Multiview Clustering with Incomplete Views

Piyush Rai
School of Computing
University of Utah
piyush@cs.utah.edu

Anusua Trivedi
School of Computing
University of Utah
anusua@cs.utah.edu

Hal Daumé III
Dept. of Computer Science
University of Maryland
hal@umiac.umd.edu

Scott L. DuVall
VA SLC Health Care System &
University of Utah
Salt Lake City, Utah, USA
scott.duvall@utah.edu

Abstract

Multiview clustering algorithms allow leveraging information from multiple views of the data and therefore lead to improved clustering. A number of kernel based multiview clustering algorithms work by using the kernel matrices defined on the different views of the data. However, these algorithms assume availability of features from all the views of each example, i.e., assume that the kernel matrix for each view is complete. We present an approach that allows these algorithms to be applicable even when only one (the primary) view is complete and the auxiliary views are incomplete (i.e., features from these views are available only for some of the examples). Taking the kernel CCA based multiview clustering as an example, we apply our method on webpage clustering with multiple views of the data where one view is the page-text and other view is the social tags assigned to the webpage. We consider the case when the tags are available only for a small subset of the webpages which means that the tag view is incomplete. Experimental results establish the effectiveness of the proposed method.

1 Introduction

Explosion of the social-web has led to enormous amounts of data being generated everyday, through blogs, social bookmarking websites (e.g., Delicious, StumbleUpon), microblogging services (e.g., Twitter), etc. Often, such data is associated with auxiliary sources of information, apart from the usual page-text information, for example, social tags, hyperlink information, citation information, etc. Each source of information is essentially a view of the data, and learning with such type of data is commonly referred to as multiview learning [3, 7, 1]. A number of algorithms, both supervised and unsupervised have been proposed in the recent past to exploit multiple views of the data. However, most existing multiview learning algorithms suffer from the shortcoming that they require all the views to be complete, i.e., present for all the examples. This may however not always be the case. For example, in the context of social bookmarking datasets, user tags may be available only for a small subset of webpages. One way to apply multiview learning algorithms in this setting would be to first try to predict the set of social tags for each non-tagged webpage. This can however be very expensive since the set of possible tags (“labels”) can be really large (equal to the tag vocabulary size).

This limitation makes it necessary to develop multiview algorithms that can work with incomplete view information. In this paper, we present an approach based on reconciling the similarities between examples across all views to address this shortcoming of multiview learning algorithms. In
particular, we show how the Canonical Correlation Analysis (CCA) based approach to multiview clustering [7, 18] can be used in situations when only one view (the primary view) is complete whereas the other view(s) could potentially be incomplete, i.e., features from such view(s) are available only for a small number of examples. Our approach does not require computing the explicit features in the incomplete views (e.g., we do not require the tags to be predicted for the non-tagged webpages). In particular, we take the kernel variant of CCA [11] which works on the kernel matrices defined over each view, and propose a way to construct the full kernel matrix corresponding to the incomplete view, given the other complete view. This is followed by applying the kernel CCA based multiview clustering algorithm [18]. Our presentation is based on the kernel CCA based multiview clustering but our approach can also be applied to other kernel based multiview clustering algorithms [9].

2 Multiview Clustering using Low-Dimensional Projections

A naïve approach to do multiview clustering would be to simply concatenate the feature vectors of each view (e.g., words feature vector and tag feature vector for each webpage), and run any standard clustering algorithm such as the $k$-means algorithm. However, the concatenation approach inflates the feature vector size of each document, and therefore the approach tends to not do well if the number of webpages is small as compared to the feature dimensionality [12]. The reason can be attributed to the fact that clustering, and density estimation in general, can yield poor parameter estimates if the number of features far exceeds the number of data points. Besides, the relative importance of features in the tag view and word view of the concatenated vector can be different which may require an explicit weighting of features in the two views [14]. A number of efficient clustering algorithms deal with high data dimensionality by first projecting the data onto a lower dimensional subspace, and then performing clustering in that subspace. The projection step is usually performed using standard dimensionality reduction techniques such as principal component analysis [19] (PCA), or random projections [8]. However, PCA or random projections only preserve the data variances or pairwise distances and fail to take advantage of multiple views of the data (if such information is available). Even if PCA is performed on the joint words + tags vector, it would only maximize the variances of word and tag feature spaces individually, without capturing their correlations.

