Online Unsupervised Multi-view Feature Selection

Weixiang Shao*, Lifang He†, Chun-Ta Lu*, Xiaokai Wei* and Philip S. Yu*‡
*University of Illinois at Chicago
Email: {wshao4, clu29, xwei2, psyu}@uic.edu
†Shenzhen University, China
Email: lifanghescut@gmail.com
‡Institute for Data Science, Tsinghua University, China

***Abstract*—In this paper, we propose an Online unsupervised Multi-View Feature Selection method, OMVFS, which deals with large-scale/streaming multi-view data in an online fashion. OMVFS embeds unsupervised feature selection into a clustering algorithm via nonnegative matrix factorization with sparse learning. It further incorporates the graph regularization to preserve the local structure information and help select discriminative features. Instead of storing all the historical data, OMVFS processes the multi-view data chunk by chunk and aggregates all the necessary information into several small matrices. By using the buffering technique, the proposed OMVFS can reduce the computational and storage cost while taking advantage of the structure information. Furthermore, OMVFS can capture the concept drifts in the data streams. Extensive experiments on four real-world datasets show the effectiveness and efficiency of the proposed OMVFS method. More importantly, OMVFS is about 100 times faster than the off-line methods.**

I. Introduction

In many real-world applications, data are often with multiple modalities or coming from multiple sources. Such data are called multi-view data. Usually, multiple views provide complementary information for the semantically same data. Multi-view learning was proposed to combine different views to obtain better performance than relying on just one single view [1], [2]. However, many of the views may come from high-dimensional spaces (such as language vocabularies). Thus, accurate unsupervised feature selection on these high-dimensional multi-view data is crucial to many applications such as model interpretation and storage reduction.

Feature selection has been studied for decades [3]. It can be categorized into supervised feature selection and unsupervised feature selection in terms of the label availability. Supervised feature selection [4] uses the class labels to effectively select discriminative features to distinguish samples from different classes. As the data explode, most of the data are unlabeled and it is expensive to obtain the labels. Recently, several approaches on unsupervised feature selection has been proposed [5], [6]. Without the label information, most of the unsupervised feature selection methods combine generated pseudo labels with sparse learning [7], [8]. In this paper, we will only focus on the unsupervised feature selection.

Most recently, as complementary information can be obtained from different views, unsupervised multi-view feature selection has drawn lots of attention [9]–[11]. For example, [10] is the first to use spectral clustering and $\ell_{2,1}$-norm regression to multi-view data in social media. [9] integrates all features and learns the weights for every feature with respect to each cluster individually via a new joint structured sparsity-inducing norms. For image and text data, [11] uses image local learning regularized orthogonal nonnegative matrix factorization to learn pseudo labels and simultaneously perform robust joint $\ell_{2,1}$-norm minimization to select discriminative features.

However, several challenges prevent us from applying existing unsupervised feature selection methods to the real-world multi-view data: 1) As information explodes, multi-view data may contain too many instances or the feature dimensionality may be too large, such that the data cannot fit in memory. How to select useful features with limited memory space is the first challenge. 2) In many real-world applications, the data may come in as streams and concept drift [12] may happen. How to select features from streaming data and handle the concept drift is the second challenge. 3) Multi-view data always exhibits the heterogeneity of the features. Different views may share some consistent and complementary information. The third challenge is how to combine features from different views while taking advantages of the consistent and complementary information to improve the feature selection in the situation when the data are too big or come in as streams.

To deal with all three challenges, we propose OMVFS, the Online unsupervised Multi-View Feature Selection method. OMVFS embeds unsupervised feature selection into a Nonnegative Matrix Factorization (NMF) based clustering objective function. It further adopts the graph regularization to preserve the local structure information and help select discriminative features. By learning a consensus clustering indicator matrix, OMVFS integrates all the views in different feature spaces. Instead of storing all the historical data, OMVFS processes the multi-view data chunk by chunk and aggregates information from all the previous data into several small matrices. The aggregated matrices will be used to learn the feature selection matrices and updated as new data arrive.

