Multi-View Clustering with Constraint Propagation for Learning with an Incomplete Mapping Between Views

Eric Eaton
Bryn Mawr College[†]
Computer Science Department
Bryn Mawr, PA 19010
eeaton@brynmawr.edu

Marie desJardins
University of Maryland Baltimore County
CSEE Department
Baltimore, MD 21250
mariedj@umbc.edu

Sara Jacob
Lockheed Martin Advanced Technology Laboratories
AI Research Group
Cherry Hill, NJ 08002
sjacob@atl.lmco.com

ABSTRACT

Multi-view learning algorithms typically assume a complete bipartite mapping between the different views in order to exchange information during the learning process. However, many applications provide only a partial mapping between the views, creating a challenge for current methods. To address this problem, we propose a multi-view algorithm based on constrained clustering that can operate with an incomplete mapping. Given a set of pairwise constraints in each view, our approach propagates these constraints using a local similarity measure to those instances that can be mapped to the other views, allowing the propagated constraints to be transferred across views via the partial mapping. It uses co-EM to iteratively estimate the propagation within each view based on the current clustering model, transfer the constraints across views, and update the clustering model, thereby learning a unified model for all views. We show that this approach significantly improves clustering performance over several other methods for transferring constraints and allows multi-view clustering to be reliably applied when given a limited mapping between the views.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

algorithms, experimentation

Keywords

constrained clustering, multi-view learning, semi-supervised learning

[†]This work was conducted while the first author was at Lockheed Martin Advanced Technology Laboratories and the University of Maryland Baltimore County.

1. INTRODUCTION

Using multiple different views often has a synergistic effect on learning, improving the performance of the resulting model beyond learning from a single view. Multi-view learning is especially relevant to applications that simultaneously collect data from different modalities, with each unique modality providing one or more views of the data. For example, a textual field report may have associated image and video content, and an Internet web page may contain both text and audio. Each view contains unique complementary information about an object; only in combination do they yield a complete representation of the original object. Concepts that are challenging to learn in one view (e.g., identifying images of patrons at an Italian restaurant) may be easier to recognize in another view (e.g., via the associated textual caption), providing an avenue to improve learning. Multi-view learning can share learning progress in a single view to improve learning in the other views via the direct correspondences between views.

Current multi-view algorithms typically assume that there is a complete bipartite mapping between instances in the different views to represent these correspondences, denoting that each object is represented in all views. The predictions of a model in one view are transferred via this mapping to instances in the other views, providing additional labeled data to improve learning. However, what happens if we have only a partial mapping between the views, where only a limited number of objects have multi-view representations?

This problem arises in many industrial and military applications, where data from different modalities are often collected, processed, and stored independently by specialized analysts. Consequently, the mapping between instances in the different views is incomplete. Even in situations where the connections between views are recorded, sensor availability and scheduling may result in many isolated instances in the different views. Although it is feasible to identify a partial mapping between the views, the lack of a complete bipartite mapping presents a challenge to most current multi-view learning methods. Without a complete mapping, these methods will be unable to transfer any information involving an isolated instance to the other views.

To address this problem, we propose a method for multi-view learning with an incomplete mapping in the context of constrained clustering. Constrained clustering [21, 5, 7, 24] is a class of semi-supervised learning methods that cluster data subject to a set of hard or soft constraints that

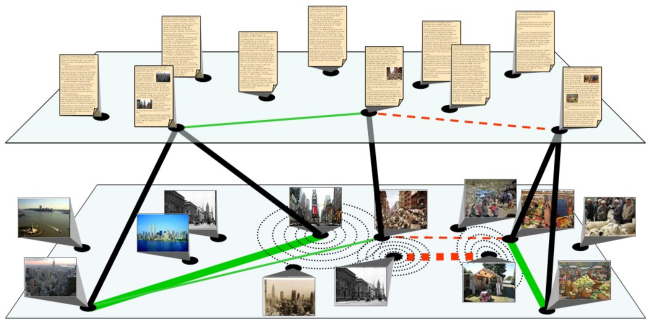


Figure 1: An illustration of multi-view constrained clustering between two disjoint data views: text and images. We are given a very limited mapping between the views (solid black lines) and a set of pairwise constraints in the images view: two must-link constraints (thick solid green lines) and one cannot-link constraint (thick dashed red line). Based on the current clustering, each given constraint is propagated to pairs of images that are in close proximity to the given constraint and can be mapped to the text view. These propagated must-link and cannot-link constraints (thin solid green and dashed red lines, respectively) are then directly transferred via the mapping to form constraints between texts and influence the clustering in the next co-EM iteration.

specify the relative cluster membership of pairs of instances. These constraints serve as background information for the clustering by specifying instance pairs that belong in either the same cluster (a *must-link* constraint) or different clusters (a *cannot-link* constraint). Given a set of constraints in each view, our approach transfers these constraints to affect learning in the other views. With a complete mapping, each constraint has a direct correspondence in the other views, and therefore can be directly transferred between views using current methods. However, with a partial mapping, these constraints may be between instances that do not have equivalences in the other views, presenting a challenge to multi-view learning, especially when the mapping is very limited.

This paper proposes the first multi-view constrained clustering algorithm that considers the use of an incomplete mapping between views. Given an incomplete mapping, our approach propagates the given constraints within each view to pairs of instances that have equivalences in the other views. Since these propagated constraints involve only instances with a mapping to the other views, they can be directly transferred to instances in those other views and affect the clustering. The weight of each propagated constraint is given by its similarity to the original constraint, as measured by a local radial basis weighting function that is based on the current estimate of the clustering. This process is depicted in Figure 1. Our approach uses a variant of co-EM [17] to iteratively estimate the propagation within each view, transfer the constraints across views, and update the clustering model. Our experiments show that using co-EM with constraint propagation provides an effective mechanism for multi-view learning under an incomplete mapping between views, yielding significant improvement over several other mechanisms for transferring constraints across views.

2. BACKGROUND AND RELATED WORK

Our approach combines constrained clustering with multi-view learning. In this section, we review the related work on both of these topics.

2.1 Constrained clustering

Constrained clustering algorithms [21, 5, 7, 24] learn a clustering model subject to a set of constraints $\mathcal{C}$ that specify the relative cluster membership of sets of instances. Depending on the algorithm, this labeled knowledge may be treated as either hard constraints that cannot be violated, as in COP-Kmeans [21], or soft constraints that can be violated with some penalty, as in SCOP-Kmeans [22], PCK-Means [5], and MPCK-Means [7]. In this paper, we focus on soft constrained clustering, where each constraint specifies the relative cluster membership of pairs of points.

A pairwise constraint $\langle x_i, x_j, w, \mathit{type}\rangle \in \mathcal{C}$ denotes the relative clustering of instances x_i and x_j, where the non-negative weight of the constraint is given by $w \in \mathbb{R}_0^+$ (the set of non-negative real numbers) and $\mathit{type} \in \{\mathit{must\text{-}link}, \mathit{cannot\text{-}link}\}$ specifies whether x_i and x_j belong in either the same cluster (must-link) or different clusters (cannot-link). In soft constrained clustering, w can be viewed as the penalty for violating the constraint. Throughout this paper, wherever the weight or type of constraint are obvious from context, we will omit them and indicate a pairwise constraint as simply $\langle x_i, x_j\rangle$ or $\langle x_i, x_j, w\rangle$. For convenience, we refer to the sets of all must-link and cannot-link constraints as, respectively, $\mathcal{C}_{ml}$ and $\mathcal{C}_{cl}$.

