Incorporating the Loss Function into Discriminative Clustering of Structured Outputs

Wenliang Zhong, Weike Pan, James T. Kwok, and Ivor W. Tsang

***Abstract*— Clustering using the Hilbert Schmidt independence criterion (CLUHSIC) is a recent clustering algorithm that maximizes the dependence between cluster labels and data observations according to the Hilbert Schmidt independence criterion (HSIC). It is unique in that structure information on the cluster outputs can be easily utilized in the clustering process. However, while the choice of the loss function is known to be very important in supervised learning with structured outputs, we will show in this paper that CLUHSIC is implicitly using the often inappropriate zero-one loss. We propose an extension called CLUHSICAL (which stands for "Clustering using HSIC and loss") which explicitly considers both the output dependency and loss function. Its optimization problem has the same form as CLUHSIC, except that its partition matrix is constructed in a different manner. Experimental results on a number of datasets with structured outputs show that CLUHSICAL often outperforms CLUHSIC in terms of both structured loss and clustering accuracy.**

***Index Terms*— Clustering methods, dependence Maximization, Hilbert Schmidt independence criterion, loss, structured clustering.**

I. Introduction

CLUSTERING aims at grouping examples that are similar to each other into a small number of classes or clusters. It is an invaluable data-analysis tool that has been widely used in diverse domains ranging from engineering, medical science, earth science, social science to economics [1]. Over the decades, a battery of clustering approaches has been developed. In general, these can be considered as representing three different views of clustering [2]. The first one is the geometric view, which includes the classic k-means clustering algorithm [3]. The second one is the statistical view, with examples including the method of information bottleneck [4] and mixture models [5]. The last one is the spectral view, which includes methods such as the normalized cut and various spectral clustering algorithms [6]. Clustering using the Hilbert Schmidt independence criterion (CLUHSIC) [2] is a recent clustering algorithm that unifies these different views of clustering. It finds the cluster partitioning such that the resultant cluster labels are maximally dependent on the data observations according to the Hilbert Schmidt independence criterion (HSIC) [7]. It can be shown that all the above views of clustering are special cases of CLUHSIC [2].

Manuscript received April 23, 2010; revised July 11, 2010; accepted July 13, 2010. Date of publication August 23, 2010; date of current version October 6, 2010. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, under Grant 614508.

W. Zhong, W. Pan, and J. T. Kwok are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China (e-mail: wzhong@cse.ust.hk; weikep@cse.ust.hk; jamesk@cse.ust.hk).

I. W. Tsang is with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: ivorisang@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2064177

CLUHSIC is also a kernel-based method that can be used on structured data. While traditional kernel methods are mainly focused on vectorial inputs and outputs, there is growing interest in extending them to more complex domains with structured data. This has been driven in part by the tremendous amount of structured data available online, such as Internet documents residing in a hierarchy and bioinformatics databases containing DNA sequences. In general, structure information may be present in the inputs and/or outputs. For structured inputs, a variety of kernels have been developed for strings, trees, and graphs [8]. Traditionally, kernel-based clustering algorithms focus only on using kernels defined in the input space [9], [10], On the other hand, CLUHSIC has the important advantage that kernels can be used on both the input and output (cluster labels). Thus, it can be used for clustering data into hierarchies, chains, or even graphs.

Recently, Xu *et al.* [11] proposed a related discriminative clustering algorithm for structured outputs. It is based on the idea of maximum margin clustering [12]–[14] that trains a support vector machine (SVM) by maximizing the margin and minimizing the loss over all possible cluster labels. However, maximum margin clustering is computationally much harder than maximum margin classification. Existing methods typically rely on reformulating and relaxing the non-convex optimization problem as semidefinite programs (SDPs) that are computationally expensive. To combat this problem, Xu *et al.* [11] proposed a heuristic procedure that avoids SDP entirely. However, as will be shown in Section IV, its empirical performance is not satisfactory. Very recently, Bach and Harchaoui [15] proposed a more efficient convex relaxation which can be solved as a sequence of singular value decompositions. However, this is designed only for vectorial data but not structured data. Moreover, loss functions are not used.

Back to the realm of supervised learning, it is well known that the popular zero-one loss function is often inappropriate for structured prediction [16], [17]. The joint kernel $k((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \varphi(\mathbf{x}, \mathbf{y}), \varphi(\mathbf{x}', \mathbf{y}') \rangle$ defined over both input features $\mathbf{x}$ and output labels $\mathbf{y}$ can be used to measure the similarity based on the input structure and the output label structure. However, even with a good joint kernel, prediction of the structured output may not always be correct due to

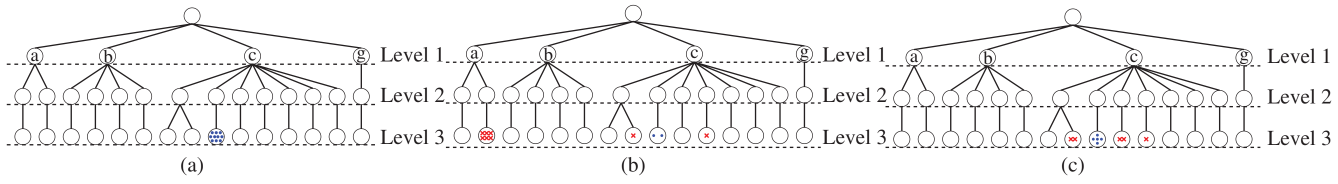


Fig. 1. Example showing that the misclassified patterns are often assigned to faraway clusters by CLUHSIC. A blue dot indicates a correct cluster assignment, while a red cross indicates an incorrect assignment. (a) Ground truth. (b) CLUHSIC. (c) CLUHSICAL.

insufficient or noisy training examples. Consider, for example, a document that belongs to the category "apple" in the hierarchical classification of text documents. Misassigning this document to a category (say, "orange") near the true label is much better than assigning it to a distant unrelated category (say, "moon"), although the losses in both cases are one according to the zero-one loss function. Hence, to measure the severity of different errors during training (or, in other words, to discourage patterns from being assigned to distant unrelated categories), the loss function also needs to utilize the structure information. In particular, one can define a loss function $\Delta(\mathbf{y}, \mathbf{u})$ that penalizes the difference between two outputs $\mathbf{y}$ and $\mathbf{u}$ according to the structure of the two objects. In general, there are many ways to define the loss function and each structured prediction problem may call for its own loss function. For example, in hierarchical classification, one can use the tree loss, which is defined as the height of the first common ancestor of the true and predicted labels in the hierarchy [17]. In natural language processing, the loss may be defined in terms of the underlying grammar.

The above observation also holds for clustering. Consider the example in Fig. 1, in which we show the cluster assignments for a subset of patterns that all belong to the same leaf node. Here, a blue dot indicates a correct cluster assignment while a red cross indicates an incorrect assignment. As shown in Fig. 1(a), ideally all these patterns should be clustered to the same leaf. Despite the fact that CLUHSIC is designed for clustering problems with structured outputs, it does not explicitly consider the use of a loss function. Indeed, as will be shown later in this paper, CLUHSIC essentially uses the often inappropriate zero-one loss function. Fig. 1(b) shows the clustering result of CLUHSIC. As can be seen, the misclassified patterns are often assigned to faraway clusters. In this paper, we extend CLUHSIC so that loss functions can be explicitly considered. Fig. 1(c) shows the clustering result of the proposed method CLUHSICAL, which will be described in more detail in later sections. As can be seen, although the obtained clustering is still imperfect as compared to the ground truth, the patterns are only misassigned to nearby categories.

The rest of this paper is organized as follows. Section II first gives a brief review on the CLUHSIC algorithm. Section III then proposes a clustering algorithm for structured outputs based on kernel ridge regression. Motivated by [11] and [15], we perform discriminative clustering by kernel ridge regression for structured data, whose optimization problem is simpler than that of the structural SVM [17]. In particular, we show that CLUHSIC can be considered as a special case with the use of a particular joint kernel and the zero-one loss. Clustering experiments on a number of datasets with structured outputs are presented in Section IV, and the last section gives some concluding remarks.

In the sequel, the transpose of vector/matrix is denoted by the superscript $'$, the identity matrix by $\mathbf{I}$, and the vector of all ones by $\mathbf{1}$. Moreover, $\text{tr}(\mathbf{A})$ denotes the trace of a matrix $\mathbf{A}$, $\|\mathbf{A}\|_p = \sup_{\mathbf{x}\neq\mathbf{0}} \|\mathbf{A}\mathbf{x}\|_p / \|\mathbf{x}\|_p$ is the matrix p-norm of $\mathbf{A}$, and $\circ$ the elementwise matrix multiplication.

II. CLUHSIC Algorithm

CLUHSIC [2] is a clustering algorithm based on the dependence maximization view. Given a sample $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$, the linear dependence between $\mathbf{x}_i$'s and $\mathbf{y}_i$'s can be easily estimated by simple statistics such as linear correlation. However, nonlinear dependencies are more difficult to measure. A recently proposed dependence (or, more precisely, independence) measure is the HSIC [7]. Specifically, let ϕ and λ be the feature maps on the input and output, respectively. Denote the corresponding reproducing kernel Hilbert space (RKHS) by $\mathcal{F}$ and $\mathcal{G}$, and the corresponding kernels by k and l. The cross-covariance operator $\mathcal{C}_{xy} : \mathcal{F} \mapsto \mathcal{G}$ is defined as [18] $\mathcal{C}_{xy} = \mathbf{E}_{\mathbf{xy}}\left[(\phi(\mathbf{x}) - \mathbf{E}[\phi(\mathbf{x})]) \otimes (\lambda(\mathbf{y}) - \mathbf{E}[\lambda(\mathbf{y})])\right]$ where $\otimes$ is the tensor product. HSIC is then defined as the square of the Hilbert-Schmidt norm $\|\cdot\|_{HS}$ of $\mathcal{C}_{xy}$

$$\begin{aligned}\text{HSIC}(\mathcal{F}, \mathcal{G}) &= \|\mathcal{C}_{xy}\|_{HS}^2 \\ &= \mathbf{E}_{\mathbf{xx}'\mathbf{yy}'}[k(\mathbf{x}, \mathbf{x}')l(\mathbf{y}, \mathbf{y}')] + \mathbf{E}_{\mathbf{xx}'}[k(\mathbf{x}, \mathbf{x}')]\mathbf{E}_{\mathbf{yy}'}[l(\mathbf{y}, \mathbf{y}')] \\ &\quad - 2\mathbf{E}_{\mathbf{xy}}[\mathbf{E}_{\mathbf{x}'}k(\mathbf{x}, \mathbf{x}')][\mathbf{E}_{\mathbf{y}'}l(\mathbf{y}, \mathbf{y}')].\end{aligned}$$

Given the sample $\mathcal{S}$, it can be shown that an empirical estimate of HSIC is

$$(m-1)^{-2}\text{tr}(\mathbf{HKHL}) \tag{1}$$

where $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)], \mathbf{L} = [l(\mathbf{y}_i, \mathbf{y}_j)]$ are kernel matrices defined on $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$, respectively, and $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{11}'$ is the so-called centering matrix. In the sequel, we assume that $\mathbf{K}$ is centered, and so (1) reduces to $(m-1)^{-2}\text{tr}(\mathbf{KL})$. Recent studies [7], [19] show that HSIC has several advantages over other independence measures. First, its empirical estimate in (1) is easy to compute. Moreover, it guarantees good uniform convergence and has very little bias even in high dimensions.

