2013 IEEE 13th International Conference on Data Mining

Clustering on Multiple Incomplete Datasets via Collective Kernel Learning

Weixiang Shao
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois 60607-7053
Email: wshao4@uic.edu

Xiaoxiao Shi
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois 60607-7053
Email: xshi9@uic.edu

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois 60607-7053
Email: psyu@uic.edu

***Abstract*—Multiple datasets containing different types of features may be available for a given task. For instance, users' profiles can be used to group users for recommendation systems. In addition, a model can also use users' historical behaviors and credit history to group users. Each dataset contains different information and suffices for learning. A number of clustering algorithms on multiple datasets were proposed during the past few years. These algorithms assume that at least one dataset is complete. So far as we know, all the previous methods will not be applicable if there is no complete dataset available. However, in reality, there are many situations where no dataset is complete. As in building a recommendation system, some new users may not have profiles or historical behaviors, while some may not have credit history. Hence, no available dataset is complete. In order to solve this problem, we propose an approach called Collective Kernel Learning to infer hidden sample similarity from multiple incomplete datasets. The idea is to collectively completes the kernel matrices of incomplete datasets by optimizing the alignment of shared instances of the datasets. Furthermore, a clustering algorithm is proposed based on the kernel matrix. The experiments on both synthetic and real datasets demonstrate the effectiveness of the proposed approach. The proposed clustering algorithm outperforms the comparison algorithms by as much as two times in normalized mutual information.**

I. Introduction

In many real world data mining problems, the same instance may appear in different datasets with different representations. Different datasets may emphasize different aspects of instances. An example is grouping the users in an user-oriented recommendation system. For this task, related datasets can be (1) user profile database (as shown in Fig. 1a), (2) users' log data (as shown in Fig. 1b), and (3) users' credit score (as shown in Fig. 1c). Learning with such type of data is commonly referred to as multiview learning [1], [2]. Although there are some previous works on multiple datasets, all of them assume the completeness of the different datasets. As far as we know, even the most recently work requires at least one dataset is complete [3]. However, in the real world applications, there are many situations in which complete datasets are not available. For instance, in Fig. 1a, User3 does not complete her profile. However, she has browsing log recorded by the browser. In Fig. 1b, checks and crosses indicates whether user visited the website recently. From the figure, we can see that User2 and User4 do not have browsing behavior history. This may because that they are new users to the system or they refuse to share the historical behaviors with the system. In Fig. 1c, only User1 and User2 have credit scores in the system. In the situation as shown in Fig. 1, all the previous method will not be applicable. It is very important to find an approach that can work for incomplete datasets.

In order to deal with the incompleteness of the datasets, it is a natural way to complete the original datasets first. However, it is very hard and time-consuming to directly predict the missing features in each dataset especially if there are large number of missing features. Instead, we propose an approach called **Co**llective **K**ernel **L**earning (CoKL). This approach iteratively completes the kernel matrix of each dataset using the kernel of other datasets. Basically, CoKL is based on aligning the similarities between examples across all datasets. The completed kernel matrices can be used in any kernel based clustering algorithms. In this paper, we also propose a clustering algorithm based on CoKL and Kernel Canonical Correlation Analysis (KCCA). The proposed clustering algorithm first uses CoKL to complete the kernel matrices. Based on the completed kernel matrices, KCCA could find the projections that maximize the correlations between the datasets. Then we can perform any standard clustering algorithms on the projected space. As compared with previous papers, this paper has several advantages:

1) The proposed clustering algorithm can be used in situations even when all the datasets are incomplete, in which the other methods are not applicable.
2) Collective kernel learning does not require predicting the missing features in the incomplete datasets using complex method. Predicting the missing features may be very time-consuming when there are large number of missing features. Instead, we construct the full kernel matrices corresponding to the incomplete datasets iteratively using the shared examples between different datasets.

In order to evaluate the quality of CoKL and the proposed clustering algorithm that uses CoKL and KCCA, we conduct several experiments on the UCI seeds data and handwritten Dutch numbers recognition data. The proposed clustering algorithm outperforms the comparison algorithms by as much as two times in normalized mutual information. The experiment on the convergence of CoKL shows that CoKL converges quickly in all the experiment settings (less than 10 iterations). Further experiment shows that the number of iterations needed to convergence does not change too much for different missing rates [4].

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.117

	Name	Age	Country
User1	Bob	31	USA
User2	Angeli	21	USA
User3	?	?	?
User4	Tom	40	Canada

(a) User profile. User3 doesn't complete her profile.

	facebook	twitter	Google+
User1	✔	✖	✖
User2	?	?	?
User3	✔	✖	✔
User4	?	?	?