The CCA based projection approach, on the other hand, computes features shared across all the views by maximizing the view correlations in the projected space. These features can then be used with any off-the-shelf clustering algorithm such as the $k$-means algorithm. Such an approach has been recently used with reasonable empirical success and theoretical guarantees [7]. In the next section, we briefly describe CCA and its kernel variant, Kernel CCA. Then, in Section 4, we will describe how the kernel CCA approach can be used with incomplete views.

3 Kernel CCA

Canonical Correlation Analysis (CCA) is a technique for modeling the relationships between two (or more) set 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. CCA has been applied with great success in the past on a variety of learning problems dealing with multi-modal data [10, 11, 15]. Canonical Correlation Analysis is a linear feature extraction algorithm. Many real world datasets, however, exhibit nonlinearities, and therefore a linear projection may not be able to capture the properties of the data. Kernel methods [17] give us a way to deal with the nonlinearities by mapping the data to a higher (potentially infinite) dimensional space and then applying linear methods in that space (e.g., Support Vector Machines [5] for classification, Kernel Principal Component Analysis [16] for dimensionality reduction, etc.). The attractiveness of kernel methods is attributed to the fact that this mapping need not be computed explicitly, via the technique called the kernel trick [17].

The kernel variant of CCA (called Kernel Canonical Correlation Analysis - KCCA) can be thought of as first (implicitly) mapping each $D$ dimensional data point $x$ to a higher dimensional space $F$ defined by a mapping $\phi$ whose range is in an inner product space (possibly infinite dimensional), followed by applying linear CCA in the feature space $F$. 

2
More formally, given a pair of datasets \( X \in \mathbb{R}^{D_1 \times N} \) and \( Y \in \mathbb{R}^{D_2 \times N} \) (one can think of these as two views of the same data having \( N \) paired examples), CCA seeks to find linear projections \( w_x \in \mathbb{R}^{D_1} \) and \( w_y \in \mathbb{R}^{D_2} \) such that, after projecting, the corresponding examples in the two datasets are maximally correlated in the projected space. To get the kernel formulation of CCA, we switch to the dual representation [11] by expressing the projection directions as \( w_x = X\alpha \) and \( w_y = Y\beta \) where \( \alpha \) and \( \beta \) are vectors of size \( N \). In the dual formulation, the correlation coefficient between \( X \) and \( Y \) can be written as: \( \rho = \max_{\alpha, \beta} \frac{\alpha^T X^T X \alpha \beta^T Y^T Y \beta}{\sqrt{\alpha^T X^T X \alpha \beta^T Y^T Y \beta}}. \)

Now using the fact that \( K_x = X^T X \) and \( K_y = Y^T Y \) are the kernel matrices for \( X \) and \( Y \), kernel CCA amounts to solving the following problem: \( \rho = \max_{\alpha, \beta} \frac{\alpha^T K_x \alpha \beta^T K_y \beta}{\sqrt{\alpha^T K_x \alpha \beta^T K_y \beta}} \) subject to the following constraints: \( \alpha^T K_x^2 \alpha = 1 \) and \( \beta^T K_y^2 \beta = 1. \)

KCCA works by using the kernel matrices \( K_x \) and \( K_y \) of the examples in the two views \( X \) and \( Y \) of the data. This is in contrast with linear CCA which works by doing an eigen-decomposition of the covariance matrix. The eigenvalue problem for kernel CCA is given by:

\[
\begin{pmatrix}
0 & K_x K_y \\
K_y K_x & 0
\end{pmatrix}
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix} = \lambda
\begin{pmatrix}
K_x^2 & 0 \\
0 & K_y^2
\end{pmatrix}
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
\]

For the case of linear kernel, KCCA reduces to the standard CCA. However, working under the kernel formalism has the additional advantage of being computationally efficient if the number of features greatly exceeds the number of examples because KCCA works on \( N \times N \) kernel matrices, whereas CCA works on \( D \times D \) covariance matrices. The former would be much more efficient than the latter if \( D \gg N \), which is usually the case with webpage clustering where the vocabulary size can often far exceed the number of webpages. To avoid overfitting and trivial solutions (non-relevant solutions), CCA literature [17, 11] suggests regularizing the projection directions \( w_x \) and \( w_y \) by penalizing them using Partial Least Squares (PLS) which basically means that their high weights are penalized. KCCA can also be easily generalized to more than two views [2].

