Online Multi-view Clustering with Incomplete Views

Weixiang Shao*, Lifang He†, Chun-ta Lu* and Philip S. Yu*

*University of Illinois at Chicago, Chicago, IL
†Shenzhen University, Guangdong, China
Email: {wshao4, clu29, psyu}@uic.edu, lifanghescut@gmail.com

Abstract—In this paper, we propose an online multi-view clustering algorithm, OMVC, which deals with large-scale incomplete views. We model the multi-view clustering problem as a joint weighted NMF problem and process the multi-view data chunk by chunk to reduce the memory requirement. OMVC learns the latent feature matrices for all the views and pushes them towards a consensus. We further increase the robustness of the learned latent feature matrices in OMVC via lasso regularization. To minimize the influence of incompleteness, dynamic weight setting is introduced to give lower weights to the incoming missing instances in different views. More importantly, to reduce the computational time, we incorporate a faster projected gradient descent by utilizing the Hessian matrices in OMVC. Extensive experiments conducted on four real data demonstrate the effectiveness of OMVC.

Keywords-Multi-view clustering; Online algorithm; Incomplete views; Nonnegative matrix factorization

I. INTRODUCTION

With the advance of technology, real data are often with multiple modalities or coming from multiple sources. Such data is called multi-view data. Usually, multiple views provide consistent and complementary information for the semantically same data. By exploiting these characteristics between multi-view data, multi-view learning can obtain better performance than relying on just one single view [19]. Multi-view clustering [4], as one of the basic tasks of multi-view learning, provides a natural way for generating clusters from multi-view data and has attracted considerable attention. Many approaches have been proposed in the recent decade [5, 6, 8, 9, 13].

Most of the above studies are based on the assumption that all of the views are complete, i.e., each instance appears in all views. However, due to the nature of the data or the cost of data collection, some views may suffer from the incompleteness of data (i.e., instances within some views missing). To address this issue, different approaches have been explored [10, 16, 17, 20]. [20] is the first to deal with incomplete views by using information from one complete view to refer to the kernel of incomplete views. [10, 17] are among the first attempts to solve multi-view clustering with none of the views complete. [10, 16] are the first attempts to solve multiple incomplete views clustering based on nonnegative matrix factorization (NMF). All the previous works require that multi-view data can fit into memory.

However, in reality, the size of multi-view data may be extremely huge. For example, in Web scale data mining, one may encounter billions of Web pages and the dimension of the features may be as large as $O(10^6)$. Such large-scale data clearly cannot fit into the memory of a single machine. Even a small corpus like Wikipedia has more than $3 \times 10^7$ pages in multiple languages. None of the existing multi-view clustering algorithms can handle data in such scale.

There are several challenges preventing us from applying multi-view clustering algorithms to large-scale data. 1) With the memory limitation, how to combine various views in different feature spaces and explore the consistency and complementary properties of different views to get better clustering solutions. 2) When the data are too large to fit into memory, how to deal with incomplete views, i.e., how to minimize the influence of incompleteness of views. 3) How to effectively and efficiently learn the clustering solution even if the multi-view data are extremely large.

In this paper, we propose OMVC (Online Multi-View Clustering) to solve the above three challenges. To the best of our knowledge, this is the first online approach to solve the large-scale multi-view clustering problem with incomplete views. Our contributions can be summarized as follows: 1) OMVC is the first attempt to solve the problem of large-scale multi-view clustering with incomplete views in an online fashion. 2) We model multi-view clustering as a joint NMF problem, which captures the relation between different heterogeneous views and learns a consensus latent feature matrix across all the views. 3) We introduce dynamic weights for missing data in different views to reduce the influence of incomplete views. 4) By utilizing lasso regularization, OMVC enforces the sparsity of latent feature matrices, and increase its robustness. 5) By doing multi-view clustering in an online fashion and adopting a faster projected gradient descent technique, OMVC can scale up to large data without appreciable sacrifice of performance.

The rest of the paper is organized as follows. We start by giving the problem description and introducing the background of NMF-based clustering in Section II. Then we present our proposed OMVC approach in Sections III and IV. Section V reports the experimental results and followed by conclusion in Section VI.

Corresponding author: Lifang He
II. PRELIMINARIES

A. Problem Description

Assume that we are given \( N \) instances in \( n_v \) incomplete views \( \{X^{(v)}, v = 1, 2, ..., n_v\} \), where \( X^{(v)} \in \mathbb{R}^{D_v \times N} \) represents the data in the \( v \)-th view and \( D_v \) is the dimensionality of features in the \( v \)-th view. Each view may be incomplete, i.e., each of the view may have instances missing. We define an instance-view indicator matrix \( M \in \mathbb{R}^{N \times n_v} \) by

\[
m_{i,j} = \begin{cases} 
1 & \text{if the } i \text{-th instance is in the } j \text{-th view.} \\
0 & \text{otherwise.}
\end{cases}
\]

(1)

where each column of \( M \) represents the instance presence in one view. Thus, in the incomplete views scenario, \( \sum_{i=1}^N m_{i,j} < N \) for \( j = 1, 2, ..., n_v \). Our goal is to partition all the \( N \) instances into \( K \) clusters by integrating all the \( n_v \) incomplete views in an online fashion.