To use HSIC in clustering, one first defines a kernel matrix $\mathbf{A} \in \mathbb{R}^{c\times c}$ on a set of c clusters. This is used to model the prior structural relationships and correlation among clusters. In general, kernel entries for clusters that are structurally close to each other will be assigned high values. For example, if the user specifies that the resultant clusters should have a chain structure as in Fig. 2(a), one can use the output kernel

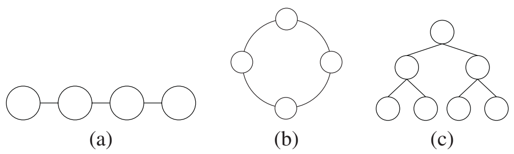


Fig. 2. Some common structures among clusters. (a) Chain. (b) Ring. (c) Tree.

matrix $\begin{bmatrix} 2&1&0&0\\ 1&2&1&0\\ 0&1&2&1\\ 0&0&1&2 \end{bmatrix}$. Here, the leftmost cluster corresponds to the first row/column of the kernel matrix, the second-left cluster corresponds to the second row/column of the kernel matrix, and so on. Similarly, for more complicated structures such as ring and tree[1][Fig. 2(b) and (c)], the corresponding kernel matrices are $\begin{bmatrix} 2&1&0&1\\ 1&2&1&0\\ 0&1&2&1\\ 1&0&1&2 \end{bmatrix}$ and $\begin{bmatrix} 2&1&0&0\\ 1&2&0&0\\ 0&0&2&1\\ 0&0&1&2 \end{bmatrix}$ respectively.

Let $\mathbf{\Pi}$ be a partition matrix such that its ith row specifies the assignment of the ith pattern to one of the c clusters, i.e., $\Pi_{ij} \in \{0, 1\}$ and $\mathbf{\Pi 1} = \mathbf{1}$. The kernel matrix $\mathbf{L}$ defined on the $\mathbf{y}_i$'s can then be written as $\mathbf{L} = \mathbf{\Pi A \Pi}'$. CLUHSIC aims at finding the cluster assignment $\mathbf{\Pi}$ such that the resultant $\mathbf{L}$ is maximally dependent on the kernel matrix $\mathbf{K}$ defined on the $\mathbf{x}_i$'s according to the HSIC. Mathematically, this is formulated as the following optimization problem:

$$\begin{aligned} \max_{\mathbf{\Pi}} \quad & \mathrm{tr}(\mathbf{K\Pi A\Pi}') \\ \text{s.t.} \quad & \mathbf{\Pi 1} = \mathbf{1},\ \Pi_{ij} \in \{0, 1\}. \end{aligned} \tag{2}$$

In [2], this integer programming problem is solved by greedy local search. Specifically, the partition matrix $\mathbf{\Pi}$ is updated row by row. For the ith row, one assigns the corresponding ith pattern to the cluster with the largest objective value. This process is re-iterated until the assignment matrix does not change. More sophisticated approaches, such as spectral methods [20] and nonnegative matrix factorization [21], are also discussed in [19].

As discussed in [2], CLUHSIC is very flexible as one can use different kernels of $\mathbf{K}$ and $\mathbf{A}$ in (2). For example, on setting $\mathbf{A} = \mathbf{I}$, one recovers standard (kernel) k-means, which implicitly assumes that there is no relationship between the clusters. Alternatively, on setting

$$\mathbf{A} = \mathrm{diag}\left(\left[\sum_{i\in\mathbf{\Pi}_{\cdot 1}} w_i, \sum_{i\in\mathbf{\Pi}_{\cdot 2}} w_i, \ldots, \sum_{i\in\mathbf{\Pi}_{\cdot k}} w_k\right]\right)$$

one recovers the weighted k-means where the ith pattern is associated with a weight w_i representing its importance. In general, CLUHSIC can be considered as a family of clustering algorithms that unify the traditional geometric, spectral, and statistical dependence views of clustering. Interested readers are referred to [2] for further details.

III. Clustering for Structured Outputs

In Section III-A, we first consider the easier setting of supervised learning for structured outputs. Instead of using the SVM as in [16], [17], we consider a related method called kernel ridge regression (KRR) [22], which is also known as the least squares SVM [23]. KRR uses the square loss instead of the hinge loss, and leads to a simpler optimization problem than the SVM. This is then extended to the unsupervised learning setting in Section III-B.

A. Kernel Ridge Regression for Structured Outputs

For learning with structured outputs (such as sequences, trees, or graphs), it is often more convenient to use a joint feature representation φ that is defined on both the input $\mathcal{X}$ and output $\mathcal{Y}$ [17]. This allows the many-sided dependencies between $\mathcal{X}$ and $\mathcal{Y}$ to be captured. As in other kernel methods, this joint feature map is related to a *joint kernel* k as $k((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \varphi(\mathbf{x}, \mathbf{y}), \varphi(\mathbf{x}', \mathbf{y}')\rangle$. Given a set of training patterns $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, KRR then obtains the classification function $\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y})\rangle$ by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\xi_{i\mathbf{y}}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i,\mathbf{y}\neq\mathbf{y}_i} \xi_{i\mathbf{y}}^2 \\ \text{s.t.} \quad & \langle \mathbf{w}, \boldsymbol{\delta\varphi}_i(\mathbf{y})\rangle = 1 - \frac{\xi_{i\mathbf{y}}}{\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}}, \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}_i \end{aligned} \tag{3}$$

where $\xi_{i\mathbf{y}}$'s are slack variables for the errors, $\boldsymbol{\delta\varphi}_i(\mathbf{y}) \equiv \varphi(\mathbf{x}_i, \mathbf{y}_i) - \varphi(\mathbf{x}_i, \mathbf{y})$ is the feature map for structured prediction, $\Delta(\mathbf{y}_i, \mathbf{y})$ is a loss function penalizing the difference between $\mathbf{y}_i$ and $\mathbf{y}$, and C is a user-defined regularization parameter. Its main difference with the SVMs primal is that we now have equality constraints instead of inequality constraints. Moreover, note that the slack variables in (3) are scaled with the inverse loss, and this is often called *slack re-scaling*. Another approach, called *margin re-scaling* [16], scales the margin by the loss. In this paper, we will focus on slack re-scaling, and extension to margin re-scaling is straightforward. Besides, unlike the SVM-struct [17] that uses only one slack variable for each training instance, here we use one slack variable for each $(i, \mathbf{y})$ pair. The following proposition shows that this leads to an efficient optimization problem.

Proposition 1: The dual of (3) can be written as

$$\max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\mathbf{Q}\boldsymbol{\alpha} \tag{4}$$

where $\boldsymbol{\alpha} = [\alpha_{i\mathbf{y}}]$ is the vector of Lagrange multipliers, $\mathbf{Q} = [Q_{i\mathbf{y},j\bar{\mathbf{y}}}]$ is a positive definite matrix with entries

$$Q_{i\mathbf{y},j\bar{\mathbf{y}}} = \langle \boldsymbol{\delta\varphi}_i(\mathbf{y}), \boldsymbol{\delta\varphi}_j(\bar{\mathbf{y}})\rangle + \frac{m}{2C}\frac{\delta_{i\mathbf{y}j\bar{\mathbf{y}}}}{\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}\sqrt{\Delta(\mathbf{y}_j, \bar{\mathbf{y}})}}.$$

The proof is in Appendix I. Later on, it will be more convenient to write $\mathbf{Q}$ as $\mathbf{J} + \mathbf{S}$, where

$$\mathbf{J} = [\langle \boldsymbol{\delta\varphi}_i(\mathbf{y}), \boldsymbol{\delta\varphi}_j(\bar{\mathbf{y}})\rangle] \tag{5}$$

and $\mathbf{S}$ is a diagonal matrix with the $(i\mathbf{y})$th entry $(\forall i, \mathbf{y} \neq \mathbf{y}_i)$

$$S_{i\mathbf{y}} = \frac{m}{2C}\frac{1}{\Delta(\mathbf{y}_i, \mathbf{y})}. \tag{6}$$

Since the objective function in (4) is a quadratic function with variable $\boldsymbol{\alpha}$, it is an unconstrained quadratic programming (QP) problem. Its optimum $\boldsymbol{\alpha}$ can be easily obtained as $\boldsymbol{\alpha}^* = (\mathbf{J} + \mathbf{S})^{-1}\mathbf{1}$. Plugging this back in (4), the corresponding optimal

[1]Note that for the tree structure in Fig. 2(c), only the four leaves are the observed clusters. The internal nodes and root node are latent and so not involved in defining the output kernel matrix.

primal/dual objective value (which are equal as convex QPs have zero duality gap) is then equal to

$$\frac{1}{2}\mathbf{1}'(\mathbf{J}+\mathbf{S})^{-1}\mathbf{1}. \tag{7}$$

In general, there can be different ways of defining the joint kernel and the corresponding joint feature map φ. In this paper, we focus on the construction via the tensor product

$$\varphi(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x}) \otimes \lambda(\mathbf{y}) \tag{8}$$

where ϕ and λ are feature maps defined on $\mathcal{X}$ and $\mathcal{Y}$, respectively. This feature map has been popularly used in classification with taxonomies [17]. Moreover, it can be shown that its joint kernel $\langle \varphi(\mathbf{x},\mathbf{y}), \varphi(\bar{\mathbf{x}},\bar{\mathbf{y}})\rangle$ is simply a product of the corresponding input kernel $\kappa(\mathbf{x},\bar{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\bar{\mathbf{x}})\rangle$ and output kernel $\Lambda(\mathbf{y},\bar{\mathbf{y}}) = \langle \lambda(\mathbf{y}), \lambda(\bar{\mathbf{y}})\rangle$ [17].

B. CLUHSICAL Algorithm

While the training outputs $\mathbf{y}_i$'s are known in classification, they are missing in clustering. Hence, as in maximum margin clustering [13], we want to find $\{\mathbf{y}_1,\ldots,\mathbf{y}_m\}$ such that the objective in (7) is minimized

$$\min_{\mathbf{y}_1,\ldots,\mathbf{y}_m} \mathbf{1}'(\mathbf{J}+\mathbf{S})^{-1}\mathbf{1}. \tag{9}$$

However, because of the rather complicated definition of $\mathbf{J}+\mathbf{S}$, a direct minimization of $\mathbf{1}'(\mathbf{J}+\mathbf{S})^{-1}\mathbf{1}$ is difficult. With the use of the tensor-product feature map in (8), the following proposition shows that this optimization problem can be simplified by using first-order approximation (the proof is in Appendix II).