The contributions of this paper can be summarized as following: 1) It is the first attempt to solve online unsupervised multi-view feature selection problem on large-scale/streaming data. 2) OMVFS processes the data in an online fashion and aggregates information about all the previous data into small matrices. The aggregated information can be used to help feature selection in the future. Thus, OMVFS can greatly reduce

Corresponding author: Lifang He, lifanghescut@gmail.com

the memory requirement and scale up to large data without appreciable sacrifice of performance. 3) By using a buffering technique, OMVFS can reduce the computational and storage cost while taking advantages of the structure information. Furthermore, it can capture the concept drifts in data streams. 4) Extensive experiments on four datasets demonstrate that the effectiveness of OMVFS is comparative and even better than the best off-line method. More importantly, OMVFS is about 100 times faster than the off-line methods.

II. Preliminaries

A. Problem Description

Throughout this paper, matrices are written as boldface capital letters (*e.g.*, $\mathbf{M} \in \mathbb{R}^{n\times m}$) and vectors are denoted as boldface lowercase letters (*e.g.*, $\mathbf{m}_i$). $\|\cdot\|_F$ is the matrix Frobenius norm and $Tr(\cdot)$ is the trace of a square matrix. The $\ell_{2,1}$-norm is defined as $\|\mathbf{M}\|_{2,1} = \sum_{i=1}^{n} \mathbf{m}_i = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} \mathbf{M}_{i,j}^2}$.

Assume we are given a dataset with N instances in n_v views $\{\mathbf{X}^{(v)}, v = 1, 2, ..., n_v\}$, where $\mathbf{X}^{(v)} \in \mathbb{R}_+^{N\times D_v}$ and D_v represent the nonnegative data and it's dimenson in the v-th view. Our goal is to leverage complementary information from multi-view data to select p_v features from the v-th view, while dealing with high-dimensional and large-scale problems.

B. Unsupervised Feature Selection using NMF

Nonnegative Matrix Factorization (NMF) has been widely used in unsupervised learning such as clustering and dimension reduction. Let $\mathbf{X} \in \mathbb{R}_+^{N\times D}$ denote a nonnegative data matrix, where each row represents an instance and each column represents one normalized attribute (each column $\|\mathbf{X}_{.,i}\|_2 = 1$). Assume K is the number of clusters, the NMF based clustering can be formulated as below:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 \quad s.t.\ \mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{U} \geq 0, \mathbf{V} \geq 0, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}_+^{N\times K}$ and $\mathbf{V} \in \mathbb{R}_+^{D\times K}$. The orthogonality constraint on $\mathbf{U}$ can be seen as a relaxed form from the original clustering indicator constraint, where $\mathbf{U} \in \{0, 1\}^{N\times K}, \mathbf{U}^T\mathbf{1} = \mathbf{1}$.

Another advantage of this orthogonality constraint on $\mathbf{U}$ is to allow us to perform feature selection using $\mathbf{V}$. As proposed in [13], the feature selection can be achieved by adding the $\ell_{2,1}$-norm on $\mathbf{V}$ to the objective function:

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{V}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \beta\|\mathbf{V}\|_{2,1} \\ &s.t.\ \mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{U} \geq 0, \mathbf{V} \geq 0, \end{aligned} \tag{2}$$

where β is the parameter to control the sparsity of $\mathbf{V}$. By adding the $\ell_{2,1}$-norm on $\mathbf{V}$, we force some of the rows in $\mathbf{V}$ close to 0. Thus, we can achieve feature selection by sorting the features according to the row norms of $\mathbf{V}$ in descending order, and selecting the top ranked ones. To facilitate discussions, we call $\mathbf{U}$ the ***cluster indicator matrix***, and $\mathbf{V}$ the ***feature selection matrix***.