B. NMF-based Clustering

Let \( X \in \mathbb{R}^{D \times N} \) denote the nonnegative data matrix where each column represents an instance and each row represents a feature. Nonnegative matrix factorization (NMF) aims to factorize the data matrix \( X \) into two nonnegative matrices. We denote the two nonnegative matrices as \( U \in \mathbb{R}^{D \times R} \) and \( V \in \mathbb{R}^{R \times N} \). Here \( R \) is the desired reduced dimensionality.

To facilitate discussions, we call \( U \) the basis matrix and \( V \) the latent feature matrix. The objective function of NMF can be formulated as below:

\[
\min_{U,V} \mathcal{L} = \|X - UV^T\|_F^2 \quad \text{s.t. } U \geq 0, V \geq 0,
\]

(2)

where \( \| \cdot \|_F \) is the Frobenius norm. In clustering problems, the latent feature matrix \( V \) is used to extract the clustering solution. One option is to apply standard \( K \)-means on \( V \) to get the clustering solution. Another option is to add constraints to further restrict the rows of \( V \) and get the clustering indicators directly from \( V \). For example, we can constrain \( \sum_j v_{i,j} = 1 \) for every row \( i \). Thus, \( v_{i,j} \) becomes the probability that instance \( i \) belongs to cluster \( j \).

In general, it is difficult to solve Eq. (2) as the objective function is not convex with \( U \) and \( V \) jointly. A common solution is to use an alternating way to update \( U \) and \( V \) [2]. One of the most well-known algorithms that solves this problem is Projected Gradient Descent (PGD) [11]. By fixing \( V \), PGD updates \( U \) using the first-order gradient:

\[
U \leftarrow P[U - \gamma_k \nabla_U \mathcal{L}(U, V)]
\]

(3)

where \( \nabla_U \mathcal{L}(U, V) \) is the gradient of \( \mathcal{L} \) in Eq. (2) with respect to \( U \), \( \gamma_k \) is the step size and \( P \) is defined as

\[
P[u_{i,j}] = \begin{cases} 
 u_{i,j}, & \text{if } u_{i,j} \geq 0 \\
 0, & \text{otherwise.}
\end{cases}
\]

Similarly, PGD can be applied to update \( V \) with \( U \) fixed. PGD iteratively updates \( U \) and \( V \) until convergence. Although PGD is proved to converge, it only uses the first-order information, and thus the convergence rate is slow. To further accelerate the solving process, we borrow the idea from Newton’s method [3] by utilizing the second-order information (i.e., Hessian matrix) in this paper. Thus, the update equation for \( U \) in Eq. (3) becomes:

\[
U \leftarrow P[U - \gamma_k \mathcal{H}^{-1} \{ \nabla_U \mathcal{L}(U, V) \} \nabla_U \mathcal{L}(U, V)]
\]

(4)

where \( \mathcal{H}^{-1} \{ \nabla_U \mathcal{L}(U, V) \} \) is the inverse of the Hessian matrix. Similarly, we can apply the second-order PGD to update \( V \).

III. ONLINE MULTI-VIEW CLUSTERING

The proposed online multi-view clustering algorithm processes the data in a streaming fashion with low computational and storage complexity. We will first describe how to derive the objective function.

A. Objective of OMVC

Given a set of incomplete multi-view data \( \{X^{(v)} \in \mathbb{R}^{D_v \times N}, v = 1, 2, ..., n_v\} \), we aim to find the latent feature matrices for each of the view and a common consensus, which represents the integrated information of all the views. The objective function can be written as below:

\[
\min_{\{U^{(v)}, v \}} \mathcal{L} = \sum_{v=1}^{n_v} \|X^{(v)} - U^{(v)}V^{(v)}^T\|_F^2 + \sum_{v=1}^{n_v} \alpha_v \|V^{(v)} - V^*\|_F^2
\]

s.t. \( V^* \geq 0, U^{(v)} \geq 0, v = 1, 2, .., n_v \).

(5)

where \( U^{(v)} \in \mathbb{R}^{D_v \times K} \) and \( V^{(v)} \in \mathbb{R}^{N \times K} \) are the basis matrix and latent feature matrix for the \( v \)-th view, \( V^* \in \mathbb{R}^{N \times K} \) is the consensus latent feature matrix across all the views, \( K \) is the number of clusters and \( \alpha_v \) is the trade-off parameter for balancing the reconstruction error and the disagreement of the \( v \)-th view.