(b) Browsing behaviors. User2 and User4 are new users, thus no browsing behaviors are available.

	Credit Score
User1	Fair
User2	Good
User3	?
User4	?

(c) Credit scores. No credit scores for User3 and User4.

Fig. 1: Different datasets for grouping the users in the recommendation systems.

TABLE I: Notations used in this paper.

Notation	Description
X and Y	Incomplete datasets
$\mathcal{C}$	$\mathcal{C} = \{(x_1, x_2), ..., (x_c, y_c)\}$. The set of examples with features present in both X and Y. c is the set size.
$\mathcal{M}_1$	$\mathcal{M}_1 = \{x_{c+1}, ..., x_{c+m_1}\}$. The set of examples with features only present in dataset X. m_1 is the set size.
$\mathcal{M}_2$	$\mathcal{M}_2 = \{y_{c+m_1+1}, ..., y_{c+m_1+m_2}\}$. The set of examples with features only present in dataset Y. m_2 is the set size.
K_x	A full kernel matrix of data set X with dimension $(c + m_1 + m_2) \times (c + m_1 + m_2)$.
K_y	A full kernel matrix of data set Y with dimension $(c + m_1 + m_2) \times (c + m_1 + m_2)$.
$k(x_i, x_j)$	The kernel similarity between two examples x_i and x_i
$\mathcal{L}_x = D_x - K_x$	The graph Laplacian of Kernel K_x, where D_x is the diagonal matrix consisting of the row sums of K_x.
$\mathcal{L}_y = D_y - K_y$	The graph Laplacian of Kernel K_y, where D_y is the diagonal matrix consisting of the row sums of K_y.
$w_x = X\alpha$	The projection directions for X in CCA problem. Here, α is a vector of size N
$w_y = Y\beta$	The projection directions for Y in CCA problem. Here, β is a vector of size N
ϕ	The mapping function that maps a lower dimension data into higher dimension space.

The rest of this paper is organized as follows: In the next section, we will describe the formulation of the problem. In section III, we will describe the proposed collective kernel learning. CCA and KCCA are introduced and the clustering algorithm using CoKL and KCCA is described in section IV. Experiment settings and result analysis are described in section V. The results on different data settings show that the proposed clustering algorithm outperforms the comparison algorithms.

II. Problem Fomulation

Before we describe the formulation of the problem, we summarize some notations used in this paper in Table I.

Given two related datasets X and Y, we assume both of these two related datasets are incomplete. The features for dataset X are available for only a subset of the total examples, and the features for dataset Y are available for another subset of the total examples. We also assume these two datasets can cover all the examples, i.e., here are no examples that are missing in both datasets. The goal is to derive a clustering solution $\mathcal{S}$ based on both datasets. Since both datasets are incomplete, we denote $\mathcal{C} = \{(x_1, y_1), ..., (x_c, y_c)\}$ as the set of examples with features present in both X and Y, $\mathcal{M}_1 = \{x_{c+1}, ..., x_{c+m_1}\}$ as the set of examples with features only present in dataset X, and $\mathcal{M}_2 = \{x_{c+m_1+1}, ..., x_{c+m_1+m_2}\}$ as the set of examples with feature only present in dataset Y. So we can rewrite examples in these two datasets as:

$$X = \begin{pmatrix} X_c \\ X_{m_1} \\ X_{m_2} =? \end{pmatrix} \qquad Y = \begin{pmatrix} Y_c \\ Y_{m_1} =? \\ Y_{m_2} \end{pmatrix}.$$

Then we can denote K_x, a $(c + m_1 + m_2) \times (c + m_1 + m_2)$ matrix, as kernel matrix defined over all the examples using features from dataset X. The corresponding graph Laplacian is defined as $\mathcal{L}_x = D_x - K_x$, where D_x is the diagonal matrix consisting of the row sums of K_x along it's diagonals. Likewise, for dataset $\mathcal{Y}$, we denote the kernel matrix by K_y, and the corresponding graph Laplacian by $\mathcal{L}_y = D_y - K_y$. However, since features for both $\mathcal{X}$ and $\mathcal{Y}$ are only available for a subset of the total examples, only 4 subblock of the full kernel matrix K_x (K_y) with size $c \times c$, $c \times m_1$, $m_1 \times c$, $m_1 \times m_1$ ($c \times c$, $c \times m_2$, $m_2 \times c$, $m_2 \times m_2$) will be available (see Equation 1).