III. Method

A. Objective of OMVFS

Given data in n_v views $\{\mathbf{X}^{(v)} \in \mathbb{R}_+^{N\times D_v}, v = 1, 2, ..., n_v\}$, we aim to find a feature selection matrix $\mathbf{V}^{(v)}$ for each view v and a consensus cluster indicator matrix $\mathbf{U}$, which integrates information of all the views. Following the constrained NMF framework in Section II-B, we can form the objective function

$$\begin{aligned} &\min_{\mathbf{U},\{\mathbf{V}^{(v)}\}} \sum_{v=1}^{n_v} \left(\|\mathbf{X}^{(v)} - \mathbf{U}{\mathbf{V}^{(v)}}^T\|_F^2 + \beta_v\|\mathbf{V}^{(v)}\|_{2,1}\right) \\ &s.t.\ \mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{U} \geq 0, \mathbf{V}^{(v)} \geq 0, v = 1, 2, ..., n_v, \end{aligned} \tag{3}$$

where β_v is the parameter that controls the sparsity of $\mathbf{V}^{(v)}$.

To take advantage of local manifold information from the structure of the original data $\{\mathbf{X}^{(v)}\}$, *i.e.*, similar data instances should have similar labels, we add the spectral clustering objective term for every view

$$\min Tr({\mathbf{U}^{(v)}}^T\mathbf{L}^{(v)}\mathbf{U}^{(v)}) \tag{4}$$

where $\mathbf{L}^{(v)} = {\mathbf{D}_\mathbf{w}}^{(v)} - \mathbf{W}^{(v)}$ is the Laplacian matrix for the v-th view, $\mathbf{W}^{(v)} \in \mathbb{R}^{N\times N}$ is the similarity matrix based on $\mathbf{X}^{(v)}$ and ${\mathbf{D}_\mathbf{w}}^{(v)}$ is a diagonal matrix with its diagonal elements as the row sums of $\mathbf{W}^{(v)}$. Now, we can obtain the objective function for OMVFS as:

$$\begin{aligned} &\min_{\mathbf{U},\{\mathbf{V}^{(v)}\}} \sum_{v=1}^{n_v} \left(\|\mathbf{X}^{(v)} - \mathbf{U}{\mathbf{V}^{(v)}}^T\|_F^2 + \alpha_v tr(\mathbf{U}^T\mathbf{L}^{(v)}\mathbf{U}) + \beta_v\|\mathbf{V}^{(v)}\|_{2,1}\right) \\ &s.t.\ \mathbf{U}^T\mathbf{U} = \mathbf{I},\ \mathbf{U} \geq 0,\ \mathbf{V}^{(v)} \geq 0,\ v = 1, 2, ..., n_v \end{aligned} \tag{5}$$

where α_v is the importance of the spectral clustering term for the v-th view.

To solve the problem in Eq. (5), we need to have the entire data $\{\mathbf{X}^{(v)}\}$ in memory. However, in real-world applications, the data may be too large to fit into the memory or may only come on stream. Hence, it is crucial to solve the above problem in an incremental way. Let $\mathbf{X}_t^{(v)} \in \mathbb{R}_+^{m\times D_v}$ denote the data received at time t in the v-th view, and $\mathbf{X}_{[t]}^{(v)} \in \mathbb{R}_+^{mt\times D_v}$ denote all the data received up to time t, where m is the number of instances (size of the data chunk) received at each time. The objective function at time t is:

$$\begin{aligned} &\min_{\mathbf{U}_{[t]},\{\mathbf{V}^{(v)}\}} \sum_{v=1}^{n_v} \left(\|\mathbf{X}_{[t]}^{(v)} - \mathbf{U}_{[t]}{\mathbf{V}^{(v)}}^T\|_F^2 + \alpha_v Tr(\mathbf{U}_{[t]}^T\mathbf{L}_{[t]}^{(v)}\mathbf{U}_{[t]})\right) \\ &\qquad + \sum_{v=1}^{n_v} \beta_v\|\mathbf{V}^{(v)}\|_{2,1} \\ &s.t.\ {\mathbf{U}_{[\mathbf{t}]}}^T\mathbf{U}_{[t]} = \mathbf{I},\ \mathbf{U}_{[t]} \geq 0,\ \mathbf{V}^{(v)} \geq 0,\ v = 1, 2, ..., n_v, \end{aligned} \tag{6}$$

where $\mathbf{U}_{[t]} \in \mathbb{R}_+^{mt\times K}$ is the consensus cluster indicator matrix for all the instances received up to time t and $\mathbf{L}_{[t]}^{(v)} \in \mathbb{R}^{mt\times mt}$ is the Laplacian matrix constructed from $\mathbf{X}_{[t]}^{(v)}$.