Due to the incompleteness of each view, we cannot directly optimize the above objective function. One simple solution is to fill the missing instances with average value of the features first, and then solve the above objective function. However, this approach depends on the quality of the filled instances. For small incomplete percentages, the quality of the information contained in the filled features may be sufficient, whereas when the number of missing instance increases, the quality is often poor or even misleading. To eliminate the influence of the incomplete data, we introduce a diagonal weight matrix \( W^{(v)} \in \mathbb{R}^{N \times N} \), whose diagonal element \( w_{i,i}^{(v)} \) represents the weight of the \( i \)-th instance in the \( v \)-th view. We give weight 1 to the instances that appear in the view, and give lower weight to the missing instances (average filled instances) in the view. We will discuss how to dynamically adjust the weight later in this section. The objective function after adding the weight matrices is:

\[
\mathcal{L} = \sum_{v=1}^{n_v} \|X^{(v)} - U^{(v)}V^{(v)}^T W^{(v)}\|_F^2 + \sum_{v=1}^{n_v} \alpha_v \|V^{(v)} - V^*\|_F^2.
\]

(6)

By assigning different weights to instances in difference views, we can give larger weights to more informative estimations of the missing instances and lower weights to less informative or misleading estimations. Additionally, considering the nature of incomplete views, we adopt \( \ell_1 \) norm (lasso regularization) to enforce the sparsity of the
latent feature matrix, which is robust to noises and outliers and widely used in many algorithms [7, 10].
\[ \mathcal{L} = \sum_{v=1}^{n_v} \left( \| \mathbf{X}^{(v)} - \mathbf{U}^{(v)} \mathbf{V}^{(v)\top} \mathbf{V}^{(v)} \|_F^2 + \sum_{i=1}^{n_v} \alpha_i \| \mathbf{W}^{(v)} (\mathbf{V}^{(v)} - \mathbf{V}_t^*) \|_F^2 + \sum_{i=1}^{n_v} \beta_i \| \mathbf{V}^{(v)} \|_1 \right) \]
(7)
where \( \| \cdot \|_F \) is the \( \ell_1 \) norm and \( \beta_i \) is the trade-off parameter between sparsity and accuracy of reconstruction for view \( v \).
In real-world applications, the data matrices may be too large to fit into the memory. We propose to solve the above optimization problem in an online/streaming fashion with low computational and storage complexity. Note that the objective function \( \mathcal{L} \) can be decomposed as:
\[ \mathcal{L} = \sum_{v=1}^{n_v} \sum_{i=1}^{N_v} \| x_{t,i}^{(v)} - u_{t,i}^{(v)} v_{t,i}^{(v)} \|_F^2 \]
(8)
where \( x_{t,i}^{(v)} \) is the \( i \)-th column of \( \mathbf{X}^{(v)} \) and \( v_{t,i}^{(v)} \in \mathbb{R}^K \) is the \( i \)-th column of \( \mathbf{V}^{(v)\top} \). Clearly, when all the basis matrices \( \mathbf{U}^{(v)} \) are fixed, the calculation of \( v_{t,i}^{(v)} \) and \( v_{t}^{*} \) is independent for different \( i \). This property would allow us to approximate the optimal solution by processing the data chunk by chunk. Let \( \{ \mathbf{X}_{t,i}^{(v)} \} \in \mathbb{R}^{n_v \times D_v}, v = 1, \ldots, n_v \) denote the input data chunk at time \( t \), where \( \mathbf{X}_{t,i}^{(v)} \) is the \( t \)-th data chunk in the \( v \)-th view, and \( s \) is the size of the data chunk (number of instances). Eq. (8) can be rewritten as:
\[ \mathcal{L} = \sum_{v=1}^{n_v} \sum_{i=1}^{N_v} \| (\mathbf{X}_{t,i}^{(v)} - \mathbf{U}^{(v)} \mathbf{V}_{t,i}^{(v)\top}) \mathbf{W}_{t,i}^{(v)} \|_F^2 \]
(9)
where \( \mathbf{V}_{t,i}^{(v)} \in \mathbb{R}^{s \times K} \) and \( \mathbf{W}_{t,i}^{(v)} \in \mathbb{R}^{s \times s} \) are the latent feature matrix and diagonal weight matrix for the \( t \)-th data chunk, respectively.