### 4 Kernel CCA with Incomplete Views

One shortcoming of both CCA and KCCA is that they assume that features across all views are available for each example. This may however not be the case with many multiview datasets. For example, not all webpages in a corpus might be tagged by users. Likewise, not all webpages can be expected to have hyperlinks pointing towards them. Therefore, although one view (i.e., page-text) would be available for all the webpages, the other view might be available only for a small number of webpages. To apply multiview clustering on such datasets, one needs a way to deal with the lack of data in the incomplete view(s). In this section, we present an approach to address this shortcoming for KCCA. The problem for standard CCA can also be dealt with by using KCCA with a linear kernel. Also, our approach is not limited to kernel CCA based multiview clustering. It can also be used for other kernel based multiview clustering algorithms such as the multiview spectral clustering [9].

Note that KCCA works by first constructing the kernel matrix for each view of the data. For simplicity, let us denote the two views by \( \mathcal{X} \) and \( \mathcal{Y} \). Generalization to more than two views with one complete and remaining incomplete views can be done in a likewise manner. Let us assume that view \( \mathcal{X} \) is complete whereas view \( \mathcal{Y} \) is incomplete, i.e., the features for this view are available for only a subset of the total examples. To formalize, we denote the set of webpages with features present in both the views \( \mathcal{X} \) and \( \mathcal{Y} \) (i.e., the fully paired or complete) as \( C = \{(x_1, y_1), \ldots, (x_c, y_c)\} \) and the set of webpages with features present only in view \( \mathcal{X} \) (i.e., unpaired or missing) as \( \mathcal{M} = \{x_{c+1}, \ldots, x_{c+m}\} \). Let us denote by \( K_x \) the \((c+m) \times (c+m) \) kernel matrix defined over all the examples using features from view \( \mathcal{X} \). The corresponding graph Laplacian defined as \( L_x = D_x - K_x \).

![Figure 1: Completing the full kernel matrix using the incomplete view \( \mathcal{Y} \)](image)
where $D_x$ is the diagonal matrix consisting of the row sums of $K_x$ along its diagonals.

Likewise, for view $\mathcal{Y}$, we denote the kernel matrix by $K_y$. However, since features for view $\mathcal{Y}$ are only available for a small number of examples, only an $c \times c$ subblock of the full kernel matrix $K_y$ will be available for this view (see Figure 1). In order to apply kernel CCA, one must first construct the full kernel matrix $K_y$. Using the ideas from Laplacian regularization, this can be achieved by solving the following optimization problem for kernel matrix completion:

$$
\min_{K_y \succeq 0} tr(\mathcal{L}_y K_y) \quad (2)
$$

$$
s.t. K_y(i, j) = k(y_i, y_j), \forall 1 \leq (i, j) \leq c
$$

The objective above optimizes w.r.t. $K_y$ the alignment between $K_x$ and $K_y$, given the known part of $K_y$. Here $tr$ denotes the matrix trace. Note that although the multiview assumption requires the views to be conditionally independent, since both views are just expressing different representations of the same object, both kernel matrices $K_x$ and $K_y$ are still expected to have a high degree of alignment between them.

The positive semi-definite constraint on the kernel matrix $K_y$ makes it a semi-definite program (SDP) [4], which can be solved using the existing SDP solvers. One problem with the SDP based solvers is their lack of scalability to a large number of examples. Although the scalability can still be dealt with using first order solvers such as SDPNAL [20], assessing convergence can be an issue with such approaches. In this paper, we take a different approach and, due to the special problem structure (i.e., upper left sub-block of $K_y$ being known), we can in fact obtain a closed-form solution for $K_y$. Furthermore, our approach is much less computationally intensive than having to solve an SDP since, as we will show, it only requires a couple of matrix multiplication and inversions. A similar approach was proposed in [6] to compute the embedding of out-of-sample datapoints in Laplacian eigenmaps.

We denote $K_y(i, j) = k(y_i, y_j), \forall 1 \leq (i, j) \leq c$ in Equation (2) as $K_y^{cc}$, the $c \times c$ kernel matrix for the set of examples with view $\mathcal{Y}$ available.