B. Optimization

To solve the objective function of OMVFS in Eq. (6), we first rewrite the optimization problem as follows

$$\begin{aligned} &\min_{\mathbf{U}_{[t]},\{\mathbf{V}^{(v)}\}} \sum_{v=1}^{n_v} \left(\|\mathbf{X}_{[t]}^{(v)} - \mathbf{U}_{[t]}{\mathbf{V}^{(v)}}^T\|_F^2 + \alpha_v Tr(\mathbf{U}_{[t]}^T\mathbf{L}_{[t]}^{(v)}\mathbf{U}_{[t]})\right) \\ &\qquad + \sum_{v=1}^{n_v} \beta_v\|\mathbf{V}^{(v)}\|_{2,1} + \gamma\|\mathbf{U}_{[t]}^T\mathbf{U}_{[t]} - \mathbf{I}\|_F^2 \\ &s.t.\ \mathbf{U}_{[t]} \geq 0,\ \mathbf{V}^{(v)} \geq 0,\ v = 1, 2, ..., n_v \end{aligned} \tag{7}$$

where $\gamma > 0$ is a parameter to control the orthogonality condition. In practice, γ should be large enough to ensure the orthogonality is satisfied (γ is set to 10^7 throughout the experiments). From Eq. (7), we can see that at each time t, we need to optimize $\mathbf{U}_{[t]}$ and $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$. However, the objective function is not jointly convex, so we have to update

$\mathbf{U}_{[t]}$ and $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$ in an alternating way. Thus, there are two subproblems in OMVFS:

1) Optimize $\mathbf{U}_{[t]}$ with $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$ Fixed: To optimize $\mathbf{U}_{[t]}$ with $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$ fixed at time t, we only need to minimize the following objective:

$$\mathcal{J}_t(\mathbf{U}_{[t]}) = \sum_{v=1}^{n_v} \left(\|\mathbf{X}_{[t]}^{(v)} - \mathbf{U}_{[t]}\mathbf{V}^{(v)^T}\|_F^2 + \alpha_v tr(\mathbf{U}_{[t]}^T \mathbf{L}_{[t]}^{(v)} \mathbf{U}_{[t]}) \right) + \gamma \|\mathbf{U}_{[t]}^T \mathbf{U}_{[t]} - \mathbf{I}\|_F^2 \quad s.t. \quad \mathbf{U}_{[t]} \geq 0 \tag{8}$$

After deriving the gradient of $\mathcal{J}_t$ with respect to $\mathbf{U}_{[t]}$ and using the Karush-Kuhn-Tucker (KKT) complementary condition for the nonnegativity constraint on $\mathbf{U}_{[t]}$, we can get the update rule for $\mathbf{U}_{[t]}$:

$$(\mathbf{U}_{[t]})_{i,k} \leftarrow (\mathbf{U}_{[t]})_{i,k} \sqrt{\frac{(\sum_{v=1}^{n_v} \mathbf{X}_{[t]}^{(v)} \mathbf{V}^{(v)} + \lambda \mathbf{U}_{[t]} + \mathbf{M}_{[t]}^{-} \mathbf{U}_{[t]})_{i,k}}{(\mathbf{U}_{[t]} \sum_{v=1}^{n_v} \mathbf{V}^{(v)^T} \mathbf{V}^{(v)} + \gamma \mathbf{T}_{[t]} + \mathbf{M}_{[t]}^{+} \mathbf{U}_{[t]})_{i,k}}} \tag{9}$$