$$K_x = \begin{pmatrix} K_x^{cc} & K_x^{cm_1} & K_x^{cm_2} =? \\ (K_x^{cm_1})^T & K_x^{m_1m_1} & K_x^{m_1m_2} =? \\ (K_x^{cm_2})^T =? & (K_x^{m_1m_2})^T =? & K_x^{m_2m_2} =? \end{pmatrix}$$
$$K_y = \begin{pmatrix} K_y^{cc} & K_y^{cm_1} =? & K_y^{cm_2} \\ (K_y^{cm_1})^T =? & K_y^{m_1m_1} =? & K_y^{m_1m_2} =? \\ (K_y^{cm_2})^T & (K_y^{m_1m_2})^T =? & K_y^{m_2m_2} \end{pmatrix} \quad (1)$$

In order to apply any kernel approach for clustering, one must first build the full kernel matrix K_x and K_y. To achieve this goal, we borrow the idea from Laplacian regularization [5], [6]. In other words, we first generate the graph Laplacian $\mathcal{L}_x$ for the kernel matrix K_x. Then $tr(\mathcal{L}_x K_y)$ reflects the "inconsistence" of the kernel matrix K_y when we "explain" it with the graph Laplacian $\mathcal{L}_x$ from K_x. In this paper, tr denotes the matrix trace. Under the assumption that K_x and K_y should contain consensus information, we should minimize the "inconsistence" $tr(\mathcal{L}_x K_y)$, and similarly for $tr(\mathcal{L}_y K_x)$. More formally, the objective can be written as follows:

$$\min_{K_y \succeq 0} tr(\mathcal{L}_x K_y) \quad (2)$$
$$\min_{K_x \succeq 0} tr(\mathcal{L}_y K_x) \quad (3)$$
$$s.t. \quad K_y(i_1, j_1) = k(y_{i_1}, y_{j_1}),$$
where i_1 and j_1 are instances in dataset Y.

$$K_x(i_2, j_2) = k(x_{i_2}, x_{j_2}),$$
where i_2 and j_2 are instances in dataset X.

Here, $k(y_{i_1}, y_{j_1})$ is the kernel similarity between two examples y_{i_1} and y_{i_2} in dataset Y, and $k(x_{i_2}, x_{j_2})$ is the kernel similarity between two examples x_{i_1} and x_{i_2} in dataset X. The objective functions above optimize the alignment between K_x and K_y, given the known part of K_x and K_y. Now we only need to solve the above optimization problems.

III. Collective Kernel Learning

To construct the full kernel matrices for incomplete datasets, we need to solve the optimization problems in Equations 2 and 3. However, optimizing Equation 2 requires the completeness of K_x, and optimizing Equation 3 requires the completeness of K_y. Since both datasets are incomplete, none of K_x and K_y is complete. We could not just solve these optimization problems directly. However, we could use collective kernel learning to approximately solve the problem. We can first fix one of the kernel matrix, say fix K_x by giving the missing features in dataset $\mathcal{X}$ initial guesses to construct the initial full kernel matrix K_x. Then we can optimize K_y by solving one of the two optimization problems $\min_{K_y \succeq 0} tr(\mathcal{L}_x K_y)$. Using the completed kernel matrix K_y, we can optimize K_x by solving $\min_{K_x \succeq 0} tr(\mathcal{L}_y K_x)$. This optimization process can continue until it converges.

Without loosing generality, we first fix K_x (filling the missing features in X with average values for continuous features and majority values for discrete features) and use K_x to solve the optimization problem in Equation 2. Since both the kernel matrices K_x and K_y should satisfy the positive semi-definite constraint, we can express K_y as AA^T (or K_x as BB^T), where A (or B) is a matrix of real numbers. Let us write A as $A = \begin{pmatrix} A_c \\ A_{m_1} \\ A_{m_2} \end{pmatrix}$, and $\mathcal{L}_x$ as:

$$\mathcal{L}_x = \begin{pmatrix} \mathcal{L}_x^{cc} & \mathcal{L}_x^{cm_1} & \mathcal{L}_x^{cm_2} \\ (\mathcal{L}_x^{cm_1})^T & \mathcal{L}_x^{m_1 m_1} & \mathcal{L}_x^{m_1 m_2} \\ (\mathcal{L}_x^{cm_2})^T & (\mathcal{L}_x^{m_1 m_2})^T & \mathcal{L}_x^{m_2 m_2} \end{pmatrix}.$$