Although our approach can use most current constrained clustering algorithms, we focus on the PCK-Means [5] and MPCK-Means [7] algorithms. PCK-Means performs soft constrained clustering by combining the K-Means objective function with penalties for constraint violations. MPCK-Means builds on PCK-Means to learn the distance metrics for each cluster during the clustering process. In the remainder of this section, we provide a brief overview of these methods; further details are available in the original papers.

We first describe the MPCK-Means algorithm, and then show the simplifications that yield PCK-Means. The MPCK-Means algorithm generates a k-partitioning of the data $X \subseteq \mathbb{R}^d$ by minimizing the following objective function, which combines the K-Means model with penalties for violating must-link and cannot-link constraints:

$$\begin{aligned}\mathcal{J}_{MPCK} = & \sum_{x_i \in X} \left(\|x_i - \mu_{x_i}\|^2_{\mathbf{M}_{x_i}} - \mathit{log}(\mathit{det}(\mathbf{M}_{x_i})) \right) \\ & + \sum_{\langle x_i, x_j, w\rangle \in \mathcal{C}_{ml}} w f_{ml}(x_i, x_j) \mathbb{1}(\mu_{x_i} \neq \mu_{x_j}) \\ & + \sum_{\langle x_i, x_j, w\rangle \in \mathcal{C}_{cl}} w f_{cl}(x_i, x_j) \mathbb{1}(\mu_{x_i} = \mu_{x_j}) \ ,\end{aligned}$$

where

$$\begin{aligned} f_{ml}(x_i, x_j) &= \tfrac{1}{2}\|x_i - x_j\|^2_{\mathbf{M}_{x_i}} + \tfrac{1}{2}\|x_i - x_j\|^2_{\mathbf{M}_{x_j}} \\ f_{cl}(x_i, x_j) &= \|x'_{x_i} - x''_{x_i}\|^2_{\mathbf{M}_{x_i}} - \|x_i - x_j\|^2_{\mathbf{M}_{x_i}} \ , \end{aligned}$$

μ_{x_i} and $\mathbf{M}_{x_i}$ are respectively the centroid and metric of the cluster to which x_i belongs, x'_{x_i} and x''_{x_i} are the points with the greatest separation according to the $\mathbf{M}_{x_i}$ metric, the function $\mathbb{1}(b) = 1$ if predicate b is true and 0 otherwise, and $\|x_i - x_j\|_{\mathbf{M}} = \sqrt{(x_i - x_j)^{\intercal}\mathbf{M}(x_i - x_j)}$ is the Mahalanobis distance between x_i and x_j using the metric $\mathbf{M}$. The first term of $\mathcal{J}_{MPCK}$ attempts to maximize the log-likelihood of the K-Means clustering, while the second and third terms incorporate the costs of violating constraints in $\mathcal{C}$.

MPCK-Means uses expectation-maximization (EM) to locally minimize $\mathcal{J}_{MPCK}$ to generate the clustering. The E-step consists of assigning each point to the cluster that minimizes $\mathcal{J}_{MPCK}$ from the perspective of that data point, given the previous assignments of points to clusters. The M-step consists of two parts: re-estimating the cluster centroids given the E-step cluster assignments, and updating the metric matrices $\{\mathbf{M}_h\}_{h=1}^K$ to decrease $\mathcal{J}_{MPCK}$. The latter step enables MPCK-Means to learn the metrics for each cluster in combination with learning the constrained clustering model. Learning a Mahalanobis metric has also been considered by Xing et al. [24] and Bar-Hillel et al. [2]. The PCK-Means algorithm is a simplified form of this approach that minimizes the same objective function as MPCK-Means, but eliminates the metric learning aspect and assumes an identity distance metric, setting $f_{ml}(x_i, x_j) = 1$ and $f_{cl}(x_i, x_j) = 1$.

2.2 Multi-view learning

Multi-view learning was originated by Blum et al. in the co-training algorithm [8] for semi-supervised classification. Co-training uses the model for each view to incrementally label the unlabeled data. Labels that are predicted with high confidence are transferred to the corresponding unlabeled instances in the other views to improve learning, and the process iterates until all instances are labeled. Co-training assumes independence between the views, and shows decreased performance when this assumption is violated [17].

Nigam and Ghani [17] propose the co-EM algorithm as an iterative multi-view form of expectation-maximization. At each iteration, co-EM estimates the model for a view and uses it to probabilistically label all of the data; these labels are then transferred to train another view during the next iteration. Co-EM repeats this process until the models for all views converge. Unlike co-training, co-EM does not require the views to be independent in order to perform well. The approach we explore in this paper uses a variant of co-EM to iteratively infer constraints in each view, and transfer those constraints to affect learning in the other views.

Clustering with multiple views has previous been explored by Bickel and Scheffer [6], who developed a multi-view EM algorithm that alternates between the views used to learn the model parameters and estimate the cluster assignments. Multi-view clustering has also been studied using canonical correlation analysis to construct low-dimensional embeddings from multiple views [9], spectral clustering that minimizes the disagreement between views [12], cross-modal clustering between perceptual channels [11], and information-theoretic frameworks [19, 14, 20].

3. PRELIMINARIES

Our multi-view constrained clustering approach (described in the next section) takes as input multiple views of the data $\mathcal{X} = \{X^A, X^B, \ldots\}$. Each view V of the data is given by a set of instances $X^V = \{x_1^V, x_2^V, \ldots, x_{n_V}^V\}$, with each $x_i^V \in \mathbb{R}^{d_V}$. The feature set and dimensionality d_V may differ between the views. We will initially focus on the case of two views, given by X^A and X^B, and extend our approach to handle an arbitrary number of views in Section 4.3.

Within $\mathcal{X}$, there are pairs of instances that correspond to different views of the same objects. We denote this connection between two instances x_u^A and x_v^B in different views by a relation $r_i = \langle x_u^A, x_v^B \rangle \in X^A \times X^B$. The set of relations $\mathcal{R}^{A\times B} = \{r_1, r_2, \ldots\} \subseteq X^A \times X^B$ defines a bipartite graph between X^A and X^B. Most other work on multi-view learning [17, 8, 6] assumes that $\mathcal{R}^{A\times B}$ defines a complete bipartite mapping between the two views. We broaden this assumption and consider the case where $\mathcal{R}^{A\times B}$ provides only a partial mapping between the views, and moreover when there are many more data instances than relations between views (i.e., $|\mathcal{R}^{A\times B}| << |X^A| + |X^B|$).

We also have a set of pairwise must-link and cannot-link constraints for each view V, given by $\mathcal{C}^V \subseteq X^V \times X^V \times \mathbb{R}_0^+ \times \{$*must-link*, *cannot-link*$\}$. Depending on the application, these constraints may either be manually specified by the user or extracted automatically from labeled data. Note that the constraints describe relationships between instances within a *single* view, while the mapping $\mathcal{R}^{A\times B}$ defines connections between instances in *different* views.

4. MULTI-VIEW CONSTRAINED CLUSTERING

Our multi-view constrained clustering approach takes as input multiple views of the data $\mathcal{X} = \{X^A, X^B, \ldots\}$, their associated sets of pairwise constraints $\mathcal{C}^A, \mathcal{C}^B, \ldots$, and a (potentially incomplete) mapping $\mathcal{R}^{U\times V}$ between each pair of different views U and V. Although we focus primarily on the case of two views A and B, we also generalize our approach to multiple views, as described in Section 4.3. The objective of our approach is to determine a k-partitioning of the data for each view that respects both the constraints within each view and the mapping between the views.