Proposition 2: Suppose that the tensor-product feature map in (8) is used, and the corresponding joint kernel $k(\cdot,\cdot)$ is scaled by a factor $\gamma > 0$ to $\gamma k(\cdot,\cdot)$, where

$$\gamma < \frac{1}{\|\mathbf{S}^{-1}\|_p \|\mathbf{J}\|_p} \tag{10}$$

w.r.t. some matrix p-norm $\|\cdot\|_p$. Then, as a first-order approximation, (9) can be formulated as

$$\begin{aligned} \max_{\widetilde{\mathbf{\Pi}}} \quad & \tilde{C}\,\mathrm{tr}(\mathbf{K}\widetilde{\mathbf{\Pi}}\mathbf{A}\widetilde{\mathbf{\Pi}}') - \frac{C}{m}\|\widetilde{\mathbf{\Pi}}\|_1 \\ \text{s.t} \quad & \widetilde{\mathbf{\Pi}}_{i\cdot} \text{ is as defined in (12)} \end{aligned} \tag{11}$$

where $\mathbf{K} = [\kappa(\mathbf{x}_i,\mathbf{x}_j)] \in \mathbb{R}^{m\times m}$ is the kernel matrix defined on the input $\mathbf{A} = [\Lambda(\mathbf{y}_i,\mathbf{y}_j)] \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ is the kernel matrix defined on the output $\tilde{C} = 4C^2\gamma/m^2$, and $\widetilde{\mathbf{\Pi}}$ is a matrix such that its ith row (corresponding to pattern i) is defined as

$$\begin{aligned} \widetilde{\mathbf{\Pi}}_{i\cdot} = \Big[& -\Delta(\mathbf{y}_i,\mathbf{y}^1), \cdots -\Delta(\mathbf{y}_i,\mathbf{y}^{\ell-1}), \sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta(\mathbf{y}_i,\mathbf{y}), \\ & -\Delta(\mathbf{y}_i,\mathbf{y}^{\ell+1}), \cdots -\Delta(\mathbf{y}_i,\mathbf{y}^{|\mathcal{Y}|})\Big] \end{aligned} \tag{12}$$

where $\mathcal{Y} = \{\mathbf{y}^1,\ldots,\mathbf{y}^{|\mathcal{Y}|}\}$ and output $\mathbf{y}_i$ of the ith pattern is equal to some $\mathbf{y}^\ell \in \mathcal{Y}$.

Problem (11) involves an additional scaling parameter γ (hidden in the parameter $\tilde{C}$), and this may appear cumbersome in practical applications. Interestingly, it will be shown in the following that γ can be removed from the optimization with a proper normalization of the partition matrix $\widetilde{\mathbf{\Pi}}$. In CLUHSIC [2], each column of $\mathbf{\Pi}$ is often normalized to have unit ℓ_2-norm [i.e., $\|\mathbf{\Pi}_{\cdot j}\|_2 = 1$] before aligning $\mathbf{\Pi}'\mathbf{K}\mathbf{\Pi}$ with $\mathbf{A}$. This ensures that the solution is not biased toward clusters having more samples. A similar normalization can also be performed on the $\widetilde{\mathbf{\Pi}}$ in (11). However, here, it is more convenient to normalize each column of $\widetilde{\mathbf{\Pi}}$ with the ℓ_1-norm, because $\|\widetilde{\mathbf{\Pi}}\|_1$ in (11) is then a constant and can be dropped, along with the associated $\tilde{C}$ parameter (and thus also the γ parameter). Denote the ℓ_1-normalized $\widetilde{\mathbf{\Pi}}$ by $\bar{\mathbf{\Pi}}$. The optimization problem (11) then simplifies to

$$\max_{\bar{\mathbf{\Pi}}} \mathrm{tr}(\mathbf{K}\bar{\mathbf{\Pi}}\mathbf{A}\bar{\mathbf{\Pi}}'). \tag{13}$$

This is of the same form as CLUHSIC, except that the loss function is now explicitly considered in $\widetilde{\mathbf{\Pi}}$ (12). In the sequel, this will be called "CLUstering using HSIC And Loss" (CLUHSICAL).

C. Relationship with CLUHSIC

Unlike the CLUHSIC algorithm, CLUHSICAL allows the explicit specification of a loss function. In particular, when the zero-one loss is used (i.e., $\Delta(\mathbf{y}_i,\mathbf{y}_j) = 1$ for $\mathbf{y}_i \neq \mathbf{y}_j$), the following corollary shows that the unnormalized version of CLUHSIC and CLUHSICAL in (11) will yield the same clustering solution. The proof is in Appendix III.

Corollary 1: With the zero-one loss, (11) without normalization reduces to CLUHSIC.

While CLUHSIC and CLUHSICAL are equivalent under the zero-one loss before normalization (which is also verified experimentally), they can have different results after normalization. Recall that the main difference between CLUHSIC and CLUHSICAL lies in the definition of the partition matrix. In the following, we illustrate that the partition matrix in (12) is more advantageous than the one in CLUHSIC by studying the simplest clustering algorithm in the CLUHSIC family, namely (kernel) k-means clustering. Experimental results in Section IV-C also show that CLUHSICAL often performs better than CLUHSIC even when both are trained with the zero-one loss.

Consider the objective

$$\sum_{c=1}^{|\mathcal{Y}|}\sum_{i=1}^{m} |\mathbf{\Pi}_{ic}|\,\|\mathrm{sgn}(\mathbf{\Pi}_{ic})\phi(\mathbf{x}_i) - \boldsymbol{\mu}_c\|_{\mathcal{F}}^2 \tag{14}$$

where $\mathbf{\Pi}$ is an unnormalized partition matrix and $\mathcal{F}$ is the feature space induced by the kernel k. Let m_c be the number of patterns belonging to the cth cluster and assume that all clusters are of the same size, i.e., $m_c = m/|\mathcal{Y}|$. The following lemmas show that the optimization problems of the ℓ_1-normalized CLUHSIC and CLUHSICAL are of the form in (14) (proofs are in Appendix III).

Lemma 1: The optimization problem of the ℓ_1-normalized CLUHSIC with $\mathbf{A} = (m/|\mathcal{Y}|)\mathbf{I}$ is of the form (14) with

$$\mathbf{\Pi}_{ic} = \begin{cases} 1 & \phi(\mathbf{x}_i) \in c, \\ 0 & \phi(\mathbf{x}_i) \notin c \end{cases} \tag{15}$$

and the corresponding centers can be obtained as

$$\boldsymbol{\mu}_c = \frac{1}{m_c} \sum_{\phi(\mathbf{x}_i)\in c} \phi(\mathbf{x}_i), \quad c = 1, \ldots, |\mathcal{Y}|. \tag{16}$$

Lemma 2: The optimization problem of the ℓ_1-normalized CLUHSICAL with $\mathbf{A} = ((2|\mathcal{Y}| - 1)m/|\mathcal{Y}|)\mathbf{I}$ is of the form (14) with

$$\boldsymbol{\Pi}_{ic} = \begin{cases} |\mathcal{Y}| - 1 & \phi(\mathbf{x}_i) \in c, \\ -1 & \phi(\mathbf{x}_i) \notin c \end{cases} \tag{17}$$

and the corresponding centers can be obtained as

$$\boldsymbol{\mu}_c = \sum_{i=1}^{m} \frac{\boldsymbol{\Pi}_{ic}}{\sum_{j=1}^{m} |\boldsymbol{\Pi}_{jc}|} \phi(\mathbf{x}_i), \quad c = 1, \ldots, |\mathcal{Y}|. \tag{18}$$

Note that the $\mathbf{A}$s in the two lemmas differ only by the constant $2(|\mathcal{Y}| - 1)$, which is immaterial. Hence, we can examine the difference between the ℓ_1-normalized versions of CLUHSIC and CLUHSICAL by considering the two different settings of $\boldsymbol{\Pi}_{ic}$'s in (14). Note that, for CLUHSIC, $\boldsymbol{\Pi}_{ic}$ is nonzero only when $\phi(\mathbf{x}_i)$ is in cluster c. On the other hand, for CLUHSICAL, $\boldsymbol{\Pi}_{ic}$ in (17) is positive when $\phi(\mathbf{x}_i)$ belongs to cluster c, and negative otherwise. Hence, from (14), $\boldsymbol{\mu}_c$ will be drawn close to $\phi(\mathbf{x}_i)$ if $\phi(\mathbf{x}_i)$ belongs to cluster c, otherwise, $\boldsymbol{\mu}_c$ will be pushed away from $\phi(\mathbf{x}_i)$. This thus resembles the effect of a margin and is also similar to the rival penalized competitive learning algorithm [24], which has better performance than standard k-means clustering.

IV. Experiments

In this section, we perform experiments on a number of datasets, whose outputs have the structure of a hierarchy (Section IV-A) or a ring (Section IV-B). All these patterns have been manually grouped into different categories, and these will be used as the ground truth cluster labels in the performance evaluation.

A. Hierarchy-Structured Datasets

1) Data Sets and Input Kernels: Structural classification of proteins (SCOP) data: The SCOP database [25] provides a detailed description of the structural relationships of all the known proteins. Here, we use the subset extracted in [26] with 15 superfamilies. Ten proteins are selected for each superfamily, leading to a total of 150 patterns. The classification of the proteins is based on a hierarchical structure. The lowest level consists of 15 superfamilies, the next higher level is the fold, then the class, and finally the root [Fig. 3(a)]. We use the three alignment kernels ($\mathbf{K}_1^{Al}$, $\mathbf{K}_2^{Al}$, and $\mathbf{K}_3^{Al}$) proposed in [26] as input kernels. These alignment kernels are inspired from the convolution kernel in [27], and are constructed based on kernels defined on the protein's constituent substructures.

World Intellectual Property Organization (WIPO) data: This is a set of patent documents from the WIPO and has been popularly used [17], [28], [29]. We use the six largest categories in Section D (each category contains at least 50 documents), with a total of 354 documents. The corresponding hierarchical structure is shown in Fig. 3(b). The linear kernel,

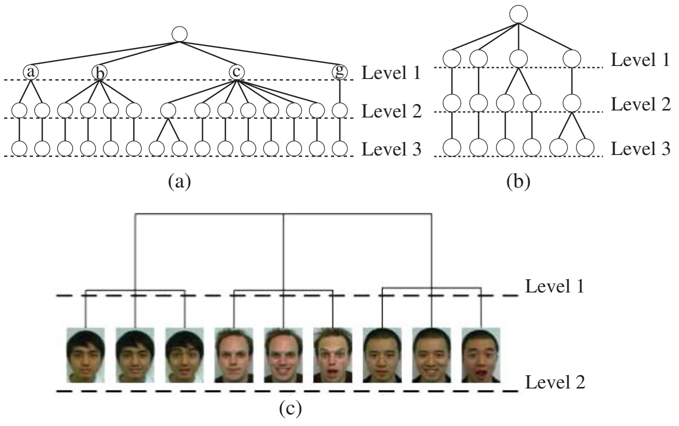


Fig. 3. Hierarchically structured datasets. (a) SCOP. (b) WIPO. (c) Facial expression.

which is often appropriate for text classification, is used on the input.