Using these and the property of trace, we can rewrite Equation 2 as follows:

$$\begin{aligned} &\min_A tr\left(\mathcal{L}_x A A^T\right) = \min_A tr\left(A^T \mathcal{L}_x A\right) \\ &= \min_{A_c, A_{m_1}, A_{m_2}} tr\left(\begin{pmatrix} A_c \\ A_{m_1} \\ A_{m_2} \end{pmatrix}^T \times \right. \\ &\left. \begin{pmatrix} \mathcal{L}_x^{cc} & \mathcal{L}_x^{cm_1} & \mathcal{L}_x^{cm_2} \\ (\mathcal{L}_x^{cm_1})^T & \mathcal{L}_x^{m_1 m_1} & \mathcal{L}_x^{m_1 m_2} \\ (\mathcal{L}_x^{cm_2})^T & (\mathcal{L}_x^{m_1 m_2})^T & \mathcal{L}_x^{m_2 m_2} \end{pmatrix} \times \begin{pmatrix} A_c \\ A_{m_1} \\ A_{m_2} \end{pmatrix}\right). \end{aligned} \tag{4}$$

Expanding the above, and using the fact that A_c and A_{m_2} are constant (since $A_c A_c^T = K_y^{cc}$ and $A_{m_2} A_{m_2}^T = K_y^{m_2 m_2}$ are constant), we get:

$$\begin{aligned} &\min_{A_{m_1}} tr(A_c^T \mathcal{L}_x^{cc} A_c + A_{m_1}^T (\mathcal{L}_x^{cm_1})^T A_c + A_{m_2}^T (\mathcal{L}_x^{cm_2})^T A_c + \\ &A_c^T \mathcal{L}_x^{cm_1} A_{m_1} + A_{m_1}^T \mathcal{L}_x^{m_1 m_1} A_{m_1} + A_{m_2}^T (\mathcal{L}_x^{m_1 m_2})^T A_{m_1} + \\ &A_c^T \mathcal{L}_x^{cm_2} A_{m_2} + A_{m_1}^T \mathcal{L}_x^{m_1 m_2} A_{m_2} + A_{m_1}^T \mathcal{L}_x^{m_2 m_2} A_{m_2}). \end{aligned}$$

Using the fact that $A_c A_c^T = K_y^{cc}$ and $A_{m_2} A_{m_2}^T = K_y^{m_2 m_2}$ are constant, and for any matrix X, $tr(X) = tr(X^T)$, we can simplify the above as:

$$\begin{aligned} &\min_{A_{m_1}} 2\, tr(A_{m_1} A_c^T \mathcal{L}_x^{cm_1}) + 2\, tr(A_{m_2} A_c^T \mathcal{L}_x^{cm_2}) + \\ &2\, tr(A_{m_2} A_{m_1}^T \mathcal{L}_x^{m_1 m_2}) + tr(A_{m_1} A_{m_1}^T \mathcal{L}_x^{m_1 m_1}). \end{aligned}$$

Taking derivative w.r.t. A_{m_1}, setting it to zero and solving it, we get:

$$A_{m_1} = -(\mathcal{L}_x^{m_1 m_1})^{-1} \left((\mathcal{L}_x^{cm_1})^T A_c - \mathcal{L}_x^{m_1 m_2} A_{m_2}\right). \tag{5}$$

Thus

$$A = \begin{pmatrix} A_c \\ -(\mathcal{L}_x^{m_1 m_1})^{-1} \left((\mathcal{L}_x^{cm_1})^T A_c - \mathcal{L}_x^{m_1 m_2} A_{m_2}\right) \\ A_{m_2} \end{pmatrix}.$$

Then using $K_y = AA^T$, $A_c A_c^T = K_y^{cc}$, $A_c A_{m_2}^T = K_y^{cm_2}$, $A_{m_2} A_{m_2} = K_y^{m_2 m_2}$, we get:

$$K_y = \begin{pmatrix} K_y^{cc} & K_y^{cm_1} & K_y^{cm_2} \\ (K_y^{cm_1})^T & K_y^{m_1 m_1} & K_y^{m_1 m_2} \\ (K_y^{cm_2})^T & (K_y^{m_1 m_2})^T & K_y^{m_2 m_2} \end{pmatrix}, \tag{6}$$