where $\mathbf{T}_{[t]} = \mathbf{U}_{[t]}\mathbf{U}_{[t]}^T\mathbf{U}_{[t]}$, $\mathbf{M}_{[t]} = \mathbf{M}_{[t]}^{+} - \mathbf{M}_{[t]}^{-} = \sum_{v=1}^{n_v} \alpha_v \mathbf{L}_{[t]}^{(v)}$, $(\mathbf{M}_{[t]}^{+})_{i,j} = \frac{1}{2}\left(\|(\mathbf{M}_{[t]})_{i,j}\| + (\mathbf{M}_{[t]})_{i,j}\right)$, and $(\mathbf{M}_{[t]}^{-})_{i,j} = \frac{1}{2}\left(\|(\mathbf{M}_{[t]})_{i,j}\| - (\mathbf{M}_{[t]})_{i,j}\right)$.

2) Optimize $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$ with $\mathbf{U}_{[t]}$ Fixed: From Eq. (7), we can observe that the optimization of $\mathbf{V}^{(v)}$ is independent with different v when $\mathbf{U}_{[t]}$ is fixed. We only need to minimize the following objective function for the v-th view:

$$\mathcal{J}_t(\mathbf{V}^{(v)}) = \sum_{i=1}^{t} \|\mathbf{X}_i^{(v)} - \mathbf{U}_i \mathbf{V}^{(v)^T}\|_F^2 + \beta_v \|\mathbf{V}^{(v)}\|_{2,1} \quad s.t. \quad \mathbf{V}^{(v)} \geq 0 \tag{10}$$

where $\mathbf{X}_i^{(v)}$ is the data chunk received at time i and $\mathbf{U}_i$ is the cluster indicator matrix for data received at time i.

Taking the first-order derivative, the gradient of $\mathcal{J}_t$ with respect to $\mathbf{V}^{(v)}$ is

$$\frac{\partial \mathcal{J}_t}{\partial \mathbf{V}^{(v)}} = 2\mathbf{V}^{(v)} \sum_{i=1}^{t} \mathbf{U}_i^T \mathbf{U}_i - 2 \sum_{i=1}^{t} \mathbf{X}_i^{(v)^T} \mathbf{U}_i + \beta_v \mathbf{D}^{(v)} \mathbf{V}^{(v)} \tag{11}$$

where $\mathbf{D}^{(v)}$ is a diagonal matrix with the j-th diagonal element given by $\mathbf{D}_{j,j}^{(v)} = \frac{1}{\|\mathbf{v}_j^{(v)}\|_2}$ and $\mathbf{v}_j^{(v)}$ is the j-th row of $\mathbf{V}^{(v)}$.

For the sake of convenience, we introduce two terms:

$$\mathbf{A}_t = \sum_{i=1}^{t} \mathbf{U}_i^T \mathbf{U}_i = \mathbf{A}_{t-1} + \mathbf{U}_t^T \mathbf{U}_t \tag{12}$$

$$\mathbf{B}_t^{(v)} = \sum_{i=1}^{t} \mathbf{X}_i^{(v)^T} \mathbf{U}_i = \mathbf{B}_{t-1}^{(v)} + \mathbf{X}_t^{(v)^T} \mathbf{U}_t \tag{13}$$

Both $\mathbf{A}_t$ and $\mathbf{B}_t^{(v)}$ can be computed incrementally with low storage. Using the KKT complementary condition, we get the update rule for $\mathbf{V}^{(v)}$:

$$\mathbf{V}_{j,k}^{(v)} \leftarrow \mathbf{V}_{j,k}^{(v)} \sqrt{\frac{\left(\mathbf{B}_{t-1}^{(v)} + \mathbf{X}_t^{(v)^T} \mathbf{U}_t\right)_{j,k}}{\left(\mathbf{V}^{(v)}(\mathbf{A}_{t-1} + \mathbf{U}_t^T \mathbf{U}_t) + \frac{1}{2}\beta_v \mathbf{D}^{(v)} \mathbf{V}^{(v)}\right)_{j,k}}} \tag{14}$$