Facial expression data: This data set[2] has been used in the CLUHSIC paper [2]. It consists of 185 images (each of size 217 × 308) of three types of facial expressions from three subjects. The facial expressions of the same person are first grouped together in the hierarchy [Fig. 3(c)]. As in [2], each pixel of the face image is normalized in each dimension and the Gaussian kernel is used as the input kernel.

2) Experimental Setup: We follow [17] to construct the output feature map for these hierarchically structured data sets. Let $\mathcal{Z}$ be the set of nodes in the hierarchy, and let the hierarchy structure be represented by the partial order $\prec$, where $\mathbf{z} \prec \mathbf{y}$ means that node $\mathbf{z}$ is an ancestor of node $\mathbf{y}$. A feature $\lambda_{\mathbf{z}}$ is then defined with every node $\mathbf{z}$, as $\lambda_{\mathbf{z}}(\mathbf{y}) = \begin{cases} 1 & \mathbf{z} \prec \mathbf{y} \text{ or } \mathbf{z} = \mathbf{y} \\ 0 & \text{otherwise} \end{cases}$. By excluding the feature associated with the root node (which is shared by all the classes), it can be easily seen that this feature map is also the same as the one used in [2].

We will compare CLUHSICAL with two existing clustering algorithms for structured data.

1) CLUHSIC in [2].
2) Discriminative unsupervised training algorithm (labeled "XWSS" in the sequel) in [11]. Since XWSS is based on SDP and so can only be used on very small datasets, we will adopt the heuristic iterative procedure proposed in [11].

The optimization of both CLUHSIC and CLUHSICAL involve integer programming. Since the focus of this paper is not on how to solve this integer program, we will simply follow CLUHSIC in [2] and obtain an approximate solution by greedy local optimization. Its time complexity per iteration is $O(mk)$, and the procedure often converges in fewer than 20 iterations. More sophisticated approaches, such as spectral methods [20] and nonnegative matrix factorization [21] are also discussed in [19].

For further benchmarking, we also report results of kernelized versions (using the same kernel as CLUHSICAL) of several agglomerative hierarchical clustering methods [30], including the following.

[2]http://www.it.usyd.edu.au/~/lesong/cluhsic_datasets.html

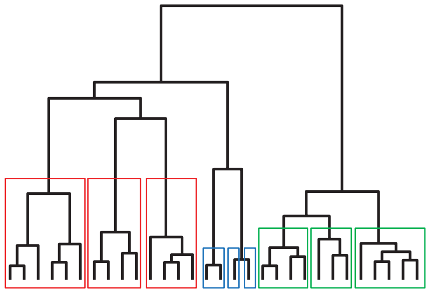


Fig. 4. Example of the dendrogram and clustering result obtained on the facial expression data. Each color corresponds to a level-1 cluster, and each rectangle is a level-2 cluster. Since the dataset has 185 samples, the whole dendrogram has 185 leaves. To reduce clutter, we only show the top 30 clusters.

1) Average linkage, which defines the dissimilarity between two clusters A and B as $D(A,B) = (1/n_A n_B)\sum_{i\in A, j\in B} d(a_i, b_j)$. Here, n_A and n_B are the numbers of instances in clusters A and B, respectively, and $d(a_i, b_j)$ is the distance (induced by the same kernel used in CLUHSICAL) between instances a_i and b_j.
2) Complete linkage, with $D(A,B) = \max_{i\in A, j\in B} d(a_i, b_j)$.
3) Ward's linkage [31], with $D(A,B) = \sqrt{\frac{n_A n_B}{n_A+n_B} d(A_m, B_m)^2}$, where A_m, B_M are the cluster means of A and B, respectively,

and hierarchical k-means algorithm. Since standard agglomerative hierarchical clustering and k-means do not utilize structure information, we implement a structure-sensitive variant as follows. Take as an example the facial expression data [Fig. 3(c)], which has three level-1 clusters and three level-2 sub-clusters under each cluster. For hierarchical clustering, we first run the standard algorithm and obtain the complete dendrogram. By going down the root using the dendrogram, we can obtain the three level-1 clusters. For each of these, we further trace down the corresponding branch of the dendrogram and obtain the three sub-clusters (Fig. 4). For the more complex hierarchies in Fig. 3(a) and (b), we align the obtained clusters so that the one with the largest variance corresponds to the tree node having the largest subtree. The hierarchical k-means procedure is similar, except that instead of going down the dendrogram, we perform kernel k-means to divide a node into several sub-clusters.

In general, since hierarchical clustering does not utilize structure information in constructing the dendrogram, there is no guarantee that the variants used in our experiments (average linkage, complete linkage, and Ward's linkage) can always produce clustering results that conform to the given structure. Nevertheless, empirically, they are able to do so in our experiments.

For performance evaluation, we will use the accuracy and tree loss, which is defined as the height of the first common ancestor of the true and predicted labels in the hierarchy [17]. Note that the tree loss is the same as the zero-one loss when measured at the first level of the hierarchy, and so will not be reported in the sequel. Moreover, since the positions of some of the clusters can be permuted [e.g., the leaves under each level-1 cluster in Fig. 3, and the three level-1 clusters in Fig. 3(c)], we will report the performance based on the best permutation of the clustering result.

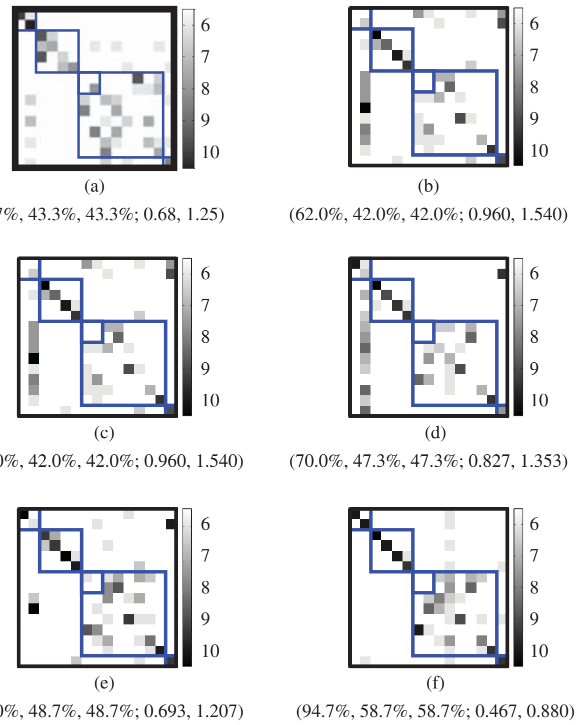


Fig. 5. Confusion matrices on the SCOP data for a typical run. Shown inside the brackets are the accuracies at the first, second, and third levels, and the tree losses at the second and third levels. (a) Complete-linkage. (b) Kernel hierarchical k-means. (c) XWSS. (d) CLUHSIC. (e) CLUHSICAL (0–1 loss). (f) CLUHSICAL (tree loss).

The kernel hierarchical k-means clustering result is used to initialize CLUHSICAL, CLUHSIC, and XWSS (except for the teapot data where kernel hierarchical k-means produces a very poor initialization and so random initialization is used instead). To reduce statistical variability, results for all the methods (except for agglomerative hierarchical clustering, which is deterministic) are averaged over 10 repetitions with 10 random restarts in each repetition. In the ℓ_2-normalized version of CLUHSICAL, we fix $C = 0.1m$ and $\gamma = 1$.

3) Importance of the Loss Function: We first illustrate the importance of the loss function. Fig. 5 shows the confusion matrices obtained on the SCOP data for a typical run.

The four squares lying along the diagonal correspond to the four subtrees rooted at nodes a, b, c, and g in Fig. 3(a). The small square inside the largest square corresponds to the leftmost subtree under node c. The gray level represents the number of patterns in the cluster (darker color indicates more patterns). In the ideal case, the confusion matrix is diagonal and all the patterns are correctly clustered, resulting in zero error and zero loss. As can be seen from Fig. 5, while all the methods are not perfect, CLUHSICAL, when trained with the tree loss, assigns the mis-clustered patterns to clusters that are still within the target subtree. On the other hand, the other methods may assign the mis-clustered patterns to distant unrelated categories.

Fig. 6 illustrates this effect in more detail. For clarity, we only show the assignments for patterns belonging to a particular leaf node. Here, a (blue) dot indicates a correct

TABLE I
CLUSTERING PERFORMANCE ON THE SCOP DATA

kernel matrix	method		accuracy (%) level 1	accuracy (%) level 2	accuracy (%) level 3	tree loss level 2	tree loss level 3
$\mathbf{K}_1^{Al}$	hierarchical clustering	average-linkage	75.3	34.7	32.7	0.90	1.57
		complete-linkage	**88.7**	43.3	43.3	0.68	**1.25**
		Ward's linkage	55.3	32.7	30.7	1.12	1.81
	kernel hierarchical k-means		55.2	35.2	35.2	1.10	1.74
	XWSS		55.2	35.2	35.2	1.10	1.74
	CLUHSIC		40.9	31.9	31.9	1.27	1.95
	CLUHSICAL	0-1 loss	71.0	**47.7**	**47.6**	0.81	1.34
	(ℓ_1-normalized)	tree loss	**84.9**	**54.1**	**53.9**	**0.61**	**1.07**
	CLUHSICAL	0-1 loss	65.9	44.4	44.1	0.90	1.46
	(ℓ_2-normalized)	tree loss	**86.6**	**53.7**	**53.7**	**0.60**	**1.06**
$\mathbf{K}_2^{Al}$	hierarchical clustering	average-linkage	54.7	14.0	12.0	1.31	2.19
		complete-linkage	54.0	12.0	10.0	1.34	2.24
		Ward's linkage	72.0	41.3	38.0	0.87	1.49
	kernel hierarchical k-means		51.9	32.1	32.1	1.17	1.84
	XWSS		51.8	32.1	32.1	1.17	1.84
	CLUHSIC		50.1	29.6	29.6	1.20	1.91
	CLUHSICAL	0-1 loss	69.1	39.0	39.0	0.92	1.53
	(ℓ_1-normalized)	tree loss	**80.0**	**45.5**	**45.3**	**0.75**	**1.29**
	CLUHSICAL	0-1 loss	66.7	**39.3**	**38.5**	0.94	1.56
	(ℓ_2-normalized)	tree loss	69.4	**40.1**	**39.6**	0.91	1.51
$\mathbf{K}_3^{Al}$	hierarchical clustering	average-linkage	**74.0**	29.3	25.3	0.97	1.71
		complete-linkage	51.3	28.0	28.0	1.21	1.93
		Ward's linkage	64.7	34.7	32.7	1.01	1.68
	kernel hierarchical k-means		53.2	35.7	35.7	1.12	1.76
	XWSS		53.2	35.7	35.7	1.12	1.76
	CLUHSIC		36.6	25.5	25.5	1.38	2.12
	CLUHSICAL	0-1 loss	58.6	38.0	38.0	1.03	1.65
	(ℓ_1-normalized)	tree loss	**71.3**	43.3	43.2	0.86	1.42
	CLUHSICAL	0-1 loss	**75.5**	**51.3**	**50.9**	**0.73**	**1.22**
	(ℓ_2-normalized)	tree loss	**76.9**	47.5	47.1	**0.76**	**1.29**