where

$$\begin{aligned} K_y^{cm_1} &= -(K_y^{cc} \mathcal{L}_x^{cm_1} + \mathcal{L}_x^{cm_2} (\mathcal{L}_x^{m_1 m_2})^T)((\mathcal{L}_x^{m_1 m_1})^{-1})^T \\ K_y^{m_1 m_1} &= (\mathcal{L}_x^{m_1 m_1})^{-1}((\mathcal{L}_x^{cm_1})^T K_y^{cc} \\ &\quad + \mathcal{L}_x^{m_1 m_2} (K_y^{cm_2})^T \mathcal{L}_x^{cm_1} + (\mathcal{L}_x^{cm_1})^T K_y^{cm_2} (\mathcal{L}_x^{m_1 m_2})^T \\ &\quad + \mathcal{L}_x^{m_1 m_2} K_y^{m_2 m_2} (\mathcal{L}_x^{m_1 m_2})^T)((\mathcal{L}_x^{m_1 m_1})^{-1})^T \\ K_y^{m_1 m_2} &= -(\mathcal{L}_x^{m_1 m_1})^{-1}((\mathcal{L}_x^{cm_1})^T K_y^{cm_2} + \mathcal{L}_x^{m_1 m_2} K_y^{m_2 m_2}). \end{aligned}$$

Similarly, fix K_y, solving the optimization in Equation 3, we can get a new K_x. So we can iteratively solve the optimization problems in Equations 2 and 3 until it gets convergence. The whole algorithm is shown in Algorithm 1.

Although Algorithm 1 is for two incomplete datasets, it is important to note that the generalization can be easily done. By completing the kernel matrices in a cyclic iteration, Algorithm 1 can be easily generalized to more than two incomplete datasets . Assume we have k incomplete datasets $X_1, ..., X_k$. We first complete the kernel matrix K_2 using the initial kernel matrix K_1 by Equation 6. We can continue completing kernel matrix K_{i+1} using kernel matrix K_i by Equation 6, until we complete K_k. After using K_k to complete K_1, we can start another iteration cycle from K_1 to K_k, until it converges.

IV. Clustering Algorithm Based on Collective Kernel Learning and KCCA

In this section, we propose a clustering algorithm based on collective kernel learning and kernel canonical correlation analysis.

Algorithm 1 Collective Kernel Learning (CoKL)

Input: Incomplete Datasets X and Y
Output: The full kernel matrices K_x and K_y

1: Give initial values to the missing features in the two dataset.
2: Calculate the kernel matrices K_x and K_y.
3: $\mathcal{L}_x \leftarrow D_x - K_x$
4: $\mathcal{L}_y \leftarrow D_Y - K_y$
5: **repeat**
6:

$$A \leftarrow \begin{pmatrix} A_c \\ -(\mathcal{L}_x^{m_1 m_1})^{-1}\left((\mathcal{L}_x^{cm_1})^T A_c - \mathcal{L}_x^{m_1 m_2} A_{m_2}\right) \\ A_{m_2} \end{pmatrix}$$

7: Calculate the new full kernel matrix K_y' using A.
8:

$$B \leftarrow \begin{pmatrix} B_c \\ B_{m_1} \\ -(\mathcal{L}_y^{m_2 m_2})^{-1}\left((\mathcal{L}_y^{cm_2})^T B_c - \mathcal{L}_y^{m_2 m_1} B_{m_1}\right) \end{pmatrix}$$

9: Calculate the new full kernel matrix K_x' using B.
10: $K_x \leftarrow K_x'$.
11: $K_y \leftarrow K_y'$.
12: **until** Convergence

A. CCA and Kernel CCA

Canonical Correlation Analysis (CCA) [7] is a technique for modeling the relationships between two (or more) sets of variables. CCA computes a low-dimensional shared embedding of both sets of variables such that the correlations among the variables between the two sets is maximized in the embedded space. It has been applied with great success in the past on a variety of learning problems dealing with multi-modal data or multi view data [8]. However, Canonical Correlation Analysis is a linear feature extraction algorithm. In real world applications, the data usually exhibit nonlinearities, and therefor a linear projection like CCA may not be able to capture the properties of the data. To deal with the nonlinearities, kernel method has been successfully used in many applications (e.g. Support Vector Machines and Kernel Principal Component Analysis). [9], [10] apply the kernel method to CCA, which first maps each D dimensional data point x to a higher dimensional space $\mathcal{F}$ defined by a mapping function ϕ whose range is in an inner product space, then applies linear CCA in the feature space $\mathcal{F}$.

B. A Clustering Algorithm with Collective Kernel Learning and KCCA

In this section, we will describe a clustering algorithm based on collective kernel learning and KCCA. Given two incomplete datasets X and Y, the goal is to derive a clustering solution $\mathcal{S}$ based on the information contains in both datasets. The algorithm is shown in Algorithm 2.