C. Further Optimization Using Buffering

From the update rules (9), we observe that the update process for $\mathbf{U}_{[t]}$ still needs all the data up to time t to reside in the memory. Thus, the memory usage will continue growing as more data come in. To overcome this deficiency, we adopt the buffering technique by keeping a limited number of samples for each update step. Another benefit of adopting buffering is to capture the concept drift in streaming data. The intuition behind buffering is based on the assumption: the incoming data are more related with the recent data than the far more old data, and the underlying concept distribution of incoming data are more similar to those of recent data.

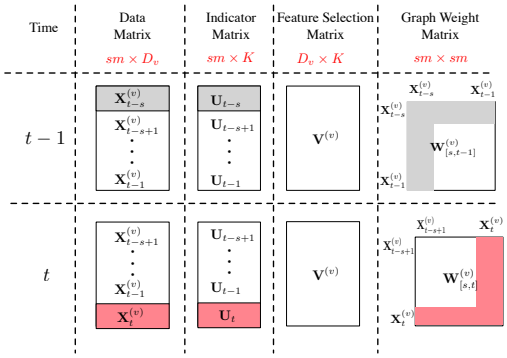


Fig. 1: The detailed update process for major matrices in OMVFS at time t. When the buffer is full, data for the oldest chunk (gray parts) will be discarded, and incoming data (red parts) will be added.

Thus, we consider the information only within a time window. Assume that the buffer size is s, we define $\mathbf{X}_{[s,t]}^{(v)} = [\mathbf{X}_{t-s+1}^{(v)}; \ldots; \mathbf{X}_t^{(v)}] \in \mathbb{R}_+^{sm \times D_v}$ as the most recent s data chunks received up to time t and $\mathbf{U}_{[s,t]} = [\mathbf{U}_{t-s+1}; \ldots; \mathbf{U}_t] \in \mathbb{R}_+^{sm \times K}$ as the cluster indicator matrix for them. $\mathbf{W}_{[s,t]}^{(v)} \in \mathbb{R}^{sm \times sm}$ and $\mathbf{L}_{[s,t]}^{(v)} \in \mathbb{R}^{sm \times sm}$ are the similarity matrix constructed from the buffer and the Laplacian matrix from $\mathbf{W}_{[s,t]}^{(v)}$. Note that, for the similarity matrix $\mathbf{W}_{[s,t]}^{(v)}$, we do not need to recompute the similarities among all the sm instances ($sm \times sm$ pairwise similarities). We only need to calculate the similarities between the m new instances with all the sm instances ($m \times sm$ pairwise similarities). Thus, we can redefine the Eq. (8) by substituting $\mathbf{X}_{[t]}^{(v)}$, $\mathbf{U}_{[t]}$ and $\mathbf{L}_{[t]}^{(v)}$ with the above new definitions. The new update rule for $\mathbf{U}_{[s,t]}$ is

$$(\mathbf{U}_{[s,t]})_{i,k} \leftarrow (\mathbf{U}_{[s,t]})_{i,k} \sqrt{\frac{(\sum_{v=1}^{n_v} \mathbf{X}_{[s,t]}^{(v)} \mathbf{V}^{(v)} + \lambda \mathbf{U}_{[s,t]} + \mathbf{M}_{[s,t]}^{-} \mathbf{U}_{[s,t]})_{i,k}}{(\mathbf{U}_{[s,t]} \sum_{v=1}^{n_v} \mathbf{V}^{(v)^T} \mathbf{V}^{(v)} + \gamma \mathbf{T}_{[s,t]} + \mathbf{M}_{[s,t]}^{+} \mathbf{U}_{[s,t]})_{i,k}}} \tag{15}$$

For the feature selection matrices $\{\mathbf{V}^{(v)}\}_{v=1}^{n_v}$, we fortunately observe that $\mathbf{A}_t$ and $\{\mathbf{B}_t^{(v)}\}_{v=1}^{n_v}$ accumulate the information of all the previous t chunks. Therefore, we can still use update rule (14) without any modification, which means that employing the buffering technique has no effect on updating the feature selection matrices.