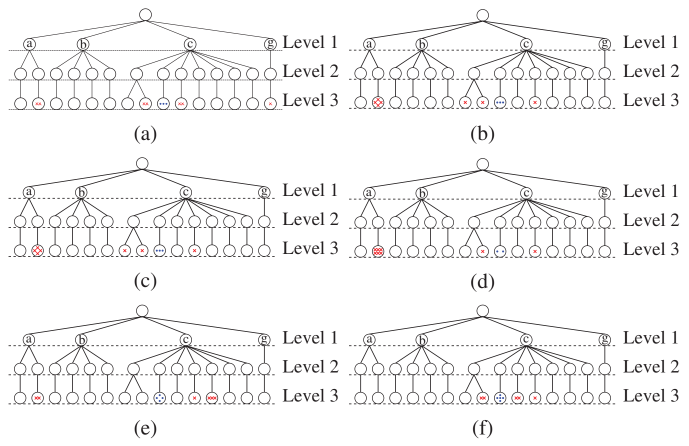


Fig. 6. Cluster assignments for patterns coming from a specific cluster. (a)–(f) Results obtained by agglomerative hierarchical clustering with complete-linkage, kernel hierarchical k-means, XWSS, CLUHSIC, CLUHSICAL (0–1 loss) and CLUHSICAL (tree loss), respectively.

cluster assignment while a (red) cross indicates an incorrect assignment. As can be seen, CLUHSICAL, when used with the tree loss, tries to avoid assigning mis-clustered patterns to faraway clusters. Hence, incorporation of the loss function is beneficial in clustering structured data.

4) Clustering Performance: Detailed clustering performance results on the various datasets are shown in Tables I–III. The best results and those that are not significantly worse (according to the t-test with a p-value less than 0.05) are printed in bold. As can be seen, CLUHSICAL trained with the structured loss performs well on the SCOP and WIPO datasets, and is comparable to the best result (attained by CLUHSICAL trained with the 0-1 loss) on the facial expression data. In particular, CLUHSICAL performs much better than kernel hierarchical k-means. This is because our kernel hierarchical k-means variant can only consider the sub-structure information separately, but not as a whole, whereas CLUHSICAL considers both the output structure and loss function in a more principled way. Results of the various hierarchical clustering variants are also inferior because the structure information cannot be utilized while constructing the dendrogram. Besides, both versions of CLUHSICAL are often better than CLUHSIC. CLUHSICAL trained with the structured loss is often better than that trained with the zero-one loss. Moreover, as discussed in Section III-C, CLUHSICAL often performs better than CLUHSIC even when both are trained with the zero-one loss. The ℓ_2-normalized version also outperforms CLUHSIC, but is sometimes slightly inferior to its ℓ_1-normalized version.

B. Ring-Structured Dataset

The datasets used in the previous section are all hierarchical in structure. In this section, we will experiment with the teapot data set[3] [2], [32] which has a ring structure. It contains 400 teapot images (each of size 76×101) rotated from 1°–360°. The Gaussian kernel is used on the input. As for the output, we follow [2] and set $A(\mathbf{y}_i, \mathbf{y}_j)$ to 2 if $i = j$, 1 if i and j are neighbors, and 0 otherwise.

In the first experiment, we use (standard) kernel k-means, CLUHSIC, and CLUHSICAL to cluster the full dataset into 10

[3]http://www.it.usyd.edu.au/~/lesong/cluhsic_datasets.html

TABLE II

CLUSTERING PERFORMANCE ON THE WIPO DATA

method		accuracy (%)			tree loss	
		lvl 1	lvl 2	lvl 3	lvl 2	lvl 3
hierarchical clustering	average	**32.5**	18.4	18.4	1.49	2.31
	complete	24.9	24.9	20.3	1.50	2.30
	Ward's	27.1	27.1	22.9	1.46	2.23
kernel hierarchical k-means		26.0	23.5	20.9	1.51	2.30
XWSS		26.0	23.5	20.9	1.51	2.30
CLUHSIC		26.4	24.7	21.8	1.49	2.27
CLUHSICAL (ℓ_1-normalized)	0-1 loss	30.6	27.0	**25.7**	1.42	2.17
	tree loss	**33.0**	**29.6**	**27.3**	**1.37**	**2.10**
CLUHSICAL (ℓ_2-normalized)	0-1 loss	31.4	27.7	**26.1**	1.41	2.15
	tree loss	**35.1**	**32.1**	**29.3**	**1.33**	**2.04**

TABLE III

CLUSTERING PERFORMANCE ON THE FACIAL EXPRESSION DATA

method		accuracy (%)		tree loss
		level 1	level 2	level 2
hierarchical clustering	average	78.4	42.7	0.79
	complete	78.4	42.7	0.79
	Ward's	88.7	71.9	0.39
kernel hierarchical k-means		70.1	54.6	0.75
XWSS		70.1	55.0	0.75
CLUHSIC		89.0	80.8	0.30
CLUHSICAL (ℓ_1-normalized)	0-1 loss	**96.5**	**94.8**	**0.09**
	tree loss	**92.3**	**90.2**	**0.18**
CLUHSICAL (ℓ_2-normalized)	0-1 loss	**94.3**	**90.8**	**0.15**
	tree loss	**93.4**	**91.8**	0.15

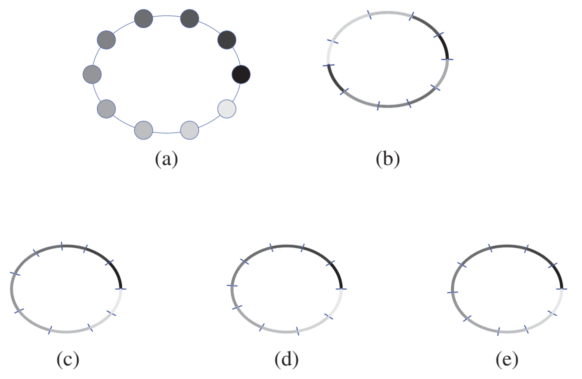


Fig. 7. Clusters (separated by strokes) obtained on the full teapot dataset in a typical run. Note that the ordering of the grayscales in the k-means solution does not align with those of the ground truth. (a) Ring structure. (b) Kernel k-means. (c) CLUHSIC. (d) CLUHSICAL (0–1 loss). (e) CLUHSICAL (tree loss).

clusters. Fig. 7 shows the clusters obtained for a typical run. Here, the positions of the clusters on the chain are color-coded by the grayscale [Fig. 7(a)]. As can be seen, all the methods can group the patterns into reasonable clusters. However, the ordering of the clusters in the kernel k-means solution does not align with the given ring structure. This is not surprising as structure information is not used in kernel k-means. As for the solutions produced by CLUHSIC and CLUHSICAL, they are very similar and differ only in the slight variations for the sizes of the clusters obtained.

However, a quantitative comparison is difficult. Recall that the images are obtained by continuously rotating the teapot from 1°–360°. Hence, there is no natural ground truth on the number of clusters and the size of each cluster. Even if these were known, it would still be difficult to have a fair performance comparison because of the continuous ring structure.

TABLE IV

CLUSTERING PERFORMANCE ON THE TEAPOT SUBSET

method		accuracy (%)	ring loss
kernel k-means		32.32	1.16
XWSS [11]		13.06	1.54
CLUHSIC [2]		65.51	0.39
CLUHSICAL (ℓ_1-normalized)	0-1 loss	**93.83**	**0.06**
	ring loss	**98.20**	**0.02**
CLUHSICAL (ℓ_2-normalized)	0-1 loss	55.29	0.85
	ring loss	54.71	0.85

TABLE V

CLUSTERING PERFORMANCE IN THE SPECIAL CASE OF $\mathbf{A} = \mathbf{I}$. NUMBER IN BRACKETS IS THE STANDARD DEVIATION

data set	method	accuracy (%)
SCOP ($\mathbf{K}_1^{Al}$)	CLUHSIC	66.4 (3.6)
	CLUHSICAL	**72.8 (4.6)**
SCOP ($\mathbf{K}_2^{Al}$)	CLUHSIC	64.3 (3.7)
	CLUHSICAL	**69.4 (4.1)**
SCOP ($\mathbf{K}_3^{Al}$)	CLUHSIC	66.3 (4.3)
	CLUHSICAL	**71.7 (4.1)**
WIPO	CLUHSIC	43.0 (4.4)
	CLUHSICAL	**51.5 (4.3)**
face	CLUHSIC	77.0 (7.6)
	CLUHSICAL	**90.9 (5.9)**
teapot	CLUHSIC	32.7 (5.3)
	CLUHSICAL	**35.1 (9.4)**

To avoid this problem, we perform a second experiment in which 5 images are removed from every 40 images. Thus, in this ground truth solution, we have a total of 10 well-separated clusters each with 35 images. Besides reporting the zero-one loss, we will also show the ring loss, which is defined as zero if the predicted position is correct, 1 if the predicted position and ground truth position are neighbors, and 2 otherwise. Moreover, as in Section IV-A, the positions of the clusters can be permuted[4], and so the reported performance will be based on the best permutation. Results are shown in Table IV. Again, CLUHSICAL trained with the structured loss performs best on both metrics.

C. Special Case of $\mathbf{A} = \mathbf{I}$ and Zero-One Loss

In this section, we consider the special case where there is no output structure (i.e., $\mathbf{A} = \mathbf{I}$). In this case, CLUHSIC is equivalent to standard kernel k-means. As can be seen from Table V, although CLUHSIC and CLUHSICAL are equivalent under the zero-one loss before normalization (which is also experimentally verified), CLUHSICAL outperforms CLUHSIC after normalization.

D. Equation (20) as a Good Approximation of $(\gamma\,\mathbf{J} + \mathbf{S})^{-1}$

Finally, we provide empirical evidence that (20) is a good approximation of $(\gamma\,\mathbf{J} + \mathbf{S})^{-1}$. We study the relative difference $|t_1 - t_2|/t_1$ in approximating the objective

[4]For example, for a ring with c clusters, there are $2c$ possible permutations obtained by starting from each of the c clusters and then visiting them in either clockwise or anticlockwise manner.