We first apply the collective kernel learning to complete the two full kernel matrices. Then we use Kernel CCA to find the projected feature space, in which the correlation of the two datasets is maximized, and get the projected two datasets. In case that the dimension of the projected feature space is still too large for clustering, we apply Principal Component Analysis (PCA) to the projected datasets if needed. The clustering solution $\mathcal{S}$ can be acquired using any standard clustering algorithm, like k-means.

Algorithm 2 Clustering using Collective Kernel Learning and KCCA

Input: Incomplete Datasets X and Y.
Output: The clustering solution $\mathcal{S}$.

$[K_x, K_y] = CoKL(X, Y)$.
$[X_p, Y_p] = KCCA(K_x, K_y)$. {$X_p$ and Y_p are the projected datasets.}
if The feature space is still too large **then**
  $X_p = PCA(X_p)$.
  $Y_p = PCA(Y_p)$.
end if
Apply k-means to the projected datasets X_p and Y_p.

V. Experiments and Results

In this section, we analyze the proposed clustering algorithm on two sets of datasets.

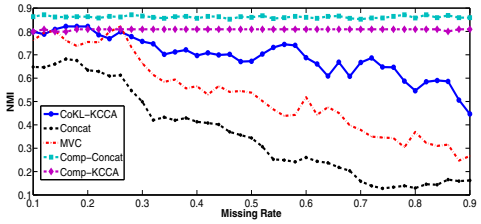
A. Comparison Approaches and Evaluation Strategy

Since there was no previous method that can be directly used to handle the same problem, the first comparison algorithm is straightforward one. The comparison strategy is to first fill the missing features with average values for continuous features and majority values for discrete features, and then concatenate all features together, referred as Concat. In other words, given two incomplete datasets X and Y, we just fill the missing features and get X_c and Y_c. The concatenated features can be represented as follows:

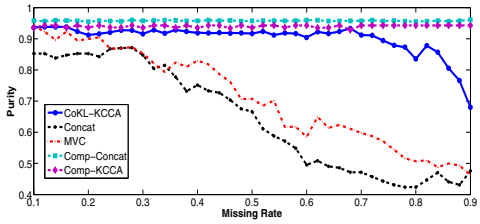
$$F_{XY} = [X_c^T, Y_c^T]^T. \tag{7}$$

So any traditional clustering algorithm can be applied on the concatenated datasets to obtain a solution. Another comparison Approach is the algorithms in [3] referred as MVC. This algorithm assumes at least one dataset is complete. To apply this algorithm to the incomplete datasets, we complete one dataset with average values for continuous features and majority values for discrete features, leaving other datasets incomplete. We also compare the proposed CoKL+KCCA with simple concatenation of complete datasets(Comp-Concat) and KCCA for complete kernels(Comp-KCCA). To show the effectiveness of KCCA, we compare the proposed algorithm (CoKL+KCCA) with Kernel Addition/Production + Spectral Clustering referred as CoKL-KA-SC and CoKL-KP-SC. Although the proposed clustering algorithm could work with any standard clustering algorithm, in all the experiments, we use k-means as the clustering algorithm for convenience.

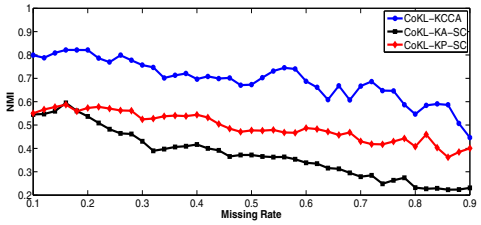
In order to evaluate the quality of the proposed clustering algorithm, we use normalized mutual information (NMI) and the average purity. Note that NMI equals to zero when clustering algorithm is random, and it is close to one when the clustering result is good. Average purity is also close to



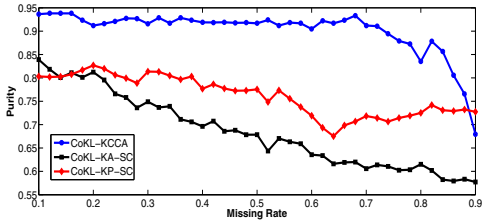
(a) NMI for seeds dataset on different missing rates.



(b) Average purity for seeds dataset on different missing rates.



(c) NMI for seeds dataset on different missing rates.



(d) Average purity for seeds dataset on different missing rates.

Fig. 2: The performance of seeds dataset on different missing rates.

one when the clustering result is good. Note that k-means is sensitive to initial seed selection. Hence, we run k-means 30 times on each parameter setting, and report the averaged NMI and purity with mean value. All the datasets we use in the experiments are complete, but we randomly delete some of the instances in datasets. It is also important to note that the missing rate for each dataset is equal, i.e., two datasets have the same number of missing instances. Since all the original datasets are complete, to generate a missing rate of 60% on a pair of datasets, we randomly select 60% of the instances and delete them alternately from one of the datasets. This will make all the datasets have equal missing rate. We test the performance of the proposed algorithm for different total missing rates (from 10% to 90%).