D. Algorithm

The complete algorithm procedure is shown in Algorithm 1. There are several things need to be clarified. First, at the beginning of the algorithm ($t = 0$), OMVFS will initialize the buffered data $\mathbf{X}_{[s,t]}^{(v)}$, the cluster indicator matrix for buffered data $\mathbf{U}_{[s,t]}$ and similarity matrix for buffered data $\mathbf{W}_{[s,t]}^{(v)}$ with empty matrices. The feature selection matrices $\{\mathbf{V}^{(v)}\}$ will also be initialized randomly at $t = 0$.

Second, as the data come in, if the buffer is full, OMVFS will remove the oldest data (*e.g.*, $\mathbf{X}_{t-s}^{(v)}$), and concatenate the new data to the buffer. The procedure for updating the matrices in the buffer is shown in Fig. 1. The grey shadow areas are

Algorithm 1: OMVFS algorithm.

Input: Data matrices $\{\mathbf{X}^{(v)}\}$. The number of clusters K, the batch size m, the buffer size s. Parameters $\{\alpha_v\}$ and $\{\beta_v\}$.
Output: Feature selection matrices $\{\mathbf{V}^v\}$.

Initialize $\mathbf{V}^{(v)}$ randomly for each view v.
Initialize $\mathbf{X}^{(v)}_{[s,0]}$, $\mathbf{U}_{[s,0]}$, $\mathbf{W}^{(v)}_{[s,0]}$ as empty matrices.
Set $\mathbf{A}^{(v)}_0 = \mathbf{0}$, $\mathbf{B}^{(v)}_0 = \mathbf{0}$ for each view v.
for $t = 1 : \lceil N/m \rceil$ **do**
  Draw $\mathbf{X}^{(v)}_t$ for all the views.
  Initialize $\mathbf{U}_t$ randomly.
  for $v = 1 : n_v$ **do**
    Construct $\mathbf{X}^{(v)}_{[s,t]}$, $\mathbf{U}_{[s,t]}$, $\mathbf{W}^{(v)}_{[s,t]}$ and $\mathbf{L}^{(v)}_{[s,t]}$.
  repeat
    Update $\mathbf{U}_{[s,t]}$ according to Eq. (15).
    for $v = 1 : n_v$ **do**
      Update $\mathbf{V}^{(v)}$ according to Eq. (14).
  until *Convergence*;
  $\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{U}_t^T \mathbf{U}_t$
  $\mathbf{B}^{(v)}_t = \mathbf{B}^{(v)}_{t-1} + \mathbf{X}^{(v)T}_t \mathbf{U}_t$
for $v = 1 : n_v$ **do**
  Sort the features of $\mathbf{X}^{(v)}$ according to the ℓ_2-norm of the rows in $\mathbf{V}^{(v)}$ in descending order.

the parts for the oldest data, and the red shadow areas are the parts for the incoming new data.

1) Convergence Analysis: Although the objective function for the proposed OMVFS at time t is not jointly convex with $\mathbf{U}_{[s,t]}$ and $\{\mathbf{V}^{(v)}\}$, the alternating optimization strategy used in OMVFS is guaranteed to converge to a local minimum [14]. The proof is similar to that of Theorem 1 in [15] and we omit it. It is important to note that our method essentially applies stochastic gradient descent to the new coming data. Since we perform a stochastic approximation for minimizing an objective function that is written as a sum of differentiable functions. This method is expected to decrease the objective function in every iteration on the new data [16]. Therefore, the same convergence proof can be adapted to the problem setting and algorithm design that considered here. In the experiments, we also find that the proposed OMVFS method converges within 200 iterations for all the datasets used.