B. Dynamic Weight Setting for Missing Instances

In the previous discussion, we mentioned that we will fill the missing instances with the average values of the features and assign different weights to them. We would like to assign lower weights to the less informative estimations (averaged instances) and higher weights to the more informative estimations. However, since the entire data cannot be held in memory, the data can only be read in a streaming fashion. Thus, the average values cannot be directly calculated. Instead of filling the missing instances with the global average values, we fill the missing instances with the dynamic (up-to-date) average when we read in a new data point/chunk:
\[ x_{t,i}^{(v)} = \frac{\sum_{k=1}^{t} m_{t,k} x_{k,i}^{(v)}}{\sum_{k=1}^{t} m_{t,k}}, \]
(10)
which can be calculated efficiently for every incoming instances. The weight \( u_{t,i}^{(v)} \) is set dynamically to the up-to-date percentage of the available instances in view \( v \):
\[ u_{t,i}^{(v)} = \begin{cases} 1 & \text{if instance } t \text{ appears in view } v, \\ \sum_{m=1}^{t} m_{t,m} v_{t,i}^{(v)} \end{cases} \]
(11)
We can observe that \( u_{t,i}^{(v)} \) is lower if the estimate of \( x_{t,i}^{(v)} \) is made under a higher percentage of missing instances. Thus, \( u_{t,i}^{(v)} \) represents the quality of the estimated average features. Next, we will describe how to optimize the objective function in an online fashion.

IV. Optimization Algorithms

In this section, we first solve the objective function of OMVC derived in the previous section. We then discuss the one-pass OMVC and the multi-pass OMVC algorithms.

A. Solution

From Eq. (9) we can see that at each time \( t \), we need to optimize \( \{ \mathbf{U}^{(v)} \}, \{ \mathbf{V}_{t,i}^{(v)} \} \) and \( \mathbf{V}_t^* \). However, the objective function is not jointly convex, so we have to update \( \{ \mathbf{U}^{(v)} \}, \{ \mathbf{V}_{t,i}^{(v)} \} \) and \( \mathbf{V}_t^* \) in an alternating way. Thus, there are three subproblems in OMVC described as follows.

1) Optimize \( \{ \mathbf{U}^{(v)} \} \) with \( \{ \mathbf{V}_{t,i}^{(v)} \} \) and \( \mathbf{V}_t^* \) Fixed: From Eq. (9) we can observe that the optimization of \( \mathbf{U}^{(v)} \) is independent for different \( v \) with \( \{ \mathbf{V}_{t,i}^{(v)} \} \) and \( \mathbf{V}_t^* \) fixed. To optimize \( \mathbf{U}^{(v)} \) for a specific view \( v \) at time \( t \), we only need to minimize the following objective:
\[ \mathcal{J}_v^{(t)}(\mathbf{U}^{(v)}) = \sum_{i=1}^{t} \| \mathbf{X}_{t,i}^{(v)} - \mathbf{U}^{(v)} \mathbf{V}_{t,i}^{(v)\top} \mathbf{W}_{t,i}^{(v)} \|_F^2 \]
(12)
\[ \text{s.t. } \mathbf{U}^{(v)} \geq 0 \]
Taking the first-order derivative, the gradient of \( \mathcal{J}_v^{(t)} \) with respect to \( \mathbf{U}^{(v)} \) is
\[ \nabla \mathcal{J}_v^{(t)}(\mathbf{U}^{(v)}) = 2\mathbf{U}^{(v)} \mathbf{A}_v^{(v)} - 2\mathbf{B}_v^{(v)}, \]
(13)
where \( \mathbf{A}_v^{(v)} = \sum_{i=1}^{t} \mathbf{V}_{t,i}^{(v)\top} \mathbf{W}_{t,i}^{(v)} \mathbf{V}_{t,i}^{(v)\top} \), \( \mathbf{B}_v^{(v)} = \sum_{i=1}^{t} \mathbf{X}_{t,i}^{(v)} \mathbf{W}_{t,i}^{(v)} \mathbf{V}_{t,i}^{(v)\top} \), and \( \mathbf{W}_{t,i}^{(v)} = \mathbf{W}_{t,i}^{(v)\top} \mathbf{W}_{t,i}^{(v)} \). Therefore, the Hessian matrix of \( \mathcal{J}_v^{(t)}(\mathbf{U}^{(v)}) \) with respect to \( \mathbf{U}^{(v)} \) is
\[ \mathbf{H} \left[ \mathbf{U}^{(v)} \right] = 2\mathbf{A}_v^{(v)} \]
(14)
Using the second order PGD, the update equation for \( \mathbf{U}^{(v)} \) at time \( t \) is:
\[ \mathbf{U}^{(v)}_{k+1} \leftarrow P \left[ \mathbf{U}^{(v)}_k - \gamma_k \nabla \mathcal{J}_v^{(t)}(\mathbf{U}^{(v)}_k) \mathbf{H}^{-1} \left[ \mathbf{U}^{(v)}_k \right] \right] \]
(15)
where \( k \) is the number of iterations, and \( \gamma_k \) is the step size selected by Armijo rule.