TABLE VI
RELATIVE APPROXIMATION ERRORS USING THE ZERO-ONE LOSS AND STRUCTURED LOSS

γ		data set	C					
			m	$0.1m$	$0.01m$	$0.001m$	$0.0001m$	$0.00001m$
1	0-1 loss	face	2.82	3.77e-02	3.94e-04	3.95e-06	3.96e-08	3.96e-10
		WIPO	1.26e-02	1.29e-04	1.29e-06	1.29e-08	1.29e-10	1.29e-12
		teapot	4.36e-01	4.69e-03	4.74e-05	4.74e-07	4.74e-09	4.74e-11
		SCOP ($\mathbf{K}_1^{Al}$)	2.75e-01	3.04e-03	3.09e-05	3.09e-07	3.09e-09	3.09e-11
		SCOP ($\mathbf{K}_2^{Al}$)	5.27e-01	6.47e-03	6.67e-05	6.70e-07	6.70e-09	6.70e-11
		SCOP ($\mathbf{K}_3^{Al}$)	2.95e-01	3.32e-03	3.38e-05	3.39e-07	3.39e-09	3.39e-11
	structured loss	face	4.26	5.40e-02	5.60e-04	5.62e-06	5.62e-08	5.62e-10
		WIPO	1.48e-02	1.52e-04	1.52e-06	1.52e-08	1.52e-10	1.52e-12
		teapot	5.53e-01	5.88e-03	5.93e-05	5.93e-07	5.93e-9	5.93e-11
		SCOP ($\mathbf{K}_1^{Al}$)	3.47e-01	3.92e-03	3.99e-05	4.00e-07	4.00e-09	4.00e-11
		SCOP ($\mathbf{K}_2^{Al}$)	6.84e-01	8.64e-03	8.95e-05	8.99e-07	8.99e-09	8.99e-11
		SCOP ($\mathbf{K}_3^{Al}$)	3.74e-01	4.32e-03	4.41e-05	4.43e-07	4.43e-09	4.43e-11
0.1	0-1 loss	face	3.77e-02	3.94e-04	3.95e-06	3.96e-08	3.96e-10	3.96e-12
		WIPO	1.29e-04	1.29e-06	1.29e-08	1.29e-10	1.29e-12	1.40e-14
		teapot	4.69e-03	4.73e-05	4.74e-07	4.74e-09	4.74e-11	4.72e-13
		SCOP ($\mathbf{K}_1^{Al}$)	3.04e-03	3.09e-05	3.09e-07	3.09e-09	3.09e-11	3.07e-13
		SCOP ($\mathbf{K}_2^{Al}$)	6.47e-03	6.67e-05	6.70e-07	6.70e-09	6.70e-11	6.69e-13
		SCOP ($\mathbf{K}_3^{Al}$)	3.32e-03	3.38e-05	3.39e-07	3.39e-09	3.39e-11	3.38e-13
	structured loss	face	5.40e-02	5.58e-04	5.62e-06	5.62e-08	5.62e-10	5.62e-12
		WIPO	1.52e-04	1.52e-06	1.52e-08	1.52e-10	1.52e-12	1.48e-14
		teapot	5.88e-03	5.92e-05	5.93e-07	5.93e-09	5.94e-11	7.49e-13
		SCOP ($\mathbf{K}_1^{Al}$)	3.92e-03	3.99e-05	4.00e-07	4.00e-09	4.00e-11	3.99e-13
		SCOP ($\mathbf{K}_2^{Al}$)	8.64e-03	8.95e-05	8.99e-07	8.99e-09	8.99e-11	8.97e-13
		SCOP ($\mathbf{K}_3^{Al}$)	4.32e-03	4.41e-05	4.43e-07	4.43e-09	4.43e-11	4.39e-13
0.01	0-1 loss	face	3.94e-04	3.95e-06	3.96e-08	3.96e-10	3.96e-12	3.76e-14
		WIPO	1.29e-06	1.29e-08	1.29e-10	1.29e-12	1.33e-14	1.13e-16
		teapot	4.73e-05	4.74e-07	4.74e-09	4.74e-11	4.72e-13	4.20e-15
		SCOP ($\mathbf{K}_1^{Al}$)	3.09e-05	3.09e-07	3.09e-09	3.09e-11	3.06e-13	2.89e-15
		SCOP ($\mathbf{K}_2^{Al}$)	6.67e-05	6.70e-07	6.70e-09	6.70e-11	6.69e-13	6.51e-15
		SCOP ($\mathbf{K}_3^{Al}$)	3.38e-05	3.39e-07	3.39e-09	3.39e-11	3.39e-13	3.79e-15
	structured loss	face	5.60e-04	5.62e-06	5.62e-08	5.62e-10	5.62e-12	5.53e-14
		WIPO	1.52e-06	1.52e-08	1.52e-10	1.51e-12	1.60e-14	3.38e-16
		teapot	5.93e-05	5.93e-07	5.93e-09	5.93e-11	5.88e-13	5.42e-15
		SCOP ($\mathbf{K}_1^{Al}$)	3.99e-05	4.00e-07	4.00e-09	4.00e-11	3.97e-13	4.52e-15
		SCOP ($\mathbf{K}_2^{Al}$)	8.95e-05	8.99e-07	8.99e-09	8.99e-11	8.97e-13	9.40e-15
		SCOP ($\mathbf{K}_3^{Al}$)	4.41e-05	4.43e-07	4.43e-09	4.43e-11	4.41e-13	3.61e-15

$t_1 = \mathbf{1}'(\gamma\,\mathbf{J} + \mathbf{S})^{-1}\mathbf{1}$ in (7) by its first-order approximation $t_2 = \mathbf{1}'\mathbf{S}^{-1}\mathbf{1} - \gamma\,\mathbf{1}'\mathbf{S}^{-1}\mathbf{J}\mathbf{S}^{-1}\mathbf{1}$ in (20). The labels $\mathbf{y}_i$'s required in the definition of $\mathbf{S}$ are set to the ground truths. We vary γ from $1, 0.1$ to 0.01, and C from $0.00001m$ to m (where m is the number of samples). As can be seen from Table VI, the approximation error, depending on C and γ, is usually very small.

V. CONCLUSION

In this paper, we studied the problem of clustering structured outputs. We showed that the loss function, which is known to be very important in the supervised structured prediction problems, can also be incorporated into unsupervised learning. In particular, we showed that CLUHSIC is implicitly using the often inappropriate zero-one loss. We provided detailed derivations and proposed an extension called CLUHSICAL that explicitly considers both the output dependency and loss function. Experimental results on a number of datasets show that CLUHSICAL (with structured loss) often outperforms CLUHSIC, XWSS, and kernel hierarchical k-means, and thus confirm the importance of the loss function in clustering structured data.

APPENDIX I
PROOF OF PROPOSITION 1

Introducing Lagrange multipliers $\alpha_{i\mathbf{y}}$'s for the equality constraints in (3) and denoting $\boldsymbol{\delta}_i = \boldsymbol{\delta\varphi}_i(\mathbf{y}), \bar{\boldsymbol{\delta}}_i = \boldsymbol{\delta\varphi}_i(\bar{\mathbf{y}})$, $\Delta_{i,j} = \Delta(\mathbf{y}_i, \mathbf{y}_j)$, $\Delta_{i\cdot} = \Delta(\mathbf{y}_i, \mathbf{y})$, and $\Delta_{i\bar{\cdot}} = \Delta(\mathbf{y}_i, \bar{\mathbf{y}})$, then the Lagrangian is $\mathcal{L} = 1/2\|\mathbf{w}\|^2 + (C/m)\sum_{i,\mathbf{y}\neq\mathbf{y}_i} \xi_{i\mathbf{y}}^2 - \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \left(\langle \mathbf{w}, \boldsymbol{\delta}_i\rangle - 1 + \xi_{i\mathbf{y}}/\sqrt{\Delta_{i\cdot}}\right)$. Setting its derivatives w.r.t. the primal variables to zero, and plugging these back into $\mathcal{L}$, we obtain

$$\mathcal{L} = \frac{1}{2}\left\|\sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\boldsymbol{\delta}_i\right\|^2 + \frac{C}{m}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\left[\frac{m}{2C}\frac{\alpha_{i\mathbf{y}}}{\sqrt{\Delta_{i\cdot}}}\right]^2$$
$$- \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\left(\left\langle \sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j} \alpha_{j\bar{\mathbf{y}}}\bar{\boldsymbol{\delta}}_j, \boldsymbol{\delta}_i\right\rangle - 1 + \frac{m}{2C}\frac{\alpha_{i\mathbf{y}}}{\Delta_{i\cdot}}\right)$$
$$= \frac{1}{2}\sum_{\substack{i,\mathbf{y}\neq\mathbf{y}_i\\ j,\bar{\mathbf{y}}\neq\mathbf{y}_j}} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\langle\boldsymbol{\delta}_i, \bar{\boldsymbol{\delta}}_j\rangle + \frac{C}{m}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\left[\frac{m}{2C}\frac{\alpha_{i\mathbf{y}}}{\sqrt{\Delta_{i\cdot}}}\right]^2$$

$$- \sum_{\substack{i,\mathbf{y}\neq\mathbf{y}_i \\ j,\bar{\mathbf{y}}\neq\mathbf{y}_j}} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\langle\boldsymbol{\delta}_i, \bar{\boldsymbol{\delta}}_j\rangle + \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} - \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\frac{m}{2C}\frac{\alpha_{i\mathbf{y}}^2}{\Delta_{i.}}$$

$$= \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} - \frac{1}{2}\sum_{\substack{i,\mathbf{y}\neq\mathbf{y}_i \\ j,\bar{\mathbf{y}}\neq\mathbf{y}_j}} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\langle\boldsymbol{\delta}_i, \bar{\boldsymbol{\delta}}_j\rangle - \frac{m}{4C}\sum_{i,\mathbf{y}\neq\mathbf{y}_i}\frac{\alpha_{i\mathbf{y}}^2}{\Delta_{i.}}$$

$$= \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} - \frac{1}{2}\sum_{\substack{i,\mathbf{y}\neq\mathbf{y}_i \\ j,\bar{\mathbf{y}}\neq\mathbf{y}_j}} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}}\left[\langle\boldsymbol{\delta}_i, \bar{\boldsymbol{\delta}}_j\rangle + \frac{m}{2C}\frac{\delta_{i\mathbf{y}j\bar{\mathbf{y}}}}{\sqrt{\Delta_{i.}}\sqrt{\Delta_{j\bar{.}}}}\right]$$

where $\delta_{i\mathbf{y}j\bar{\mathbf{y}}}$ is 1 if $i = j$ and $\mathbf{y} = \bar{\mathbf{y}}$, and 0 otherwise. This can then be rewritten as the form in (4).

Appendix II
Proof of Proposition 2

Note that, when the kernel k is scaled to γk as in Proposition 2, the $\mathbf{J}$ in (5) will be scaled accordingly as $\gamma\mathbf{J}$. First, we introduce a few lemmas.

