tau_l^{(trial)}\mathbf{K}^{(l)}$;
if $\mathcal{D}(\boldsymbol{\tau}^{(trial)}) \leq \mathcal{D}(\boldsymbol{\tau}^{(old)})$ **then**
  $\eta = \eta_{max}$;
else
  Perform line search for $\eta \in [0, \eta_{max}]$ along $\nabla\mathcal{P}$;
end
$\boldsymbol{\tau}^{(new)} = \boldsymbol{\tau}^{(old)} - \eta\nabla\mathcal{P}$;

3.3 The Complete Algorithm

The complete local learning based clustering algorithm with multiple kernel learning (denoted as LLC-mkl) is presented in Algorithm 2. The loop stops when the relative variation of the trace value in (4) between two consecutive iterations is below a threshold (we set it at 10^{-4} in this paper), indicating the partitioning has almost been stabilized. After the convergence, $\mathbf{Y}$ is discretized to obtain the final clustering result with the k-means as in [9].

Algorithm 2. Multiple kernel learning for local learning based clustering algorithm

input : L base kernel matrices $\mathbf{K}^{(l)}$'s, size of the neighborhood k, trade-off parameter β
output: $\mathbf{Y}, \boldsymbol{\tau}$

1 Initialize $\tau_l = \frac{1}{L}$, for $l = 1, \ldots, L$;
2 **while** *not converge* **do**
3 Find k-mutual neighborhoods, using the metric defined in (5);
4 Construct the matrix $\mathbf{T}$ by (3) with $\boldsymbol{\alpha}_i$ given in (15), and then solve the problem (4) to obtain $\mathbf{Y}$;
5 Update $\boldsymbol{\tau}$ with the steps described in Algorithm 1;
6 **end**

4 Experimental Results

Experiments on document clustering were conducted with LLC-mkl. The characteristics of the benchmark document datasets used in this experiment are summarized in Table 1.

- **`CSTR`:** This is the dataset of the abstracts of technical reports published in the Department of Computer Science at a university between 1991 and 2002. The dataset contains 476 abstracts, which are divided into four research topics.
- **`WebACE`:** This dataset is from WebACE project, and it contains 2340 documents consisting of news articles from Reuters news service with 20 different topics in October 1997.

Table 1. Characteristics of the document datasets

Dataset	Number of Samples (n)	Number of Classes (C)
CSTR	476	4
WebACE	2340	20
tr11	414	9
tr31	927	7

– **tr11** and **tr31:** Both of the two datasets are from the CLUTO toolkit [15], they contain 414 and 927 articles categorized into 9 and 7 topics, respectively.

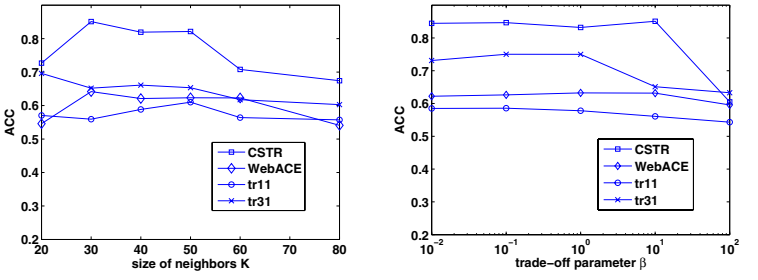
To pre-process the CSTR and WebACE datasets, we remove the stop words using a standard stop list, all HTML tags are skipped and all header fields except subject and organization of the posted articles are ignored. Then each document is represented by the term-frequency vector (Bag-of-Words). The datasets associated with the CLUTO toolkit have already been preprocessed. For all datasets, we used the top 1000 words by mutual information with class labels. For comparison, the counterpart unsupervised multiple kernel learning algorithm based on NMF [8] (denoted as NMF-mkl) was conducted. We also compared with the self-tuning spectral clustering [4] (denoted as SelfTunSpec), which tries to build a single best kernel for clustering. The algorithm in [7] is not compared because the optimization software in [7] cannot deal with the datasets that have too many samples and will cause memory overflow on the datasets used in this paper. Furthermore, we simply set the number of clusters equal to the number of classes in each dataset for all the algorithms without considering the selection of the optimal number of clusters, which is beyond the scope of this paper . We evaluated the performance with the clustering accuracy (ACC) index [1] for all algorithms. The sensitivity of the proposed LLC-mkl algorithm with respect to k and β will be presented at the end of this section.

We applied the LLC-mkl with altogether 10 pre-computed base kernels, i.e., 7 RBF kernels $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\delta^2)$, with $\delta = const * D$, where D is the maximum distance between samples, and $const$ varies in the pre-specified range $\{0.01, 0.05, 0.1, 1, 10, 50, 100\}$, 2 polynomial kernels $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^d$ with degree $d = \{2, 4\}$, and a cosine kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j/(\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|)$. All the kernels have been normalized through: $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)/\sqrt{\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i)\mathcal{K}(\mathbf{x}_j, \mathbf{x}_j)}$. Besides, we also implemented the case where each time a single candidate kernel $\mathcal{K}^{(l)}(l = 1, \ldots, 10)$ was adopted in the LLC algorithm in which the local prediction is performed with kernel ridge regression. The best (denoted as LLC-bkernel) and the worst (denoted as LLC-wkernel) performance out of the 10 kernels were reported. NMF-mkl was applied on the same 10 base kernels. The adjacency matrix in SelfTunSpec [4] was built by its local scaling method [4] on the dataset. For NMF-mkl and SelfTunSpec, we only reported the best accuracy among extensive trials of their free parameters. For LLC-mkl, the mean and standard deviation of ACC with $k = 30, \beta = 10$ over 10 runs were reported. The results are summarized in Table 2.