2) Optimize \( \{ \mathbf{V}_{t,i}^{(v)} \} \) with \( \mathbf{V}_t^* \) and \( \{ \mathbf{U}^{(v)} \} \) Fixed: Given \( \mathbf{V}_t^* \) and \( \{ \mathbf{U}^{(v)} \} \) fixed, the optimization of \( \{ \mathbf{V}_{t,i}^{(v)} \} \) is independent for different \( v \). To optimize \( \mathbf{V}_{t,i}^{(v)} \) for the \( v \)-th view, we only need to minimize the following objective:
\[ \mathcal{J}_v^{(v)}(\mathbf{V}_{t,i}^{(v)}) = \| (\mathbf{X}_{t,i}^{(v)} - \mathbf{U}^{(v)} \mathbf{V}_{t,i}^{(v)\top}) \mathbf{W}_{t,i}^{(v)} \|_F^2 + \alpha_i \| \mathbf{W}_{t,i}^{(v)} (\mathbf{V}_{t,i}^{(v)} - \mathbf{V}_t^*) \|_F^2 + \beta_i \| \mathbf{V}_{t,i}^{(v)} \|_1 \]
(16)
\[ \text{s.t. } \mathbf{V}_{t,i}^{(v)} \geq 0 \]
Let \( v_{t,i}^{(v)} \) be the \( i \)-th column of \( \mathbf{V}_{t,i}^{(v)\top} \) and \( u_{t,i}^{(v)} \) be the \( i \)-th diagonal element of \( \mathbf{W}_{t,i}^{(v)} \), then the Hessian matrix of
Algorithm 1: One-pass OMVC with mini-batch mode.

Input: Data matrices of all the incomplete views \( \{X^{(v)}\} \). The number of clusters \( K \) and the batch size \( s \). Parameters \( \{\alpha_i\} \) and \( \{\beta_i\} \).

1. \( \mathbf{A}^{(v)}_t = \mathbf{0}, \mathbf{B}^{(v)}_t = \mathbf{0} \) for each view \( v \).
2. for \( t = 1 : \left| \mathcal{N} \right| / s \) do
3. \( \) Draw \( X^{(v)}_t \) for all the views.
4. \( \) Fill in the missing instances and set the weights according to Eq. (10) (11).
5. \( \) repeat
6. \( \) for \( v = 1 : n_v \) do
7. \( \) Update \( U^{(v)}_t \) according to Eq. (15).
8. \( \) Update \( V^{(v)}_t \) according to Eq. (18).
9. \( \) Calculate \( \mathbf{V}^*_t \) according to Eq. (20).
10. until Convergence;
11. \( A^{(v)}_t = A^{(v)}_{t-1} + \mathbf{V}^{(v)}_t \mathbf{W}^{(v)}_t \mathbf{V}^{(v)}_t^T \)
12. \( B^{(v)}_t = B^{(v)}_{t-1} + \mathbf{X}^{(v)}_t \mathbf{W}^{(v)}_t \mathbf{V}^{(v)}_t^T \)
13. \( \) Extract clustering solution from \( \mathbf{V}^*_t \).

\[ J(V^{(v)}_t) \] with respect to \( V^{(v)}_t \) is
\[ H \left[ V^{(v)}_t \right] = 2u^{(v)}_t \frac{1}{2} \left( U^{(v)}_t U^{(v)} + \alpha_v I_K \right) \] (17)

Using the second-order PGD, the update rule for \( V^{(v)}_t \) is:
\[ v^{(v)}_t \leftarrow P \left[ v^{(v)}_t - \gamma H^{-1} J(v^{(v)}_t) \nabla J(v^{(v)}_t) \right] \] (18)

Here, \( \gamma \) is the step size selected by the Armijo rule.

3) Optimize \( V^*_t \) with \( \{U^{(v)}_t\} \) and \( \{V^{(v)}_t\} \) Fixed: To optimize \( V^*_t \) with \( \{V^{(v)}_t\} \) and \( \{U^{(v)}_t\} \) fixed, we only need to minimize the following objective function:
\[ J(V^*_t) = \sum_{v = 1}^{n_v} \alpha_v \| W^{(v)}_t (V^*_t - V^{(v)}_t) \|_F^2 \text{ s.t. } V^*_t \geq 0 \] (19)

Here, we assume that \( \alpha_v \) is positive. Taking the derivative of the objective \( J \) in Eq. (19) over \( V^*_t \) and set it to 0, we have an exact solution for \( V^*_t \):
\[ V^*_t = \left( \sum_{v = 1}^{n_v} \alpha_v \tilde{W}^{(v)}_t \right)^{1/2} \sum_{v = 1}^{n_v} \alpha_v \tilde{W}^{(v)}_t V^*_t \geq 0 \] (20)

It is worth noting that the inverse of \( \sum_{v = 1}^{n_v} \alpha_v \tilde{W}^{(v)}_t \) can be quickly calculated since it is a positive diagonal matrix.