Our approach, given as Algorithm 1, iteratively clusters each view, infers new constraints within each view, and transfers those inferred constraints across views via the mapping. Through this process, progress in learning the model for one view will be rapidly transmitted to other views, making this approach particularly suited for problems where different aspects of the model are easy to learn in one view but difficult to learn in others.

The base constrained clustering algorithm is given by the *CKmeans* subfunction, which computes the clustering that maximizes the log-likelihood of the data X given the set of must- and cannot-link constraints $\mathcal{C}$. Our implementation uses either the PCK-Means or MPCK-Means algorithms as the *CKmeans* subfunction due to their native support for soft constraints and, for MPCK-Means, metric learning. However, our approach can utilize other constrained K-Means clustering algorithms, provided they meet the criteria for the *CKmeans* function listed above.

We fit the clustering model across both views using a variant of the co-EM algorithm [17]. In the E-step, we propagate the set of given constraints based on the current clustering models to those instances ($\hat{X}^A$ and $\hat{X}^B$) with direct mappings to the other views (Step 11, further described in Section 4.1). These propagated constraints can then be directly transferred to the other views via the mapping $\mathcal{R}^{A\times B}$ (Steps 4–9, further described in Section 4.2) to influence clustering during the M-step (Step 10). Note that instead of taking the direct union of all of the constraints, we keep only the maximally weighted constraint of each type (must-link and cannot-link) for every pair of instances; this operation is notated by the $\overset{\max}{\cup}$ operator in Step 9.

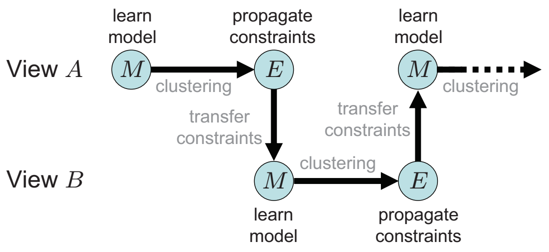


Figure 2: The relationship between the E-steps and M-steps in the different views.

Following previous work on co-EM and multi-view clustering [17, 6], we iterate the E-step in one view to propagate the constraints followed by the M-step in the other view to transfer those constraints and update the clustering. Each iteration of the co-EM loop (Steps 6–13) contains two iterations of both the E-step and the M-step, one for each view. The relationship between these steps is illustrated in Figure 2. The co-EM process continues until each view has internally converged. We assume convergence has occurred when the PCK-Means/MPCK-Means objective function's value differs by less than $\epsilon = 10^{-6}$ between successive iterations. Like Nigam and Ghani [17], we observed that our co-EM variant converged in very few iterations in practice. The iterative exchange of constraints between the views ensures a consistent clustering that respects both the constraints within and the mapping between views. The next two sections detail each step of the co-EM process.

4.1 E-step: Constraint propagation

In our model, the sets of pairwise constraints are the sole mechanisms for guiding the resulting clustering. We can directly map a constraint $\langle x_u, x_v \rangle$ between views only if the mapping is defined in $\mathcal{R}^{A\times B}$ for both endpoints x_u and x_v of the constraint. When $\mathcal{R}^{A\times B}$ is incomplete, the number of constraints with such a direct mapping for both endpoints is likely to be small. Consequently, we will be unable to directly map many of the constraints between views; each constraint that we cannot map represents lost information that may have improved the clustering.

Let $\hat{X}^V \subseteq X^V$ be the set of instances for view V that are mapped to another view. Given the initial constraints in $\mathcal{C}^V$, we infer new constraints between pairs of instances in $\hat{X}^V$ based on their local similarity to constraints in $\mathcal{C}^V$. We define this local similarity metric based on the current clustering model for view V, and propagate a constraint $\langle x_u^V, x_v^V \rangle \in \mathcal{C}^V$ to a pair of points $x_i^V, x_j^V \in \hat{X}^V$ if the pair is sufficiently similar to the original constraint. This process essentially considers these as spatial constraints [15] that affect not only the endpoints, but local neighborhoods of the instance space around those endpoints. Any effect on a pair of points in the neighborhood can be realized as a weighted constraint between those instances. Our constraint propagation method infers these constraints between instances in $\hat{X}^V$ with respect to the current clustering model. Since this set of new constraints (which we refer to as *propagated constraints*) is between instances with a direct mapping to other views, these constraints can be directly transferred to those other views via the mapping $\mathcal{R}^{A\times B}$. This approach can also be interpreted as inferring two weighted must-link constraints $\langle x_u^V, x_i^V \rangle$ and $\langle x_v^V, x_j^V \rangle$ and taking the transitive closure of them with $\langle x_u^V, x_v^V \rangle$ to obtain $\langle x_i^V, x_j^V \rangle$.

Algorithm 1 Multi-view Constrained Clustering with Constraint Propagation

```
Input: first view $X^A$ and constraints $\mathcal{C}^A$,
       second view $X^B$ and constraints $\mathcal{C}^B$,
       thresholds $t_A \in (0,1]$ and $t_B \in (0,1]$,
       the set of cross-view relations $\mathcal{R}^{A\times B}$, and
       the number of clusters $k$.
 1: Compute the transitive closure of $\mathcal{C}^A \bigcup \mathcal{C}^B \bigcup \mathcal{R}^{A\times B}$.
 2: Augment $\mathcal{C}^A$, $\mathcal{C}^B$, and $\mathcal{R}^{A\times B}$ with additional constraints
    from the transitive closure involving only instances from,
    respectively, $X^A \times X^A$, $X^B \times X^B$, and $X^A \times X^B$.
 3: Let $\hat{X}^A \subseteq X^A$ be the set of instances from $X^A$ involved
    in $\mathcal{R}^{A\times B}$; similarly define $\hat{X}^B \subseteq X^B$.
 4: Define constraint mapping functions $f_{A\mapsto B}$ and $f_{B\mapsto A}$
    across views via $\mathcal{R}^{A\times B}$.
 5: Initialize the sets of propagated constraints $\mathcal{P}^V = \emptyset$ for
    $V \in \{A, B\}$.
 6: repeat
 7:   for $V \in \{A, B\}$ do
 8:     Let $U$ denote the opposite view from $V$.
        // M-step
 9:     Define the unified set of constraints, mapped with
        respect to view $V$:
        $\tilde{\mathcal{C}}^V = \mathcal{C}^V \overset{\max}{\bigcup} f_{U\mapsto V}(\mathcal{P}^U)$
10:     Update the clustering using constrained K-Means:
        $(P^V, \mathcal{M}^V) = CKmeans(X^V, \tilde{\mathcal{C}}^V, k)$
        // E-step
11:     Estimate the set of propagated constraints:
        $\mathcal{P}^V = \Big\{ \langle x_i^V, x_j^V \rangle \;:\; x_i^V, x_j^V \in \hat{X}^V \;\wedge$
                    $\langle x_u^V, x_v^V \rangle \in \mathcal{C}^V \;\wedge$
                    $W\big(\langle x_i^V, x_j^V \rangle, \langle x_u^V, x_v^V \rangle\big) \geq t_V \Big\}$
12:   end for
13: until $P_A$ and $P_B$ have both internally converged
Output: the clustering $P_A$ and $P_B$.

Subfunction: $(P, \mathcal{M}) = CKmeans(X, \mathcal{C}, k)$
Function prototype for constrained K-Means.
Input: data $X$, must-link and cannot-link constraints $\mathcal{C}$,
       and the number of clusters $k$.
Output: the clustering $P$ and set of metrics for each
        cluster $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_k\}$.
```

The propagation process occurs with respect to the current clustering model for view V. Since we use K-Means variants as the base learning algorithm, the learned model is essentially equivalent to a Gaussian mixture model, under particular assumptions of uniform mixture priors and conditional distributions based on the set of constraints [7, 4]. Therefore, we can consider that each cluster h is generated by a Gaussian with a covariance matrix $\mathbf{\Sigma}_h$. For base clustering algorithms that support metric learning (e.g., MPCK-Means), the cluster covariance is related to the inverse of the cluster metric $\mathbf{M}_h$ learned as part of the clustering process. Bar-Hillel et al. [2] note that, in practice, metric learning typically constructs the metric modulo a scale factor α_h. Although this scale factor does not affect clustering, since only relative distances are required, constraint propagation requires absolute distances. Therefore, we must rescale the learned covariance matrix $\mathbf{M}_h^{-1}$ by α_h to match the data.