B. UCI Seeds Datasets

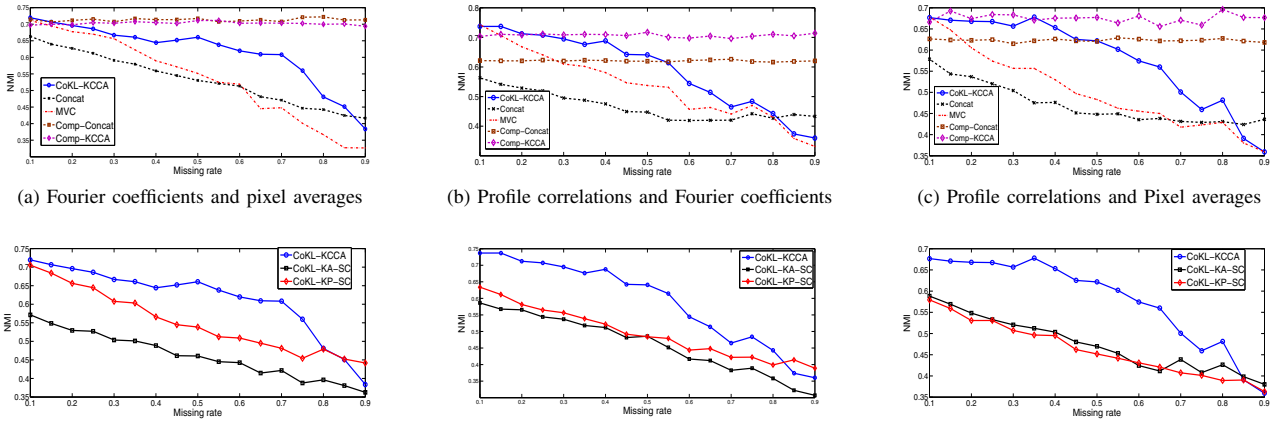
The first dataset contains 210 instances with 7 features. Each instance represents a seed belonging to one of the three different varieties of wheat. The aim is to cluster the seeds. In order to test the performance of the proposed algorithm, we randomly split the feature set into two disjoint parts, which represent two datasets. Then we randomly delete the instances in both of the datasets to make them incomplete. The results average over 30 runs are presented in Fig. 2.

Fig. 2a and Fig. 2b compare CoKL+KCCA with Concat, MVC, and two algorithms on complete data (Comp-Concat and Comp-KCCA). Fig. 2c and Fig. 2d compare different algorithms combined with CoKL on incomplete data (CoKL+KCCA, CoKL-KA-SC and CoKL-KP-SC). As it can be observed in Fig. 2a and Fig. 2a, the proposed algorithm, clustering with CoKL and KCCA, outperforms the two comparison methods (Concat and MVC) substantially for all the missing rates in both NMI and average purity. For example, when the missing rate is 0.7, the NMI obtained from CoKL+KCCA is about 0.7, while that of the comparison methods is only about 0.3. The average purity obtained from CoKL+KCCA is about 0.85, while that of the comparison methods is only less than 0.65. Even when the missing rate is 0.9, the NMI obtained from CoKL+KCCA is still 0.43, which is much larger than that of the comparison methods. Of course, the result of proposed algorithm is not as good as the results of algorithms running on complete dataset(Comp-Concat and Comp-KCCA). However, it is important to note that in Fig. 2b the proposed algorithm is very closed to the algorithms running on complete dataset in average purity. From Fig. 2c and Fig. 2d, it can be easily observed that CoKL+KCCA outperforms CoKL-KA-SC and CoKL-KP-SC almost everywhere in both NMI and average purity, which shows the effectiveness of KCCA. These results shows that CoKL+KCCA performs not only better than the intuitive strategy which directly uses the concatenated features, but also better than the latest method MVC.

C. Handwritten Dutch Numbers Recognition

This dataset contains 2000 handwritten numerals (”0”-”9”) extracted from a collection of Dutch utility maps. The handwritten numbers are scanned and digitized as binary images. The following feature spaces (datasets) with different vector-based features is available for the numbers: (1) 76 Fourier coefficients of the character shapes, (2) 216 profile correlations, (3) 240 pixel averages in 2×3 windows, and (4) 47 Zernike moments. All these features are conventional vector-based features but in different feature spaces. The aim is to cluster the numbers. We test the proposed algorithm on two incomplete datasets. Among this 4 different datasets, we can have 6 different pairs. For each pair of datasets, we randomly delete the instances in both of the datasets. Part of the results for all the 6 different pairs average over 30 runs are presented in Fig. 3. The complete results can be found in [4].