B. One-Pass OMVC

The complete one-pass algorithm procedure is shown in Algorithm 1. Several important points need to be noted. First, at each time \( t \), we do not need to recompute new \( A^{(v)}_t \) and \( B^{(v)}_t \). We only need to compute \( V^{(v)}_t^T W^{(v)}_t V^{(v)}_t \) and \( X^{(v)}_t W^{(v)}_t V^{(v)}_t^T \), and add them to old \( A^{(v)}_{t-1} \) and \( B^{(v)}_{t-1} \).

\[ A^{(v)}_t = A^{(v)}_{t-1} + \mathbf{V}^{(v)}_t \mathbf{W}^{(v)}_t \mathbf{V}^{(v)}_t^T \] (21)

\[ B^{(v)}_t = B^{(v)}_{t-1} + \mathbf{X}^{(v)}_t \mathbf{W}^{(v)}_t \mathbf{V}^{(v)}_t^T \] (22)

Second, in the algorithm, we need to calculate the inverse of Hessian matrix (with dimension \( K \times K \)) in the iteration. The computation cost of inverse of a matrix will be high if the number of clusters \( K \) becomes large. However, in most of the cases, the number of clusters is limited to a small number. Even if \( K \) is very large in some cases, we can use other ways to approximate the inverse, such as the diagonal approximation [21]. Third, the proposed alternative update procedures for \( \{U^{(v)}_t, V^*_t, V^*_t\} \) converge (see [15]).

C. Multi-Pass OMVC

In data stream scenario, often only one pass over the data is available. In many other applications, it is feasible to do multiple passes. In the one-pass OMVC, the consensus latent feature matrix \( V^* \), which represents the clustering assignment/possibility, is computed in a sequential greedy way. It is expected that for the data points that come first, the performance of clustering may not be satisfactory. However, the multi-pass OMVC gives a chance to improve the performance for those earlier data points.

In the multi-pass OMVC, \( \{V^{(v)}_t\} \) and \( V^* \) can be updated using \( \{U^{(v)}_t\} \) in the previous pass. \( \{A^{(v)}\} \) and \( \{B^{(v)}\} \) from previous pass can be used and updated. Also, the weights for missing instances will be more accurate after the first pass. Thus, the performance of clustering after multiple passes is expected to be better than that of one-pass OMVC.

V. EXPERIMENT

In this section, we compare OMVC with three state-of-the-art methods over four real datasets. More results and details, together with convergence and complexity analysis can be found in the full version of this paper [15].

Datasets: Table I summarizes the datasets. WebKB\(^1\) is a subset of web documents from four universities [18], which contains 1,051 documents with two classes (course or non-course). Two views are extracted for the experiment: 1) content text and 2) anchor text on links. Digit\(^2\) contains 2,000 handwritten numerals (“0”-“9”) extracted from a collection of maps. All views except morphological view are used in the experiments. Reuters\(^3\) contains features of 111,740 documents originally written in five different languages (English, French, German, Spanish and Italian), and their translations, over six topic categories [1]. YouTube\(^4\) contains about 120,000 videos from 31 classes corresponding to 30 popular video games and other games [14]. We select one vision view (512 cuboids histogram features), one audio view (2,000 MFCC features) and one text view (1,000 LDA features from text) in the experiments. We removed some instances with label 31 and kept 92,457 instances to create a more balanced dataset.

Comparison Methods: We compare OMVC with three state-of-the-art methods. The differences between these comparison methods are summarized in Table II, and the details of comparison methods are as follows:

OMVC is the proposed online multi-view clustering method

\(^1\)http://vikas.sindhawani.org/manifoldregularization.html
\(^2\)http://archive.ics.uci.edu/ml/datasets/Multiple+Features
\(^3\)http://archive.ics.uci.edu/ml/machine-learning-databases/00259/
\(^4\)https://archive.ics.uci.edu/ml/datasets/YouTube+Multiview+Video+Games+Dataset

Table I: Summary of the datasets

| Dataset | # of instances | # of views | # of clusters |
|---------|----------------|------------|---------------|
| WebKB   | 1,051          | 2          | 2             |
| Digit   | 2,000          | 5          | 10            |
| Reuters | 111,740        | 5          | 6             |
| YouTube | 92,457         | 3          | 31            |
in this paper. To facilitate comparison, we set $\alpha_s (\beta_s)$ to be the same for all the views. MultiNMF is the state-of-the-art off-line multi-view clustering method based on joint nonnegative matrix factorization [12] for complete views. MIC is one of the most recent works that solve the off-line multi-view clustering problem with incomplete data via weighted joint NMF [16]. ONMF is an online document clustering algorithm for single view using NMF [21]. In order to apply ONMF, we simply concatenate all the normalized views together to form a big single view. We compare two versions of ONMF from the original paper. ONMF-I is the original algorithm that calculates the exact inverse of Hessian matrix, while ONMF-DA uses diagonal approximation for the inverse of Hessian matrix.