We compute α_h based on the empirical covariance $\tilde{\mathbf{\Sigma}}_h$ of the data $P_h \subset X^V$ assigned to cluster h, given by

$$\tilde{\mathbf{\Sigma}}_h = \frac{1}{|P_h|} \sum_{x \in P_h} (x - \mu_h)(x - \mu_h)^\mathsf{T} + \gamma \mathbf{I} \ , \tag{1}$$

adding a small amount of regularization $\gamma\mathbf{I}$ to ensure that $\tilde{\mathbf{\Sigma}}_h$ is non-singular for small data samples. Given $\mathbf{M}_h^{-1}$ and $\tilde{\mathbf{\Sigma}}_h$, we compute α_h as the scale such that the variances of the first principal component of each matrix are identical. We take the eigendecomposition of each matrix

$$\mathbf{M}_h^{-1} = \mathbf{Q}_{\mathbf{M}_h^{-1}} \mathbf{\Lambda}_{\mathbf{M}_h^{-1}} \mathbf{Q}_{\mathbf{M}_h^{-1}}^\mathsf{T} \qquad \tilde{\mathbf{\Sigma}}_h = \mathbf{Q}_{\tilde{\mathbf{\Sigma}}_h} \mathbf{\Lambda}_{\tilde{\mathbf{\Sigma}}_h} \mathbf{Q}_{\tilde{\mathbf{\Sigma}}_h}^\mathsf{T} \tag{2}$$

to yield diagonal matrices of eigenvalues in $\mathbf{\Lambda}_{\mathbf{M}_h^{-1}}$ and $\mathbf{\Lambda}_{\tilde{\mathbf{\Sigma}}_h}$. To derive the scale factor α_h, we ensure that both first principal components have equal variances, which occurs when

$$\alpha_h = \frac{\max\left(\mathbf{\Lambda}_{\tilde{\mathbf{\Sigma}}_h}\right)}{\max\left(\mathbf{\Lambda}_{\mathbf{M}_h^{-1}}\right)} \ , \tag{3}$$

yielding $\mathbf{\Sigma}_h = \alpha_h \mathbf{M}_h^{-1}$ as the covariance matrix for cluster h. When the base learning algorithm does not support metric learning, such as PCK-Means, we can instead use $\mathbf{\Sigma}_h = \tilde{\mathbf{\Sigma}}_h$ as cluster h's covariance matrix. The model for cluster h is then given by

$$G_h(x^V) = \exp\left(-\tfrac{1}{2}\left\| x^V - \mu_h^V \right\|^2_{\mathbf{\Sigma}_h{}^{-1}}\right) \ , \tag{4}$$

where $\left\| x^V - \mu_h^V \right\|^2_{\mathbf{\Sigma}_h{}^{-1}} = (x^V - \mu_h^V)^\mathsf{T} \mathbf{\Sigma}_h{}^{-1} (x^V - \mu_h^V)$ is the squared Mahalanobis distance between x^V and μ_h^V according to the cluster's rescaled metric $\mathbf{\Sigma}_h{}^{-1}$.

We assume that each constraint should be propagated with respect to the current clustering model, with the shape (i.e., covariance) of the propagation equivalent to the shape of the respective clusters (as given by their covariance matrices). Additionally, we assume that the propagation distance should be proportional to the constraint's location in the cluster. Intuitively, a constraint located near the center of a cluster can be propagated a far distance, up to the cluster's edges, since being located near the center of the cluster implies that the model has high confidence in the relationship depicted by the constraint. Similarly, a constraint located near the edges of a cluster should only be propagated a short distance, since the relative cluster membership of these points is less certain at the cluster's fringe.

We propagate a given constraint $\langle x_u^V, x_v^V \rangle \in \mathcal{C}^V$ to two other points $x_i^V, x_j^V \in X^V$ according to a Gaussian radial basis function (RBF) of the distance as $\langle x_i^V, x_j^V \rangle$ moves away from $\langle x_u^V, x_v^V \rangle$. Under this construction, the weight of the propagated constraint decreases according to the RBF centered in $2d_V$-dimensional space at the original constraint's endpoints $\left[x_u^V \ x_v^V\right] \in \mathbb{R}^{2d_V}$ with a covariance matrix $\mathbf{\Sigma}_{uv}^V$ based on the respective clusters' covariance matrices.

To form the propagation covariance matrices for each endpoint, we scale the covariance matrix associated with endpoint x_u^V by the weight assigned to that endpoint according to the clustering model (Equation 4). This ensures that the amount of propagation falls off with increasing distance from the centroid, in direct relation to the model's confidence in the cluster membership of x_u^V. The covariance matrix for the constraint propagation function is then given by

$$\mathbf{\Sigma}_{uv}^V = \begin{bmatrix} G_{c_u}(x_u^V)\, \mathbf{\Sigma}_{c_u} & \mathbf{0} \\ \mathbf{0} & G_{c_v}(x_v^V)\, \mathbf{\Sigma}_{c_v} \end{bmatrix} \ , \tag{5}$$

where c_u denotes the cluster of x_u and $\mathbf{0}$ denotes the $d_V \times d_V$ zero matrix. This construction assumes independence between x_u^V and x_v^V. While this assumption is likely to be violated in practice, we empirically show that it yields good results. For convenience, we represent the covariance matrices associated with each endpoint by $\mathbf{\Sigma}_{x_u^V} = G_{c_u}(x_u^V)\, \mathbf{\Sigma}_{c_u}^V$ for x_u^V and $\mathbf{\Sigma}_{x_v^V} = G_{c_v}(x_v^V)\, \mathbf{\Sigma}_{c_v}^V$ for x_v^V. Figure 3 illustrates the results of this process on an example cluster.

Given a constraint $\langle x_u^V, x_v^V, w, \mathit{type} \rangle \in \mathcal{C}^V$ and two candidate points $x_i^V \in X^V$ and $x_j^V \in X^V$, we can now estimate the weight of the propagated constraint $\langle x_i^V, x_j^V \rangle$ as

$$\begin{aligned} W\big(\langle x_i^V, x_j^V \rangle, \langle x_u^V, x_v^V \rangle\big) = w \ \times \\ \max\Big(W'\big(\langle x_i^V, x_j^V \rangle, \langle x_u^V, x_v^V \rangle\big), \\ W'\big(\langle x_j^V, x_i^V \rangle, \langle x_u^V, x_v^V \rangle\big)\Big) \end{aligned} \tag{6}$$

where

$$\begin{aligned} W'\big(\langle x_i^V, x_j^V \rangle, \langle x_u^V, x_v^V \rangle\big) = &\exp\left(-\tfrac{1}{2}\left\| x_i^V - x_u^V \right\|^2_{\mathbf{\Sigma}^{-1}_{x_u^V}}\right) \times \\ &\exp\left(-\tfrac{1}{2}\left\| x_j^V - x_v^V \right\|^2_{\mathbf{\Sigma}^{-1}_{x_v^V}}\right) . \end{aligned} \tag{7}$$

Since the ordering of the instances matters in the propagation, we compute both possible pairings of constraint endpoints (x_u^V and x_v^V) to target endpoints (x_i^V and x_j^V), taking the maximum value of the propagation in Equation 6 to determine the best match. Under this propagation scheme, a constraint propagated to its own endpoints is given a weight of w (since the second term of the RHS of Equation 6 will be 1); the weight of the propagated constraint decreases as the endpoints x_i^V and x_j^V move farther from x_u^V and x_v^V. Section 4.4 describes mechanisms for implementing constraint propagation efficiently, taking advantage of the independence assumption between the two endpoints of a constraint and memoization of repeated computations.