Lemma 3: As a first-order approximation, minimizing $\mathbf{1}'(\gamma\mathbf{J}+\mathbf{S})^{-1}\mathbf{1}$ in (9) for the scaled kernel γk is the same as maximizing

$$\tilde{C}\mathbf{1}'\boldsymbol{\Delta}\mathbf{J}\boldsymbol{\Delta}\mathbf{1} - \frac{2C}{m}\mathbf{1}'\boldsymbol{\Delta}\mathbf{1} \tag{19}$$

where $\boldsymbol{\Delta} = \operatorname{diag}([\Delta_{i.}]_{i,\mathbf{y}\neq\mathbf{y}_i})$ and $\gamma, \tilde{C}$ are as defined in Proposition 2.

Proof: Note that $(\gamma\mathbf{J}+\mathbf{S})^{-1} = [\mathbf{S}(\gamma\mathbf{S}^{-1}\mathbf{J}+\mathbf{I})]^{-1} = \left[\mathbf{S}^{-1}(\gamma\mathbf{J})+\mathbf{I}\right]^{-1}\mathbf{S}^{-1}$. On using (10), $\|\mathbf{S}^{-1}(\gamma\mathbf{J})\|_p \leq \gamma\|\mathbf{S}^{-1}\|_p\|\mathbf{J}\|_p < 1$, this can then be expanded into an infinite series as [33]

$$(\gamma\mathbf{J}+\mathbf{S})^{-1} = \left(\sum_{i=0}^{\infty}\left(-\mathbf{S}^{-1}(\gamma\mathbf{J})\right)^i\right)\mathbf{S}^{-1}.$$

Taking the first-order approximation

$$(\gamma\mathbf{J}+\mathbf{S})^{-1} \simeq (\mathbf{I}-\mathbf{S}^{-1}(\gamma\mathbf{J}))\mathbf{S}^{-1} = \mathbf{S}^{-1} - \gamma\mathbf{S}^{-1}\mathbf{J}\mathbf{S}^{-1}. \tag{20}$$

Minimizing $-\mathbf{1}'(\gamma\mathbf{J}+\mathbf{S})^{-1}\mathbf{1}$ in (9) then becomes maximizing

$$\gamma\mathbf{1}'\mathbf{S}^{-1}\mathbf{J}\mathbf{S}^{-1}\mathbf{1} - \mathbf{1}'\mathbf{S}^{-1}\mathbf{1} = \tilde{C}\mathbf{1}'\boldsymbol{\Delta}\mathbf{J}\boldsymbol{\Delta}\mathbf{1} - \frac{2C}{m}\mathbf{1}'\boldsymbol{\Delta}\mathbf{1}$$

on using the definitions of $\mathbf{J}$ and $\mathbf{S}$ in (5) and (6). ■

Lemma 4: Using the tensor-product feature map in (8), the $\mathbf{1}'\boldsymbol{\Delta}\mathbf{J}\boldsymbol{\Delta}\mathbf{1}$ term in (19) can be simplified as $\operatorname{tr}(\mathbf{K}\widetilde{\boldsymbol{\Pi}}\mathbf{A}\widetilde{\boldsymbol{\Pi}}')$, where $\mathbf{K}, \widetilde{\boldsymbol{\Pi}}, \mathbf{A}$ are as defined in Proposition 2.

Proof: Again denote $\varphi_{ii} = \varphi(\mathbf{x}_i, \mathbf{y}_i), \varphi_{i.} = \varphi(\mathbf{x}_i, \mathbf{y})$ and $\varphi_{i\bar{.}} = \varphi(\mathbf{x}_i, \bar{\mathbf{y}})$. Note that

$$\begin{aligned}\mathbf{1}'\boldsymbol{\Delta}\mathbf{J}\boldsymbol{\Delta}\mathbf{1} &= \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j}\langle\boldsymbol{\delta}_i, \bar{\boldsymbol{\delta}}_j\rangle\Delta_{i.}\Delta_{j\bar{.}} \\ &= \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j}\langle\varphi_{ii}-\varphi_{i.}, \varphi_{jj}-\varphi_{j\bar{.}}\rangle\Delta_{i.}\Delta_{j\bar{.}} \\ &= \sum_{i,\mathbf{y}\neq\mathbf{y}_i}\sum_{j,\bar{\mathbf{y}}\neq\mathbf{y}_j}\big\langle\phi(\mathbf{x}_i)\otimes(\lambda(\mathbf{y}_i)-\lambda(\mathbf{y})), \\ &\quad\phi(\mathbf{x}_j)\otimes\big(\lambda(\mathbf{y}_j)-\lambda(\bar{\mathbf{y}})\big)\big\rangle\Delta_{i.}\Delta_{j\bar{.}} \\ &= \sum_{ij}\kappa(\mathbf{x}_i, \mathbf{x}_j)\sum_{\mathbf{y}\neq\mathbf{y}_i,\bar{\mathbf{y}}\neq\mathbf{y}_j}\big\langle\Delta_{i.}\big(\lambda(\mathbf{y}_i)-\lambda(\mathbf{y})\big), \\ &\quad\Delta_{j\bar{.}}\big(\lambda(\mathbf{y}_j)-\lambda(\bar{\mathbf{y}})\big)\big\rangle \\ &= \sum_{ij}\kappa(\mathbf{x}_i, \mathbf{x}_j)\Bigg\langle\sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta_{i.}\big(\lambda(\mathbf{y}_i)-\lambda(\mathbf{y})\big), \\ &\quad\sum_{\bar{\mathbf{y}}\neq\mathbf{y}_j}\Delta_{j\bar{.}}\big(\lambda(\mathbf{y}_j)-\lambda(\bar{\mathbf{y}})\big)\Bigg\rangle.\end{aligned} \tag{21}$$

Using (12)

$$\begin{aligned}&\sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta_{i.}\big(\lambda(\mathbf{y}_i)-\lambda(\mathbf{y})\big) \\ &= \left[\sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta_{i.}\right]\lambda(\mathbf{y}_i) + \sum_{\mathbf{y}\neq\mathbf{y}_i}\big[-\Delta_{i.}\lambda(\mathbf{y})\big] = (\widetilde{\boldsymbol{\Pi}}_{i.}\boldsymbol{\lambda})'\end{aligned}$$

where $\boldsymbol{\lambda} = [\lambda(\mathbf{y}_1), \ldots, \lambda(\mathbf{y}_{|\mathcal{Y}|})]'$. Therefore, (21) can be written as

$$\begin{aligned}&\sum_{ij}\kappa(\mathbf{x}_i, \mathbf{x}_j)(\widetilde{\boldsymbol{\Pi}}_{i.}\boldsymbol{\lambda})(\widetilde{\boldsymbol{\Pi}}_{j.}\boldsymbol{\lambda})' \\ &= \sum_{ij}\kappa(\mathbf{x}_i, \mathbf{x}_j)(\widetilde{\boldsymbol{\Pi}}_{i.}\boldsymbol{\lambda}\boldsymbol{\lambda}'\widetilde{\boldsymbol{\Pi}}'_{j.}) \\ &= \mathbf{1}'(\mathbf{K}\circ\widetilde{\boldsymbol{\Pi}}\mathbf{A}\widetilde{\boldsymbol{\Pi}}')\mathbf{1} = \operatorname{tr}(\mathbf{K}\widetilde{\boldsymbol{\Pi}}\mathbf{A}\widetilde{\boldsymbol{\Pi}}')\end{aligned}$$

where $\circ$ denotes the Hadamard (or elementwise) product. ■

Lemma 5: The $\mathbf{1}'\boldsymbol{\Delta}\mathbf{1}$ term in (19) can be simplified as $(1/2)\|\widetilde{\boldsymbol{\Pi}}\|_1$, where $\widetilde{\boldsymbol{\Pi}}$ is as defined in Proposition 2.

Proof:

$$\mathbf{1}'\boldsymbol{\Delta}\mathbf{1} = \sum_{i=1}^{m}\sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta_{i.} = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{|\mathcal{Y}|}|\widetilde{\boldsymbol{\Pi}}_{ij}| = \frac{1}{2}\|\widetilde{\boldsymbol{\Pi}}\|_1$$

■

Proof: (of Proposition 2). Result follows immediately by plugging Lemmas 4 and 5 into (19). ■

Appendix III
Proof of the Results in Section III-C

Proof: (of Corollary 1). When the zero-one loss is used, we have

$$\mathbf{1}'\boldsymbol{\Delta}\mathbf{1} = \sum_{i=1}^{m}\sum_{\mathbf{y}\neq\mathbf{y}_i}\Delta_{i.} = \sum_{i=1}^{m}(|\mathcal{Y}|-1) = m|\mathcal{Y}| - m.$$

Thus, the second term in (19) is a constant, and (11) reduces to

$$\begin{aligned}\max_{\widetilde{\boldsymbol{\Pi}}}\quad & \operatorname{tr}(\mathbf{K}\widetilde{\boldsymbol{\Pi}}\mathbf{A}\widetilde{\boldsymbol{\Pi}}') \\ \text{s.t}\quad & \widetilde{\boldsymbol{\Pi}}_{i.} \text{ is as defined in (12).}\end{aligned} \tag{22}$$

This is the same as (2) of CLUHSIC, except that the definitions of the partition matrix are, apparently, different. However, note that with the zero-one loss, $\widetilde{\boldsymbol{\Pi}}_{i.}$ in (12) reduces to

$$\boldsymbol{\Pi}_{i.} = [\underbrace{-1, \ldots, -1}_{(\mathbf{y}_i-1)\text{ times}}, |\mathcal{Y}|-1, \underbrace{-1, \ldots, -1}_{(|\mathcal{Y}|-\mathbf{y}_i)\text{ times}}]. \tag{23}$$

We can add 1 to each of its entries, and obtain

$$\hat{\mathbf{\Pi}}_{i\cdot} = [0, \ldots, 0, |\mathcal{Y}|, 0, \ldots, 0].$$

Note that $\hat{\mathbf{\Pi}}$ is essentially the same as the partition matrix $\mathbf{\Pi}$ in CLUHSIC, except that the only nonzero entry is changed from 1 to the constant $|\mathcal{Y}|$. Using the fact that $\mathbf{K}$ is always centered (i.e., $\mathbf{K1} = \mathbf{0}$) in CLUHSIC

$$\begin{aligned}\mathrm{tr}(\mathbf{K}\hat{\mathbf{\Pi}}\mathbf{A}\hat{\mathbf{\Pi}}') &= \mathrm{tr}(\mathbf{K}(\mathbf{\Pi}+\mathbf{11}')\mathbf{A}(\mathbf{\Pi}+\mathbf{11}')')\\ &= \mathrm{tr}(\mathbf{K\Pi A\Pi}') + \mathrm{tr}(\mathbf{K\Pi A11}')\\ &\quad + \mathrm{tr}(\mathbf{K11}'\mathbf{A\Pi}') + \mathrm{tr}(\mathbf{K11}'\mathbf{A11}')\\ &= \mathrm{tr}(\mathbf{K\Pi A\Pi}').\end{aligned}$$