In our experiments, normalized mutual information (NMI) is used to measure the clustering performance. Note that all the four datasets are complete. In order to simulate situations with missing instances, we randomly delete instances (0% to 40%) from each view to make the views incomplete. Since all the methods except ONMF have several parameters, we do a grid search for all the parameters in the comparison methods and present the best results obtained. Since MultiNMF, ONMF-I and ONMF-DA cannot handle incomplete views, in order to apply these methods, we fill the missing instances with average features. In the evaluation, we use $K$-means to get the clustering solution from the consensus latent feature matrix. Since $K$-means depends on initialization, we repeat clustering 20 times with random initialization and report the average performance.

A. Results

To show the performance of proposed multi-pass OMVC, we randomly deleted 40% of the instances in each view for all the datasets, and run the comparison methods. The chunk size $s$ for online methods is set to 50 for small datasets and 2000 for large datasets. We report NMI for different passes. The results are shown in Figs. 1-4 and the run times are reported in Table III. It is worth noting that both MIC and MultiNMF are off-line methods with one pass, so the NMI are two horizontal lines in the figures.

From Figs. 1-2 on WebKB and Digit, we can observe that for the three online methods (OMVC, ONMF-DA and ONMF-I), NMI increases as the number of passes increases. The performance of the online methods converges as the number of passes increases. Although the off-line method MIC achieves the best performance, the proposed OMVC gets close performance within a few passes and outperforms the other three comparison methods by a large margin. Even in the first pass, the proposed OMVC already outperforms the other two online methods and even the off-line MultiNMF. This shows that even one-pass OMVC can achieve reasonable performance. From these figures, we can also observe that MIC and OMVC achieve better performance than the other three methods. It is because that both MIC and OMVC utilize a weight matrix for each view to eliminate the influence of the incomplete data and enforce the sparsity of the latent features, while the other three methods do not consider the incompleteness and sparsity of the data.

Figs. 3-4 demonstrate the performance on the two large datasets, Reuters and YouTube. As the data is too large, the two off-line methods (MIC and MultiNMF) cannot be applied. We only report the NMI for the three online methods. In these two figures, OMVC again outperforms the other two online methods in all the passes.

From Figs. 1-4, we can conclude that the proposed OMVC outperforms all the other online methods and perform on par with the best off-line method within a small number of passes. Another interesting observation we can get from the figures is that ONMF-I performs better than ONMF-DA on all the datasets except for YouTube, which indicates that the diagonal approximation sacrifices the accuracy for the computation efficiency in the three datasets.

We also reported the run time for the comparison methods on the four datasets in Table III. From the table, we can see that all the three online methods are much faster than the two off-line methods. Although the ONMF-DA and ONMF-I run faster than the proposed OMVC, OMVC achieves much better performance than ONMF-DA and ONMF-I.

In OMVC, we need to set the size of data chunk. In order to study the performance of OMVC with different chunk sizes, we conducted another set of experiments. Moreover, to show how the instances incomplete rate affects the performance, We ran the comparison methods with different chunk sizes on WebKB and Digit with different incomplete rates and report the NMI after 10 passes in Table IV and Table V. From the tables, we can first observe that OMVC outperforms the other online methods in all the cases and is very close to the best off-line method, if not better. If we look at the performance for different incomplete rates, we can see that as the incomplete rate increases, the performance for all the methods decreases. It is because as the incomplete rate increases, the useful information contains in each view decreases, and all the methods suffer from the incompleteness of views. We can also observe that for each incomplete rate, when the chunk size is too small (e.g., $s = 2$), all the online methods show low performance. This is because the more data in one

| Methods | Multi-view | Incomplete | Sparsity | Online |
|---------|------------|-------------|----------|--------|
| OMVC    | ✓          | ×           | ×        | ×      |
| MultiNMF| ✓          | ✓           |          | ✓      |
| MIC     | ×          | ✓           |          | ×      |
| ONMF-I  | ✓          | ×           | ✓        | ×      |
| ONMF-DA | ×          | ✓           | ✓        | ×      |

| Table III: Run time for different methods |
|------------------------------------------|
| Method      | WebKB | Digit | Reuters | YouTube |
|-------------|-------|-------|---------|---------|
| OMVC/Pass   | 16.69 | 27.47 | 1963.82 | 1547.78 |
| ONMF-IPass  | 14.53 | 31.88 | 1018.97 | 1201.90 |
| ONMF-IPass  | 14.39 | 27.26 | 1213.59 | 2589.94 |
| MIC         | 1134.23 | 2187.88 | -       | -       |
| MultiNMF    | 974.12 | 747.64 | -       | -       |
chunk, the more information we can use to improve the performance. When $s$ is large enough (larger than $K$), the performance of online methods improves. From the tables, we can see that when $s$ is 50 or 250, the performance is already pretty close to the best off-line method. Also, using larger chunks means fewer iterations, which reduces the IO cost significantly comparing with using smaller chunks.