The E-step of Algorithm 1 (Step 11) uses Equation 6 to propagate all given constraints within each view to those instances $\hat{X}^V$ with cross-view mappings, thereby inferring the expected value of constraints between those instances given the current clustering. Using this expected set of constraints, we can then update the current clustering model in the M-step as described in the next section.

4.2 M-step: Updating the clustering model

Given the expected constraints between instances in $\hat{X}^V$, we transfer those constraints to the other views and then update the clustering model to reflect these new constraints. These steps together constitute the M-step of Algorithm 1.

Any propagated constraint where both endpoints are in $\hat{X}^V$ can be transferred directly to another view U via the bipartite mapping $\mathcal{R}^{V \times U}$. We define a mapping function $f_{V \mapsto U} : X^V \times X^V \times \mathbb{R}_0^+ \times \{\mathit{must\text{-}link}, \mathit{cannot\text{-}link}\} \mapsto X^U \times X^U \times \mathbb{R}_0^+ \times \{\mathit{must\text{-}link}, \mathit{cannot\text{-}link}\}$ that takes a given constraint $c = \langle x_i^V, x_j^V, w, \mathit{type} \rangle \in \mathcal{C}^V$ and maps it to constrain

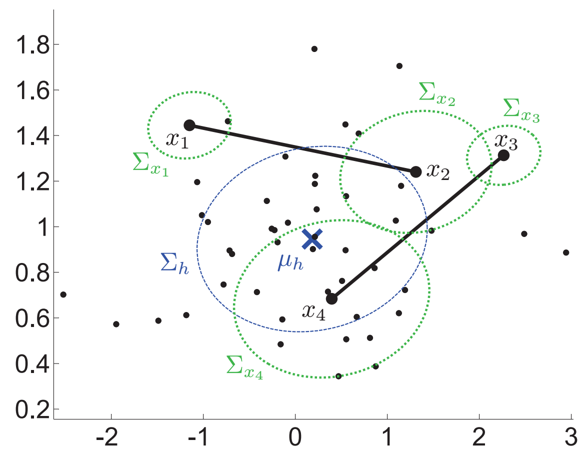


Figure 3: Constraint propagation applied to a single example cluster h, showing the learned covariance matrix $\mathbf{\Sigma}_h$ (dashed blue ellipse) rescaled to fit the data, two constraints (solid black lines) and the weighting functions centered at each endpoint (dotted green ellipses), which decrease in variance as they move farther from the centroid μ_h.

instances in X^U by:

$$f_{V\mapsto U}(c) = \left\{\langle x_u^U, x_v^U, w, \mathit{type}\rangle : \langle x_i^V, x_u^U\rangle \in \mathcal{R}^{V\times U} \wedge \langle x_j^V, x_v^U\rangle \in \mathcal{R}^{V\times U}\right\} . \quad (8)$$

Using this construction, we can define the mapping functions $f_{B\mapsto A}$ and $f_{A\mapsto B}$ in Algorithm 1. We then use these functions $f_{A\mapsto B}$ and $f_{B\mapsto A}$ to map propagated constraints between views in Step 9, transferring constraints inferred in one view to the other related views. These transferred constraints (from view U) can then be combined with the original constraints in each view V to inform the clustering. Instead of taking the direct union of these constraints, we keep only the maximally weighted constraint between each pair of instances to form the set

$$\tilde{\mathcal{C}}^V = \mathcal{C}^V \overset{\max}{\bigcup} f_{U\mapsto V}(\mathcal{P}^U) \ , \quad (9)$$

since each inferred constraint represents an estimate of the minimal strength of the pairwise relationship.

The optimal clustering models for view V can then be computed by clustering the data in each view subject to the constraints in $\tilde{\mathcal{C}}^V$ (Step 10). The *CKmeans* subfunction computes the clustering that maximizes the log-likelihood of the data subject to the set of constraints, thereby completing the M-step of Algorithm 1.

4.3 Extension to multiple views

Algorithm 1 can be easily extended to support more than two views. Each view X^V independently maintains its own sets of given constraints $\mathcal{C}^V$, threshold t_V, data $\hat{X}^V \subseteq X^V$ involved in any cross-view relations, current partitioning P^V, current cluster metrics $\mathcal{M}^V$, and propagated constraints $\mathcal{P}^V$. To handle more than two views, we maintain separate mappings $\mathcal{R}^{U\times V}$ for each pair of views X^U and X^V and use each mapping to define pairwise mapping functions $f_{U\mapsto V}$ and $f_{V\mapsto U}$ between views. For D views, $X^{(1)}, \ldots, X^{(D)}$, this approach will yield $D^2 - D$ mapping functions.

To generalize our approach to more than two views, we hold each set of propagated constraints fixed, and iteratively update the clustering (M-step), then recompute the set of propagated constraints (E-step) for one view. The unified sets of constraints for each view V becomes (Step 9)

$$\mathcal{C}^V = \mathcal{C}^V \overset{\max D}{\bigcup_{U=1}} f_{U\mapsto V}(\mathcal{P}^U) \quad (10)$$

under the convention that $f_{U\mapsto U}(\mathcal{P}^U) = \emptyset$. Each iteration of co-EM loops over the E-steps and M-steps for all views, and proceeds until the clustering for each view converges.

4.4 Implementation efficiency

The overall computational complexity of Algorithm 1 is limited by the maximum number of EM iterations and the complexity of the *CKMeans* function, which depends on the chosen clustering algorithm. Besides these aspects, the constraint propagation step (Step 11) incurs the greatest computational cost. To make this step computationally efficient, our implementation relies on the independence assumption inherent in Equation 7 between the two endpoints of the constraint. To efficiently compute the weight of all propagated constraints, we memoize the value of each endpoint's propagation $G(x_i^V, x_u^V) = \exp\left(-\frac{1}{2}(x_i^V - x_u^V)^\intercal \mathbf{\Sigma}_{x_u^V}^{-1}(x_i^V - x_u^V)\right)$ for $x_i^V \in \hat{X}^V$ and $x_u^V \in \bar{X}^V$, where $\bar{X}^V$ is the set of points involved in $\mathcal{C}^V$. Through memoization, we reduce the constraint propagation step to $|\hat{X}^V| \times |\bar{X}^V|$ Gaussian evaluations. Memoization applies similarly to all other views. Each constraint propagation is inherently independent from the others, making this approach suitable for parallel implementation using Hadoop/MapReduce [13].