From Table 2, we could first observe that there is a big gap between the best and the worst performance of LLC with different choices of kernel. On the tr11 and tr31

Table 2. Accuracies of various methods on the document datasets

Data Set	LLC-wkernel	LLC-bkernel	LLC-mkl	NMF-mkl	SelfTunSpec
CSTR	0.3487	0.7374	**0.8508±0.0012**	0.6387	0.5210
WebACE	0.2436	0.4885	**0.6316±0.0215**	0.4960	0.4880
tr11	0.4251	**0.5966**	0.5609±0.0166	0.5145	0.4106
tr31	0.5297	**0.6721**	0.6512±0.0007	0.5372	0.4412



(a) document datasets (b) document datasets

Fig. 1. The parameter sensitivity studies of LLC-mkl algorithm. (a) varying the size of neighborhood with β being fixed at 10; (b) varying β with the size of neighborhood fixed at 30. The values on each line represent the average ACC over 10 independent runs.

datasets, the performance of LLC-mkl is close to that of the LLC with the best kernel, but obviously LLC-mkl is more sensible for practical application where we often do not know which kernel is the best *a priori*. On the CSTR and WebACE datasets, the LLC-mkl even outperforms the LLC with the best kernel. Namely, by combining multiple kernels and exploiting the complementary information contained in different kernels, the LLC-mkl indeed improves the robustness and accuracy of LLC. Compared to NMF-mkl which is derived globally, the LLC-mkl is consistently superior over it on these four datasets. A plausible reason is that the document datasets are very sparse, therefore the entries in the kernel matrix may resemble to each other from the global view or on a large scale. Thereby, finding the similar points locally may produce more reliable intermediate clustering result to guide the kernel learning. From Table 2, it can also be seen that the LLC-mkl and NMF-mkl both outperform the selfTunSpec which tries to construct a single "best" kernel in this experiment.

The effects of these two parameters, i.e., k and β, on the performance of LLC-mkl are presented in Figure 1. From Figure 1, it can be seen that the proposed LLC-mkl algorithm with $k = 30 \sim 50$ and $\beta \in [0.01, 10]$ could produce considerably accurate results and the performance does not vary much.

5 Conclusion

In this paper, a novel kernel learning approach has been proposed for the local learning based clustering, where a combination of kernels is jointly learned with the clustering.

It is addressed under a regularization framework by taking the relevance of each kernel into account. Experimental results have shown that the proposed kernel learning method is able to improve the robustness and accuracy of the basic local learning clustering. Furthermore, it generally outperforms the state-of-the-art counterparts, especially when the samples are less separable from a global view.

References

1. Wu, M., Schölkopf, B.: A Local Learning Approach for Clustering. In: NIPS, pp. 1529–1536 (2006)
2. Zeng, H., Cheung, Y.M.: Feature Selection for Local Learning based Clustering. In: PAKDD, pp. 414–425 (2009)
3. Scholköpf, B., Smola, A.J.: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (2002)
4. Zelnik-Manor, L., Perona, P.: Self-tuning Spectral Clustering. In: NIPS, pp. 1601–1608 (2004)
5. Bach, F.R., Jordan, M.I.: Learning Spectral Clustering, with Application to Speech Separation. JMLR 7, 1963–2001 (2006)
6. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, M.I.E., Jordan, M.I.: Learning the Kernel Matrix with Semidefinite Programming. JMLR 5, 27–72 (2004)
7. Valizadegan, H., Jin, R.: Generalized Maximum Margin Clustering and Unsupervised Kernel Learning. In: NIPS, pp. 1417–1424 (2007)
8. Lange, T., Buhmann, J.: Fusion of Similarity Data in Clustering. In: NIPS, pp. 723–730 (2005)
9. Ng, A., Jordan, M., Weiss, Y.: On Spectral Clustering: Analysis and an Algorithm. In: NIPS, pp. 849–856 (2001)
10. Yu, S.X., Shi, J.: Multiclass Spectral Clustering. In: ICCV, pp. 313–319 (2003)
11. Calamai, P.H., Moré, J.J.: Projected Gradients Methods for Linearly Constrained Problems. Math. Prog., 93–116 (1987)
12. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: More Efficiency in Multiple Kernel Learning. In: ICML, pp. 775–782 (2007)
13. Bonnans, J.F., Sharpiro, A.: Perturbation Analysis of Optimization Problems (2000)
14. Chapelle, O., Vanpnik, V., Bousquet, O., Mukherjee, S.: Choosing Multiple Parameters for Surport Vector Machines. Mach. Learn., 131–159 (2002)
15. Karypis, G.: CLUTO-A Clustering Toolkit (2002), `http://www-users.cs.umn.edu/~karypis/cluto`