VI. CONCLUSION

In this paper, we present possibly the first attempt to solve the online multi-view clustering with incomplete views where each view may suffer from missing instances. Based on a joint NMF framework, the proposed OMVC learns the latent feature matrices for each individual incomplete view and pushes them towards a common consensus. By giving missing instances lower weights dynamically, OMVC minimizes the negative influences of the missing data. OMVC also enforces the sparsity of the learned latent feature matrices by introducing lasso regularization, which makes the method robust to noises and outliers. Most important, OMVC does not require holding the entire data into memory, which reduces the space complexity dramatically. Extensive experiments conducted on both small and large real data demonstrate the effectiveness of the proposed OMVC method comparing with other state-of-the-art methods.

ACKNOWLEDGMENT

This work is supported in part by NSF through grants IIS-1526499, CNS-1626432, NSFC through grants 61503253, 61672357, and NVIDIA Corporation with the donation of the Titan X GPU used for this research.

REFERENCES

[1] M. R. Amini, N. Usunier, and C. Goutte. Learning from multiple partially observed views - an application to multilingual text categorization. In NIPS, 2009.

[2] D. Bertsekas. Nonlinear Programming. Athena Scientific, 1999.

[3] D. P. Bertsekas. Projected newton methods for optimization problems with simple constraints. SIAM Journal on control and Optimization, 20(2):221–246, 1982.

[4] S. Bickel and T. Scheffer. Multi-View Clustering. In ICDM, 2004.

[5] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang. Diversity-induced multi-view subspace clustering. In CVPR, 2015.

[6] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan. Multi-view Clustering via Canonical Correlation Analysis. In ICML, 2009.

[7] P. Hoyer. Non-negative sparse coding. In 12th IEEE Workshop on Neural Networks for Signal Processing, pages 557–565. IEEE, 2002.

[8] A. Kumar and H. D. III. A co-training Approach for Multi-view Spectral Clustering. In ICML, 2011.

[9] A. Kumar, P. Rai, and H. D. III. Co-regularized Multi-view Spectral Clustering. In NIPS, 2011.

[10] S. Li, Y. Jiang, and Z. Zhou. Partial Multi-View Clustering. In AAAI, 2014.

[11] C. Lin. Projected gradient methods for nonnegative matrix factorization. Neurocomputing, 70:359–364, 2006.

[12] J. Liu, C. Wang, J. Gao, and J. Han. Multi-View Clustering via Joint Nonnegative Matrix Factorization. In SDM, 2013.

[13] B. Long, P. S. Yu, and Z. Zhang. A General Model for Multiple View Unsupervised Learning. In SDM, 2008.

[14] O. Madani, M. Georg, and D. A. Ross. On using nearly-independent feature families for high precision and confidence. Machine Learning, 92:457–477, 2013.

[15] W. Shao, L. He, C.-T. Lu, and P. S. Yu. Online multi-view clustering with incomplete views. Corr arXiv:1611.00481, 2016.

[16] W. Shao, L. He, and P. S. Yu. Multiple Incomplete Views Clustering via Weighted Nonnegative Matrix Factorization with L2,1 Regularization. In ECML PKDD, 2015.

[17] W. Shao, X. Shi, and P. Yu. Clustering on multiple incomplete datasets via collective kernel learning. In ICDM, 2013.

[18] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In ICDM, 2005.

[19] S. Sun. A survey of multi-view machine learning. Neural Computing and Applications, 23(7-8):2031–2038, 2013.

[20] 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, 2010.

[21] F. Wang, P. Li, and A. König. Efficient document clustering via online nonnegative matrix factorizations. In SDM, 2011.

| $s$ | 0% | 20% | 40% |
|-----|----|-----|-----|
| $s=2$ | 0.5852 | 0.3890 | 0.3884 |
| $s=10$ | 0.5999 | 0.3890 | 0.3884 |
| $s=50$ | 0.5994 | 0.3890 | 0.3884 |
| $s=250$ | 0.5259 | 0.3890 | 0.3884 |
| $s=500$ | 0.2686 | 0.3890 | 0.3884 |

| $s$ | 0% | 20% | 40% |
|-----|----|-----|-----|
| $s=2$ | 0.0175 | 0.0175 | 0.0175 |
| $s=10$ | 0.0207 | 0.0207 | 0.0207 |
| $s=50$ | 0.0240 | 0.0240 | 0.0240 |
| $s=250$ | 0.0243 | 0.0243 | 0.0243 |
| $s=500$ | 0.0246 | 0.0246 | 0.0246 |

Table IV: NMI on WebKB with different incomplete rates and chunk sizes

Table V: NMI on Digit with different incomplete rates and chunk sizes