When the covariance matrix $\mathbf{\Sigma}_{x_u^V}$ is diagonal, we can further reduce the computational cost through early stopping of the Gaussian evaluation once we are certain that the endpoint's propagation weight will be below the given threshold t_V. When $\mathbf{\Sigma}_{x_u^V}$ is diagonal, given by $\mathbf{\Sigma}_{x_u^V} = \mathit{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_{d_V}^2)$,

$$G(x_i^V, x_u^V) = \exp\left(-\frac{1}{2}\sum_{k=1}^{d_V}\frac{(x_{i,k}^V - x_{u,k}^V)^2}{\sigma_k^2}\right) \ . \quad (11)$$

Since a constraint is only propagated when the weight exceeds $t_V > 0$ and the maximum propagation for each Gaussian weight $G(x_i^V, x_u^V) \in [0,1]$, we only need to evaluate $W'\big(\langle x_i^V, x_j^V\rangle, \langle x_u^V, x_v^V\rangle\big)$ when both $G(x_i^V, x_u^V) \geq t_V$ and $G(x_j^V, x_v^V) \geq t_V$. Therefore we must ensure that

$$t_V \leq \exp\left(-\frac{1}{2}\sum_{k=1}^{d_V}\frac{(x_{i,k}^V - x_{u,k}^V)^2}{\sigma_k^2}\right) \quad (12)$$

$$-2\ln t_V \geq \sum_{k=1}^{d_V}\frac{(x_{i,k}^V - x_{u,k}^V)^2}{\sigma_k^2} . \quad (13)$$

Since all terms in the RHS summation are positive, we can compute them incrementally and stop early once the sum exceeds $-2\ln t_V$, since we will never need to evaluate any propagation weight $W'(\cdot)$ involving $G(x_i^V, x_u^V)$. In our implementation, we set $G(x_i^V, x_u^V) = 0$ in any cases where we can guarantee that $G(x_i^V, x_u^V) < t_V$.

5. EVALUATION

We evaluated multi-view constrained clustering on a variety of data sets, both synthetic and real, showing that our approach improves multi-view learning under an incomplete mapping as compared to several other methods.

5.1 Data sets

In order to examine the performance of our approach under various data distributions, we use a combination of synthetic and real data in our experiments. We follow the methodology of Nigam and Ghani [17] to create these multi-view data sets by pairing classes together to create "super-instances" consisting of one instance from each class in the pair. The two original instances then represent two different views of the super-instance, and their connection forms a mapping between the views. This methodology can be trivially extended to an arbitrary number of views. These data sets are described below and summarized in Table 1.

Four quadrants is a synthetic data set composed of 200 instances drawn from four Gaussians in $\mathbb{R}^2$ space with identity covariance. The Gaussians are centered at the coordinates $(\pm 3, \pm 3)$, one in each of the four quadrants. Quadrants I and IV belong to the same cluster and quadrants II and III belong to the same cluster. The challenge in this simple data set is to identify these clusters automatically, which requires the use of constraints to improve performance beyond random chance. To form the two views, we drew 50 instances from each of the four Gaussians, divided them evenly between views, and created mappings between nearest neighbors that were in the same quadrant but different views.

Protein includes 116 instances divided among six classes of proteins $\{c_1, c_2, \ldots, c_6\}$. This data set was previously used by Xing et al. [24]. To create multiple views of this data set, we partition it into two views containing respectively instances from classes $\{c_1, c_2, c_3\}$ and $\{c_4, c_5, c_6\}$. We connected instances between the following pairs of classes to create the two views: c_1 & c_4, c_2 & c_5, and c_3 & c_6. Through this construction, a model learned for clustering $\{c_1, c_2, c_3\}$ in one view can be used to inform the clustering of $\{c_4, c_5, c_6\}$ in the other view. Since the clusters do not contain the same numbers of instances, some instances within each view are isolated in the mapping.

Letters/Digits uses the letters-IJL and digits-389 data sets previously used by Bilenko et al. [7]. These are subsets of the letters and digits data sets from the UCI machine learning repository [1] containing only the letters $\{I, J, L\}$ and the digits $\{3, 8, 9\}$, respectively. We map instances between views according to the following pairings: I & 3, J & 8, and L & 9, leaving those instances without a correspondence in the other view isolated in the mapping.

Rec/Talk is a subset of the 20 Newsgroups data set [18], containing 5% of the instances from the newsgroups {*rec.autos*, *rec.motorcycles*} in the *rec* view, and 5% of the newsgroups {*talk.politics.guns*, *talk.politics.mideast*} in the *talk* view. We process each view independently, removing stop words and representing the data as a binary vector of the 50 most discriminatory words as determined by Weka's string-to-wordvector filter [23]. As in the previous data sets, we form the mapping between views by pairing clusters in order.

We create a low-dimensional embedding of each data set using the spectral features [16] in order to improve clustering, with the exception of Four Quadrants, for which we use the original features because the dimensionality is already low. For each view V, we compute the pairwise affinity matrix $\mathbf{A}$ between the instances x_i and x_j using a radial basis function of their distance, given by $A_{i,j} = \exp(-||x_i - x_j||^2 / 2\sigma^2)$. We use $\sigma = 1$ as the rate at which the affinity falls off with increasing distance. From $\mathbf{A}$, we form the normalized Laplacian matrix [10] for the data set, given by $\boldsymbol{\mathcal{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal degree matrix $D_{i,i} = \sum_{j=1}^{d_V} A_{i,j}$ and $\mathbf{I}$ is the identity matrix. The eigendecomposition of the normalized Laplacian matrix $\boldsymbol{\mathcal{L}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathsf{T}}$ yields the spectral features for the data set in the columns of the eigenvector matrix $\mathbf{Q}$. We keep the 2^{nd} through $d + 1^{th}$ eigenvectors (corresponding to the 2^{nd} through $d + 1^{th}$ lowest eigenvalues in $\boldsymbol{\Lambda}$) as the features for clustering; we discard the first eigenvector since it is constant and therefore does not discriminate between the instances. In this paper, we use $d = \lceil \sqrt{d_V} \rceil$ for Protein and Letters/Digits, and $d = 5$ for the Rec/Talk data set. Additionally, we standardize all features to have zero mean and unit variance. These spectral features are computed independently between the different views, further emphasizing that the mapping is the only connection between views.

Name	**#Insts**	**#Dims**	**k**	$\mathbf{t_V}$
Four Quadrants	200/200	2	2	0.75
Protein	67/49	20	3	0.50
Letters/Digits	227/317	16	3	0.95
Rec/Talk	100/94	50	2	0.75

Table 1: Data set and experiment parameters

5.2 Methodology

Within each view, we use the cluster labels on the instances to sample a set of pairwise constraints, ensuring equal proportions of must-link and cannot-link constraints. The weight of all constraints w is set to 1. We also sample a portion of the mapping to use for transferring constraints between views. Both the sets of constraints and the mapping between views are resampled each trial of our experiments.

We compare **Constraint Propagation** against several other potential methods for transferring constraints:

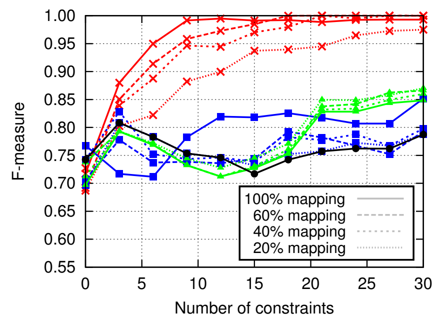
Direct Mapping transfers only those constraints that already exist between instances in $\hat{X}^V$. This approach is equivalent to other methods for multi-view learning that are only capable of transferring labeled information if there is a direct mapping between views.

Cluster Membership can be used to infer constraints between instances in $\hat{X}^V$. This approach simply considers the relative cluster membership for each pair of instances in $\hat{X}^V$ and infers the appropriate type of constraint with a weight of 1.