Since $K_y$ is a positive semi-definite matrix, we can express it as $A A^T$ where $A$ is a matrix of reals. Further, let us write $A$ as $A = \begin{pmatrix} A_c & A_m \end{pmatrix}$, and $\mathcal{L}_x$ as:

$$
\mathcal{L}_x = \begin{bmatrix} \mathcal{L}_x^{cc} & \mathcal{L}_x^{cm} \\ \mathcal{L}_x^{cm^T} & \mathcal{L}_x^{mm} \end{bmatrix}
$$

Using these, we can rewrite Equation (2) as follows:

$$
\min_A tr(\mathcal{L}_x A A^T) = \min_A tr(A^T \mathcal{L}_x A) = \min_{A_c, A_m} tr\left( \begin{bmatrix} A_c & A_m \end{bmatrix}^T \begin{bmatrix} \mathcal{L}_x^{cc} & \mathcal{L}_x^{cm} \\ \mathcal{L}_x^{cm^T} & \mathcal{L}_x^{mm} \end{bmatrix} \begin{bmatrix} A_c & A_m \end{bmatrix} \right)
$$

Simplifying the above, and using the fact that $A_c$ is a constant (since $A_c A_c^T = K_y^{cc}$, a constant), gives:

$$
\min_{A_m} tr(A_c^T \mathcal{L}_x A_c + A_c^T \mathcal{L}_x A_m + A_m^T (\mathcal{L}_x^{cm})^T A_c + A_m^T \mathcal{L}_x^{mm} A_m)
$$

Using the matrix trace property $tr(X) = tr(X^T)$, one can see that the above reduces to:

$$
\min_{A_m} tr(A_c A_c^T \mathcal{L}_x^{cc}) + 2 * tr(A_c^T \mathcal{L}_x^{cm} A_m) + tr(A_m^T \mathcal{L}_x^{mm} A_m)
$$

Again, using the fact $A_c A_c^T = K_y^{cc}$, we write the above as:

$$
\min_{A_m} tr(K_y^{cc} \mathcal{L}_x^{cc}) + 2 * tr(A_c^T \mathcal{L}_x^{cm} A_m) + tr(A_m^T \mathcal{L}_x^{mm} A_m) = \min_{A_m} 2 * tr(A_c^T \mathcal{L}_x^{cm} A_m) + tr(A_m^T \mathcal{L}_x^{mm} A_m)
$$
Taking the derivative w.r.t. $A_m$ and setting it to zero gives:

$$2 * (L_x^{cm})^T A_c + 2 * L_x^{mm} A_m = 0$$

Therefore $A_m = -(L_x^{mm})^{-1}(L_x^{cm})^T A_c$, and $A = (A_c \ A_m) = \left( -(L_x^{mm})^{-1}(L_x^{cm})^T A_c \right)$

Using $K_y = AA^T$ gives us the closed-form expression for $K_y$:

$$K_y = \left( -(L_x^{mm})^{-1}(L_x^{cm})^T A_c \ A_c^T \right) -(A_c A_c^T L_x^{cm} (L_x^{mm})^{-1}) \ A_c A_c^T \ A_c^T L_x^{cm} (L_x^{mm})^{-1})$$

Finally, substituting back for $A_c A_c^T = K_{cc}$ gives:

$$K_y = \left( -(L_x^{mm})^{-1}(L_x^{cm})^T K_{cc} \ K_{cc}^- (L_x^{mm})^{-1}) \ (L_x^{mm})^{-1}(L_x^{cm})^T K_{cc}^- K_{cc}^- (L_x^{mm})^{-1})$$

Having obtained the full kernel matrix $K_y$ for all $c + m$ examples on view $\mathcal{Y}$, we can now apply kernel CCA on the two kernel matrices $K_x$ and $K_y$, and use the extracted features in any off-the-shelf clustering algorithm such as $k$-means.

5 Experiments

We experimented with a webpage clustering task where the data consists of two views: the page-text and the social tags assigned to each webpage. Our dataset consists of a collection of 1000 tagged webpages. All webpages in our collection were downloaded from URLs that are present in both the Open Directory Project (http://www.dmoz.org/) web directory (so that their ground-truth clustering are available for evaluations), and the Delicious social bookmarking website (so that their tag information is available). After stemming and stop-word removal, bag-of-words was used for both page-text and tag-based feature vector representations. To simulate the incomplete view setting, we provide our algorithm the tag features for only a small fraction of webpages in the corpus. For the remaining webpages, we only use the page-text based features. We call webpages with both page-text and tag information available as paired, and webpages with only page-text information available as non-paired. In our experiment, we vary the fraction of paired webpages from 10% to 60%.