Fig. 3a-3c compare CoKL+KCCA with Concat, MVC, and



(a) Fourier coefficients and pixel averages (b) Profile correlations and Fourier coefficients (c) Profile correlations and Pixel averages

(d) Fourier coefficients and pixel averages (e) Profile correlations and fourier coefficients (f) Profile correlations and pixel averages

Fig. 3: NMI and average purity on 6 pairs of handwritten Dutch numbers datasets

two algorithms on complete data (Comp-Concat and Comp-KCCA). Fig. 3d-3f compare different algorithms combined with CoKL on incomplete data (CoKL+KCCA, CoKL-KA-SC and CoKL-KP-SC). As it can be observed, the proposed algorithm, clustering with CoKL and KCCA, outperforms the two comparison methods (Concat and MVC) substantially for the three pairs of datasets in NMI. Taking the pair Fourier coefficients dataset and pixel average dataset (Fig. 3a) as example, the NMI obtained from CoKL+KCCA is 0.63 at missing rate 0.7, while that of the comparison methods is only about 0.45. One interesting result is that CoKL+KCCA is even better than Comp-Concat for some settings like Fig. 3b and Fig 3c. The reason is because even with complete dataset, the correlations among the two dataset may not be significant. However, the correlations among the projected spaces between the two sets are maximized when apply KCCA. So CoKL+KCCA could be better than Comp-Concat for some settings but is worse than Comp-KCCA for almost every setting. From Fig. 3d-3f, it can be easily observed that CoKL+KCCA outperforms CoKL-KA-SC and CoKL-KP-SC for most of the cases, which shows the effectiveness of KCCA. These results shows that on incomplete datasets, CoKL+KCCA performs not only better than the intuitive strategy Concat and advanced method MVC, but even better than some simple algorithms on complete datasets.

VI. Conclusion

In this paper, we study the problem of clustering for multiple incomplete datasets. We propose a CoKL principle to deal with the incompleteness of the datasets by collectively completing the kernel matrices of the datasets using the common instances in different datasets. An optimization problem is derived from the CoKL principle to optimize the alignment of incomplete kernel matrices, and an approximation solution is obtained by iteratively solving a constrained optimization problem. Furthermore, we propose a clustering algorithm using CoKL and KCCA. By applying KCCA after CoKL, the proposed algorithm could maximize the correlation between the projected feature spaces, which will increase the performance of clustering compared with other methods. Two sets of experiments were performed to evaluate the clustering algorithm. It can be clearly observed that the proposed algorithm outperforms the comparison algorithms by as much as twice in both NMI and average purity. Further discussion about the convergency and efficiency of the proposed algorithm can be found in [4].

Acknowledgement This work is supported in part by NSF through grants CNS-1115234, DBI-0960443, and OISE-1129076, US Department of Army through grant W911NF-12-1-0066, and Huawei Grant.

References

[1] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, ser. COLT' 98. New York, NY, USA: ACM, 1998, pp. 92–100.

[2] S. Bickel and T. Scheffer, "Multi-view clustering," in *ICDM*. IEEE Computer Society, 2004, pp. 19–26.

[3] A. Trivedi, P. Rai, H. Daumé III, and S. DuVall, "Multiview clustering with incomplete views," in *NIPS 2010: Workshop on Machine Learning for Social Computing, Whistler, Canada*, 2010.

[4] W. Shao, X. Shi, and P. S. Yu, "Clustering on multiple incomplete datasets via collective kernel learning," *arXiv preprint 1310.1177 [cs.LG]*, 2013.

[5] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *COLT*, ser. Lecture Notes in Computer Science, B. Schölkopf and M. K. Warmuth, Eds., vol. 2777. Springer, 2003, pp. 144–158.

[6] R. K. Ando and T. Zhang, "Learning on graph with laplacian regularization," in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 25–32.

[7] R. A. Johnson and D. W. Wichern, Eds., *Applied multivariate statistical analysis*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[8] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view clustering via canonical correlation analysis," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 129–136.

[9] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," *International Journal of Neural Systems*, vol. 10, no. 05, pp. 365–377, 2000.

[10] S. Akaho, "A kernel method for canonical correlation analysis," *arXiv preprint cs/0609071*, no. 4, pp. 1–7, 2006.