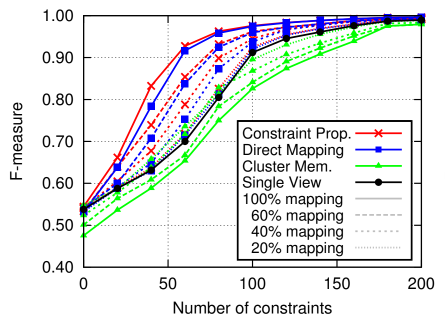
Single View performs constrained clustering on each of the individual views in isolation and serves as a lower baseline for the experiments.

For the base constrained clustering algorithm, we use the PCK-Means and MPCK-Means implementations provided in the WekaUT extension[1] to the Weka toolkit [23] with their default values. For PCK-Means, Constraint Propagation uses the full empirical covariance for each cluster; for MPCK-Means, it uses the diagonal weighted Euclidean metrics learned on a per-cluster basis by MPCK-Means.

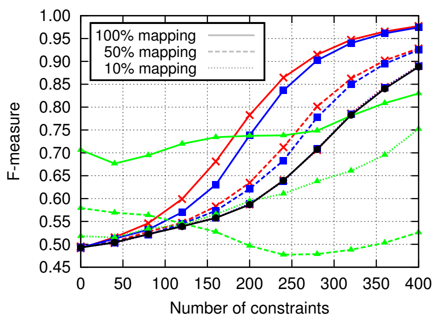
[1]`http://www.cs.utexas.edu/users/ml/risc/code/`



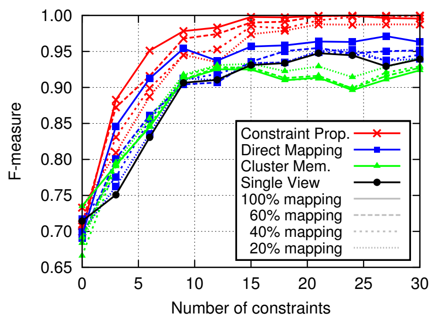
(a) Four Quadrants - PCK-Means



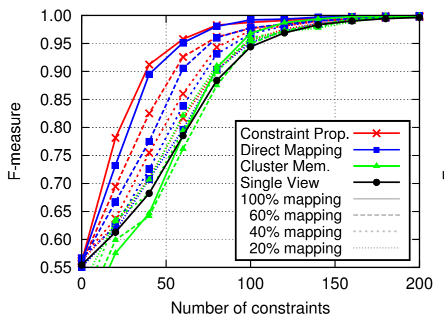
(b) Protein - PCK-Means



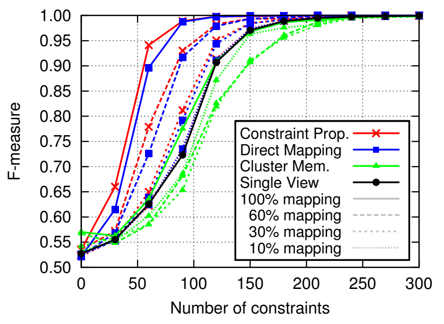
(c) Letters/Digits - PCK-Means



(d) Four Quadrants - MPCK-Means



(e) Protein - MPCK-Means



(f) Rec/Talk - MPCK-Means

Figure 4: Comparison of multi-view constrained clustering performance. The percentage of instances mapped between views (e.g., 20%, 40%, 100%) is denoted by the type of the line (dotted, dashed, solid), and the constraint transfer method is denoted by the color and marker shape. In several plots, we truncated the key due to space limitations; those plots use the same markers and colors to depict the constraint transfer methods as the other plots. (Best viewed in color.)

We measure performance using the pairwise F-measure – a version of the information-theoretic F-measure adapted to measure the number of same-cluster pairs for clustering [3]. The pairwise F-measure is the harmonic mean of precision and recall, given by

$$F\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{14}$$

where

$$precision = \frac{num\text{-}pairs\text{-}correctly\text{-}predicted\text{-}in\text{-}same\text{-}cluster}{num\text{-}total\text{-}pairs\text{-}predicted\text{-}in\text{-}same\text{-}cluster} \ ,$$

$$recall = \frac{num\text{-}pairs\text{-}correctly\text{-}predicted\text{-}in\text{-}same\text{-}cluster}{num\text{-}total\text{-}pairs\text{-}in\text{-}same\text{-}cluster} \ .$$

We take the mean of the performance for all views, yielding a single performance measure for each experiment.

In each trial, we consider performance as we vary the number of constraints used for learning and the percentage of instances in each view that are mapped to the other views. Our results are shown in Figure 4, averaged over 100 trials.

5.3 Discussion

As shown in Figure 4, Constraint Propagation clearly performs better than the baseline of Single View clustering, and better than Cluster Membership for inferring constraints in all cases, except for when learning with few constraints on Letters/Digits. Constraint Propagation also yields an improvement over transfer using the Direct Mapping method for each percentage of instances mapped between views, as shown in Figure 5. We omit the Four Quadrants (PCK-Means) results from Figure 5 due to the relatively high per-

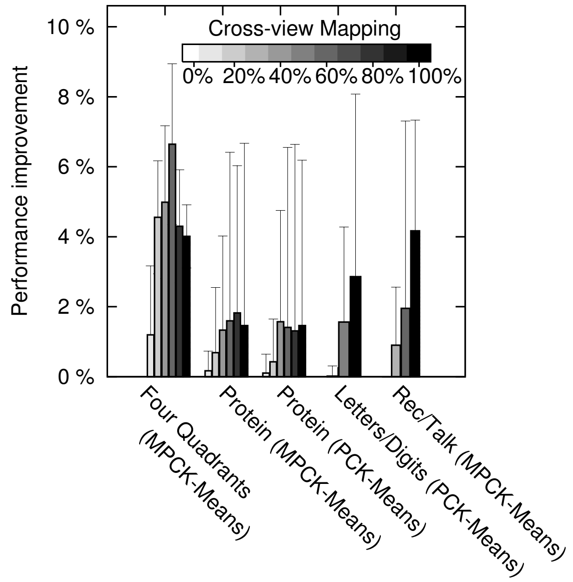


Figure 5: The performance improvement of constraint propagation over direct mapping in Figure 4, averaged over the learning curve. The peak whiskers depict the maximum percentage improvement.

formance gain of Constraint Propagation, which averages a 21.3% improvement over Direct Mapping with peak gains above 30%. Unlike Direct Mapping, Constraint Propagation

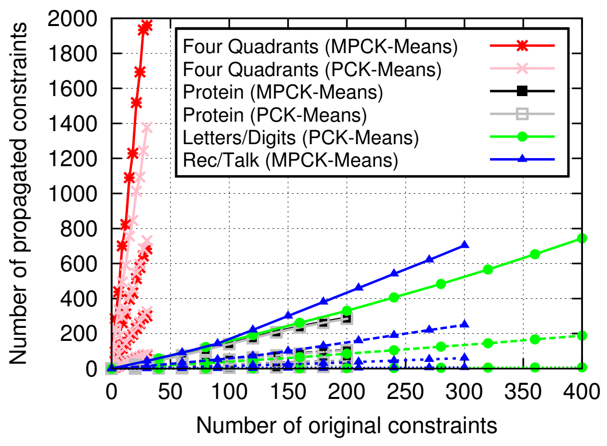


Figure 6: The number of propagated constraints as a function of the number of original constraints, averaged over all co-EM iterations. The line type (solid, dashed, dotted) depicts the mapping percentage, as defined in the corresponding plot in Figure 4.