We compare our kernel-completion-followed-by-KCCA based approach with two baselines. Our first baseline is KCCA with full view information [18], i.e., all the webpages are paired with their corresponding tag information. Our second baseline is an incomplete view setting like ours: KCCA on paired webpages but Kernel PCA on non-paired webpages (since only a single view, page-text, is available for non-paired webpages). The $k$-means algorithm was used as the base clustering algorithm in our approach as well as the other baselines. However, any other clustering algorithm can be used as well. Since $k$-means might be sensitive to initializations, we run it 20 times and report the mean and standard deviations. Gaussian kernel was used for all the kernel computations and the width parameter was set to the median pairwise distance between the examples. For the evaluation of clustering performance: we used the following measures:

- **F-score**: It is defined as the harmonic mean of Precision and Recall scores.
- **Average Cluster Entropy**: It is based on the impurity of a cluster given the true classes in the data. If $p_{ij}$ be the fraction of class $j$ in obtained cluster $i$, $N_i$ be the size of cluster $i$, and $N$ be the total number of examples, then the average cluster entropy is defined as:

$$E = \sum_{i=1}^{K} \frac{N_i}{N} \left( - \sum_{j} p_{ij} \log(p_{ij}) \right)$$

The performance of our approach and the other baselines is shown in Figure 2. As we can see, our approach with incomplete view with just about 50% to 60% paired webpages achieves comparable results to the fully paired KCCA case. With fractions higher than that, we observed the performance to wiggle around and stay roughly the same (moderately better or moderately worse) to the fully paired case. On the other hand, it significantly outperforms the other baseline that uses KCCA on the paired webpages and kernel PCA on the non-paired webpages. The inferior performance of the KCCA+KPCA approach can be attributed to the fact that only a small subset of the webpages use the tag information for the low-dimensional projection.
We also experimented to see how good is the reconstructed tag view kernel matrix with respect to the ground truth kernel matrix of tag features. To do this, we vary the fraction of paired examples as did in the previous experiment and plot the alignment of matrices in both views. As we can see from Figure 3, the alignment gets better as the fraction of paired examples increases, and with about 55-60% paired examples, the alignment is almost as high as the alignment obtained on the ground truth kernel matrix.

Note that merely having a high alignment between $K_x$ and $K_y$ does not ensure that the multiview clustering performance will be good. If the number of examples in view $\mathcal{Y}$ is very small, then the optimization could give a kernel matrix $K_y$ that may be very similar to $K_x$ and it may not give any useful information from view $\mathcal{Y}$. Therefore one needs a sufficient number of examples from view $\mathcal{Y}$ so that the obtained kernel matrix $K_y$ actually is a good representative of the similarities between examples in view $\mathcal{Y}$. As shown in our experiments, about 50% to 60% tagged webpages gave close to optimal performance. Another thing to note here is that the reconstructed kernel matrix $K_y$ in our approach depends on the kernel $K_{cc}^y$ constructed using tagged set of webpages so the reconstruction accuracy (and hence the clustering performance) depends on how good is $K_{cc}^y$. With a small but reasonably well tagged subset of the whole data, we expect the reconstructed $K_y$ to be sufficiently close to the optimal kernel matrix in view $\mathcal{Y}$.

6 Related Work and Conclusion

In the context of webpage clustering, a number of earlier approaches have considered views other than just the page-text information. The use of social-tags in particular has been proposed before in [14, 13]. However, all these approaches assume that the data is fully paired, which is not a restriction with the approach we proposed in this paper. Another way to deal with the incomplete view setting could be to use probabilistic approaches. For example, the multiview clustering algorithm proposed in [1] uses a two-view Expectation Maximization algorithm. The proposed algorithm in [1] however uses fully paired data. Although EM has its own shortcomings such as the issue of convergence to a local maxima, it should nevertheless be possible to extend multi-view EM to treat the missing features as latent variables and estimate them using the rest of the data. In future we plan to apply the proposed approach for exploiting other (possibly incomplete) views of webpage data such as the hyperlink information, and information retrieval with cross-lingual data on the web, where the corresponding translations of some documents in other languages may be missing. Finally, our proposed approach to multiview clustering with incomplete views is generic. Here we applied it on Kernel CCA based multiview clustering. However, our proposed approach can also be applied to other multiview clustering algorithms that work with kernel matrices defined over multiple views of the data, for example the multiview spectral clustering [9].

Acknowledgements This work is supported by resources and facilities of the VA Salt Lake City Health Care System with funding support from the Consortium for Healthcare Informatics Research (CHIR), VA HSR HIR 08-374 and the VA Informatics and Computing Infrastructure (VINCI), VA HSR HIR 08-204. PR was partially supported by NSF grant IIS-0712764.
References