Hence

$$\max_{\hat{\mathbf{\Pi}}} \mathrm{tr}(\mathbf{K}\hat{\mathbf{\Pi}}\mathbf{A}\hat{\mathbf{\Pi}}') \Leftrightarrow \max_{\mathbf{\Pi}} \mathrm{tr}(\mathbf{K\Pi A\Pi}').$$

and (19) leads to the same partitioning as CLUHSIC. ■

Proof: (of Lemma 1). Setting $\mathbf{\Pi}$ as in (15), objective (14) reduces to

$$\sum_{c=1}^{|\mathcal{Y}|}\sum_{\mathbf{x}_i\in c}\|\phi(\mathbf{x}_i)-\boldsymbol{\mu}_c\|^2_{\mathcal{F}} \tag{24}$$

which is the standard k-means objective. The cluster center can be easily obtained as in (16). Moreover, note that (24) can be written as $\sum_{c=1}^{|\mathcal{Y}|}\sum_{\mathbf{x}_i\in c}\left(\kappa(\mathbf{x}_i,\mathbf{x}_i) - 2\langle\phi(\mathbf{x}_i),\boldsymbol{\mu}_c\rangle + \langle\boldsymbol{\mu}_c,\boldsymbol{\mu}_c\rangle\right)$. Substituting in (16), we have

$$\begin{aligned}&\sum_{c=1}^{|\mathcal{Y}|}\sum_{\mathbf{x}_i\in c}\kappa(\mathbf{x}_i,\mathbf{x}_i) - \sum_{c=1}^{|\mathcal{Y}|}\left(\sum_{\mathbf{x}_i,\mathbf{x}_j\in c}\frac{1}{m_c}\kappa(\mathbf{x}_i,\mathbf{x}_j)\right)\\ &= \sum_{i=1}^{m}\kappa(\mathbf{x}_i,\mathbf{x}_i) - \sum_{c=1}^{|\mathcal{Y}|}\left(m_c\sum_{\mathbf{x}_i,\mathbf{x}_j\in c}\frac{1}{m_c^2}\kappa(\mathbf{x}_i,\mathbf{x}_j)\right).\end{aligned}$$

Now the first term is constant, so this can be dropped from the optimization. When $m_c = m/|\mathcal{Y}|$, the objective reduces to the CLUHSIC objective with $\mathbf{A} = (m/|\mathcal{Y}|)\mathbf{I}$ and its ℓ_1-normalized partition matrix has elements $\bar{\Pi}_{ic} = \begin{cases}\frac{1}{m_c} & \phi(\mathbf{x}_i)\in c,\\ 0 & \phi(\mathbf{x}_i)\notin c.\end{cases}$ ■

Proof: (of Lemma 2). With the setting of $\mathbf{\Pi}$, (14) reduces to

$$\sum_{c=1}^{|\mathcal{Y}|}\left((|\mathcal{Y}|-1)\sum_{\mathbf{x}_i\in c}\|\boldsymbol{\mu}_c-\phi(\mathbf{x}_i)\|^2_{\mathcal{F}} + \sum_{\mathbf{x}_i\notin c}\|\boldsymbol{\mu}_c+\phi(\mathbf{x}_i)\|^2_{\mathcal{F}}\right) \tag{25}$$

and the cluster center can be obtained by (18) as

$$\begin{aligned}\boldsymbol{\mu}_c &= \frac{\left((|\mathcal{Y}|-1)\sum_{\mathbf{x}_i\in c}\phi(\mathbf{x}_i) - \sum_{\mathbf{x}_i\notin c}\phi(\mathbf{x}_i)\right)}{m_c(|\mathcal{Y}|-1)+m-m_c}\\ &= \sum_{i=1}^{m}\frac{\mathbf{\Pi}_{ic}}{\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|}\phi(\mathbf{x}_i)\end{aligned}$$

where

$$\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}| = m_c(|\mathcal{Y}|-1)+m-m_c = m_c(|\mathcal{Y}|-2)+m.$$

Substituting (18) back into (25), we have

$$\begin{aligned}&\sum_{c=1}^{|\mathcal{Y}|}\left((|\mathcal{Y}|-1)\sum_{\mathbf{x}_i\in c}\kappa(\mathbf{x}_i,\mathbf{x}_i) + \sum_{\mathbf{x}_i\notin c}\kappa(\mathbf{x}_i,\mathbf{x}_i)\right)\\ &\quad - \sum_{c=1}^{|\mathcal{Y}|}\left(\frac{1}{\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|}\sum_{i=1,k=1}^{m}\Pi_{ic}\mathbf{\Pi}_{kc}\kappa(\mathbf{x}_i,\mathbf{x}_k)\right)\\ &= \sum_{i=1}^{m}2(|\mathcal{Y}|-1)\kappa(\mathbf{x}_i,\mathbf{x}_i)\\ &\quad - \sum_{c=1}^{|\mathcal{Y}|}\left(\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|\sum_{i=1,k=1}^{m}\frac{\mathbf{\Pi}_{ic}}{\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|}\frac{\mathbf{\Pi}_{kc}}{\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|}\kappa(\mathbf{x}_i,\mathbf{x}_k)\right).\end{aligned}$$

Since the first term is also constant and can be ignored, the objective then reduces to the CLUHSICAL objective with $\mathbf{A} = (2(|\mathcal{Y}|-1)m)/|\mathcal{Y}|$ when $m_c = m/|\mathcal{Y}|$, and its ℓ_1-normalized partition matrix has elements $\bar{\mathbf{\Pi}}_{ic} = (\mathbf{\Pi}_{ic})/\sum_{j=1}^{m}|\mathbf{\Pi}_{jc}|$. ■

References

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[2] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt, "A dependence maximization view of clustering," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 815–822.

[3] J. MacQueen, "Some methods of classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math., Statist. Probability*, 1967, pp. 281–297.

[4] N. Slonim and Y. Weiss, "Maximum likelihood and the information bottleneck," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 351–358.

[5] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.

[6] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 849–856.

[7] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in *Proc. Int. Conf. Algorithmic Learn. Theory*, Singapore, Oct. 2005, pp. 63–77.

[8] T. Gärtner, "A survey of kernels for structured data," *ACM SIGKDD Explorations Newslett.*, vol. 5, no. 1, pp. 49–58, Jul. 2003.

[9] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *J. Mach. Learn. Res.*, vol. 2, pp. 125–137, Dec. 2001.

[10] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 780–784, May 2002.

[11] L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans, "Discriminative unsupervised learning of structured predictors," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 1057–1064.

[12] H. Valizadegan and R. Jin, "Generalized maximum margin clustering and unsupervised kernel learning," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press, 2007, pp. 1417–1424.

[13] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005, pp. 1537–1544.

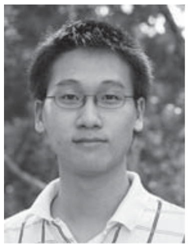
[14] L. Xu and D. Schuurmans, "Unsupervised and semi-supervised multi-class support vector machines," in *Proc. 20th Nat. Conf. Artificial Intell.*, Pittsburgh, PA, 2005, pp. 904–910.

[15] F. Bach and Z. Harchaoui, "DIFFRAC: A discriminative and flexible framework for clustering," in *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press, 2008, pp. 49–56.

[16] B. Taskar, "Learning structured prediction models: A large margin approach," Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., Palo Alto, CA, 2004.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Dec. 2005.

[18] K. Fukumizu, F. R. Bach, and M. I. Jordan, “Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces,” *J. Mach. Learn. Res.*, vol. 5, pp. 73–99, Dec. 2004.
[19] L. Song, “Learning via Hilbert space embedding of distributions,” Ph.D. thesis, School of Information Technologies, Univ. Sydney, NSW, Australia, 2008.
[20] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, “Spectral relaxation for K-means clustering,” in *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press, 2002.
[21] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 556–562.
[22] C. Saunders, A. Gammerman, and V. Vovk, “Ridge regression learning algorithm in dual variables,” in *Proc. 15th Int. Conf. Mach. Learn.*, San Francisco, CA, 1998, pp. 515–521.
[23] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.
[24] L. Xu, A. Krzyzak, and E. Oja, “Rival penalized competitive learning for clustering analysis, RBF net, and curve detection,” *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 636–649, Jul. 1993.
[25] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, “SCOP: A structural classification of proteins database for the investigation of sequences and structures,” *J. Molecular Biol.*, vol. 247, no. 4, pp. 536–540, Apr. 1995.
[26] S. Bhattacharya, C. Bhattacharyya, and N. R. Chandra, “Structural alignment based kernels for protein structure classification,” in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 73–80.
[27] D. Haussler, “Convolution kernels on discrete structures,” Dept. Comput. Sci., Univ. California, Santa Cruz, CA, Tech. Rep. UCSC-CRL-99-10, 1999.
[28] L. Cai and T. Hofmann, “Hierarchical document categorization with support vector machines,” in *Proc. ACM 13th Conf. Inf. Knowl. Manage.*, Washington D.C., Nov. 2004, pp. 78–87.
[29] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, “Kernel-based learning of hierarchical multilabel classification models,” *J. Mach. Learn. Res.*, vol. 7, pp. 1601–1626, Dec. 2006.
[30] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley-Interscience, 2001.
[31] J. H. Ward, Jr., “Hierarchical grouping to optimize an objective function,” *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 236–244, Mar. 1963.
[32] K. Q. Weinberger and L. K. Saul, “An introduction to nonlinear dimensionality reduction by maximum variance unfolding,” in *Proc. 21st Nat. Conf. Artificial Intell.*, Boston, MA, Jul. 2006, pp. 1683–1686.
[33] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.



Wenliang Zhong received the B.Sc. and M.Sc. degrees in computer science from the Sun Yat-sen University, Guangdong, China, in 2007 and 2009, respectively. Currently, he is pursuing the Ph.D. degree in computer science at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

His current research interests include kernel methods, machine learning, and computational intelligence.

Weike Pan received the B.S. degree from the College of Computer Science, Zhejiang University, Hangzhou, China, in 2005. He is a Ph.D. candidate in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

His current research interests include transfer learning, recommender systems, and statistical machine learning.

Mr. Pan is a Student Member of the Association for the Advancement of Artificial Intelligence.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, China, in 1996.

He was with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, as an Assistant Professor. Since 2000, he has been an Associate Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award for 2004 in 2006, and the Natural Science Award from the Ministry of Education, China, for 2008 in 2009. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS and the *Neurocomputing Journal*.

Ivor W. Tsang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, China, in 2007.

He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. He is also the Deputy Director of the Center for Computational Intelligence, NTU. His current research interests include machine learning, kernel methods, large-scale optimization and its applications to data mining, and pattern recognitions.

Dr. Tsang was awarded the prestigious IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award for 2004 in 2006. He clinched the second-class prize of the National Natural Science Award from the Ministry of Education, China, for 2008 in 2009. His work on transfer learning for visual event recognition was awarded the Best Student Paper Prize at the 22nd IEEE Computer Society Conference on Computer Vision and Pattern Recognition in 2010. His work on speech adaptation earned him the Best Paper Award from the IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006. He was also awarded the Microsoft Fellowship in 2005.