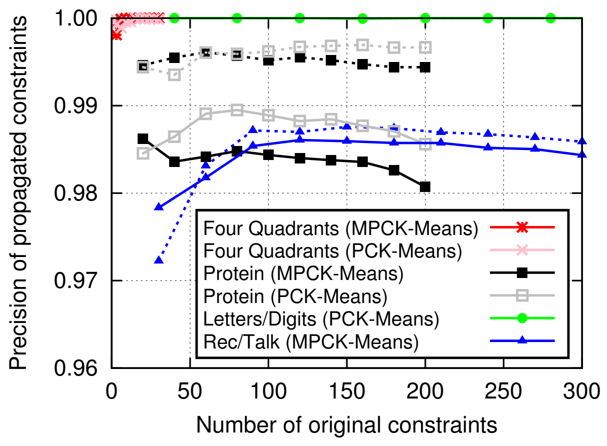


Figure 7: The precision of the propagated must-link (solid line) and cannot-link (dashed line) constraints, as measured against the true class labels.

is able to transfer those constraints that would otherwise be discarded, increasing the performance of multi-view clustering. The performance of both Constraint Propagation and Direct Mapping improve as the mapping becomes more complete between the views, with Constraint Propagation still retaining an advantage over Direct Mapping even with a complete mapping, as shown in all data sets. We hypothesize that in the case of a complete mapping, Constraint Propagation behaves similarly to spatial constraints [15], warping the underlying space with the inference of new constraints that improve performance.

On these data sets, the number of constraints inferred by Constraint Propagation is approximately linear in the number of original constraints, as shown in Figure 6. Clearly, as the mapping between views becomes more complete, Constraint Propagation is able to infer a larger number of constraints between those instances in $\hat{X}^V$.

The improvement in clustering performance is due to the high precision of the propagated constraints. Figure 7 shows the average weighted precision of the propagated constraints for the 100% mapping case, measured against the complete set of pairwise constraints that can be inferred from the true class labels. The proportion that each propagated constraint contributed to the weighted precision is given by the constraint's inferred weight w. We also measured the precision of propagated constraints for various partial mappings, and the results were comparable to those for the complete mapping. For each data set, the constraint inferred through propagation show a high average precision of 98–100%, signifying that very few incorrect constraints are inferred by the propagation method.

Interestingly, the constraint propagation method works slightly better for cannot-link constraints than must-link constraints. This phenomenon can be explained by a counting argument that there are many more chances for a cannot-link constraint to be correctly propagated than a must-link constraint. For example, with k clusters where each cluster contains n/k instances, each given must-link constraint can be correctly propagated to $numMLprop = \binom{n/k}{2} - 1$ other pairs of instances in the same cluster. However, each given cannot-link constraint can be correctly propagated to $numCLprop = \binom{n}{2} - k\binom{n/k}{2} - 1$ other pairs of instances that belong in different clusters,[2] which is *much* greater than $numMLprop$ (e.g., for $n = 1,000$ and $k = 10$, $numCLprop = 449,999 \gg 4,949 = numMLprop$). Therefore, a must-link constraint has much less chance of being propagated correctly than a cannot-link constraint.

We found that for high-dimensional data, the curse of dimensionality causes instances to be so far separated that Constraint Propagation is only able to infer constraints with a very low weight. Consequently, it works best with a low-dimensional embedding of the data, motivating our use of spectral feature reduction. Other approaches could also be used for creating the low-dimensional embedding, such as principal components analysis or manifold learning.

Additionally, like other multi-view algorithms, we found Constraint Propagation to be somewhat sensitive to the cut-off thresholds t_V, but this problem can be remedied by using cross-validation to choose t_V. Too high a threshold yields performance identical to Direct Mapping (since no constraints would be inferred), while too low a threshold yields the same decreased performance as exhibited by other co-training algorithms. For this reason, we recommend setting t_V to optimize performance as evaluated by cross-validation over the set of constrained instances.

We ran several additional experiments on data sets with poor mappings and distributions that violated the mixture-of-Gaussians assumption of K-Means clustering; we omit these results due to space limitations. On these data sets, Constraint Propagation decreased performance in some cases, due to inferring constraints that were not justified by the data. This would occur, for example, in clusters with a nested half-moon shape, where Constraint Propagation would incorrectly infer constraints between instances in the opposing cluster. In these cases, clustering using only the directly mapped constraints yielded the best performance.

[2]The number of propagated cannot-link constraints is derived by taking the number of possible different constraints $\binom{n}{2} - 1$, and subtracting off the total number of possible must-link constraints $k\binom{n/k}{2}$.

6. CONCLUSION

Constraint propagation has the ability to improve multi-view constrained clustering when the mapping between views is incomplete. Besides improving performance, constraint propagation also provides the ability for the user to interact with one view to supply constraints, and for those constraints to be transferred to improve learning in the other views. This is especially beneficial when the interaction is more natural for the user in one view (e.g., images) than the other views (e.g., text or audio) that may require lengthy examination of each instance in order to infer the constraints.

In future work, we will consider other methods for learning with a partial mapping between views, such as label inference, manifold alignment, and transfer learning. This work will improve the ability to use isolated instances that do not have a corresponding multi-view representation to improve learning, and enable multi-view learning methods to be used for a wider variety of applications

7. ACKNOWLEDGMENTS

We would like to thank Kiri Wagstaff, Katherine Guo, Tim Oates, and Tim Finin for their feedback. This work is partly based on the first author's Master's thesis at UMBC. This research was supported in part by a graduate fellowship from the Goddard Earth Sciences and Technology Center at UMBC, NSF ITR grant #0325329, and Lockheed Martin.

8. REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository. Available online at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

[3] S. Basu. *Semi-Supervised Clustering: Probabilistic Models, Algorithms, and Experiments*. PhD thesis, University of Texas at Austin, August 2005.

[4] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the International Conference on Machine Learning*, pages 19–26, 2002. Morgan Kauffman.

[5] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 333–344, 2004. SIAM.

[6] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, pages 19–26, 2004. IEEE.

[7] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the International Conference on Machine Learning*, pages 81–88, 2004. ACM.

[8] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Computational Learning Theory*, pages 92–100, 1998. Morgan Kaufmann.

[9] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the International Conference on Machine Learning*, pages 129–136, 2009. ACM.

[10] F. R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1994.

[11] M. H. Coen. Cross-modal clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 932–937, 2005. AAAI Press.

[12] V. R. de Sa. Spectral clustering with two views. In *Working Notes of the ICML'05 Workshop on Learning with Multiple Views*, 2005.

[13] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[14] Y. Gao, S. Gu, J. Li, and Z. Liao. The multi-view information bottleneck clustering. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, volume 4443 of *Lecture Notes in Computer Science*, pages 912–917. Springer, April 2007.

[15] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints. In *Proceedings of the International Conference on Machine Learning*, pages 307–314, 2002. Morgan Kauffman.

[16] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems* 14:849–856, 2001.

[17] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'00)*, pages 86–93, 2000. ACM.

[18] J. Rennie. 20 Newsgroups data set, sorted by date. Available online at http://www.ai.mit.edu/~jrennie/20Newsgroups/, September 2003.

[19] K. Sridharan and S. M. Kakade. An information theoretic framework for multi-view learning. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT'08)*, pages 403–414. Omnipress, July 2008.

[20] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1016–1021. IEEE, 2009.

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, pages 577–584, 2001. Morgan Kaufmann.

[22] K. L. Wagstaff. *Intelligent Clustering with Instance-Level Constraints*. PhD thesis, Cornell University, May 2002.

[23] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[24] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 15:505–512, 2003.