2) Complexity Analysis: We can prove that the overall time complexity of OMVFS is approximately $O(tn_vNDK)$, where t is the average number of iterations to converge, D is the average feature dimension for all the views. More details can be found in the full version of this paper [17]. Most of the offline methods require at least $O(n_vND)$ space, however OMVFS only requires approximately $O(n_vsmD)$ memory, which makes OMVFS suitable for large-scale/streaming data.

IV. Experiments and Results

In this section, we compare the OMVFS approach with five baseline methods over four real-world datasets. More details can be found in the full version of this paper [17].

Dataset CNN and FOX[1] are the two small datasets containing news articles. Each articles can be represented in text view and image view. YouTube[2], is one of the largest public multi-view dataset containing about 120,000 videos in 31 classes. We use unigrams extracted from the video tags as the text view and MFCC features as the audio view. Reuters[3] contains 111,740 documents in two languages/views (English and French). The summary of these four datasets is shown in Table I.

TABLE I: Summary of the datasets

Dataset	# Instance	# Feature	# Class
CNN	$2,107$	$996 + 35,719$	7
FOX	$1,523$	$996 + 27,072$	4
YouTube	$95,343$	$2,000 + 94,012$	31
Reuters	$111,740$	$21,531 + 24,893$	6

TABLE II: Summary of the comparison methods

Methods	Multi-view	Online	Direct Embedding
LapScore	×	×	–
EUFS	×	×	✓
MVUFS	✓	×	×
FSDS	×	✓	×
OMVFS	✓	✓	✓

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

OMVFS is the online unsupervised multi-view feature selection method proposed in this paper [4]. **LapScore** is a single view unsupervised feature selection method, which evaluates the importance of a feature via its power of locality preservation [18]. **EUFS** embeds unsupervised feature selection into a clustering algorithm via sparse learning for single view data [13]. **FSDS** is the most recent online unsupervised feature selection algorithms for single view data [19]. **MVUFS** is the most advanced off-line unsupervised feature selection for multi-view data based on orthogonal NMF [20].

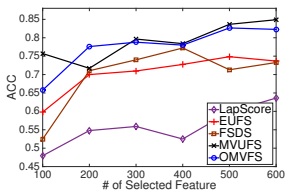
In our experiments, Accuracy (ACC) and Normalized Mutual Information (NMI) are used to measure the clustering performance. For all the comparison methods, the multi-view spherical K-means algorithm [21] is applied on the selected features for evaluation. Since LapScore, EUFS and FSDS only work for single view data, in the experiments, we apply these three methods on each of the view to select features.

Most of the comparison methods use kernel matrices, thus we used the same kernal matrices as stated in the original papers. For OMVFS, we used the Gaussian kernel. If not stated, for the online methods (FSDS and OMVFS) the size of data chunk is set to 200 for small datasets and 1000 for large datasets. The buffer size is set to 2 and γ is set to 10^7 for OMVFS. For brevity, the parameters α_v and β_v are all set equally for different views. For FSDS, we do grid search in $\{1, 2, ..., 10\}$ for the index of parameter α. For all the other parameters in the comparison methods, we do grid search in $\{10^{-2}, 10^{-1}, ..., 10^2\}$. We also vary the number of selected features from 100 to 600 for all the views. The performance of the best parameter setting is reported for all the methods.

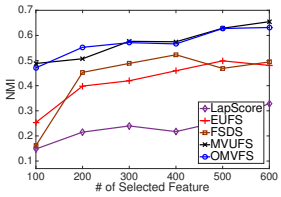
A. Results on Small Datasets

We first apply all the methods to two small datasets, FOX and CNN. Fig. 2 shows the performance of all the methods on FOX and CNN with different numbers of selected features.

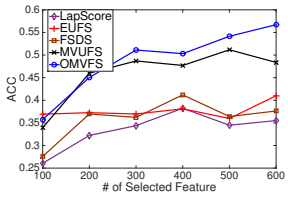
[1]https://sites.google.com/site/qianmingjie/home/datasets/
[2]https://goo.gl/pPZ6dI
[3]http://archive.ics.uci.edu/ml/machine-learning-databases/00259/
[4]code available at: https://github.com/software-shao/



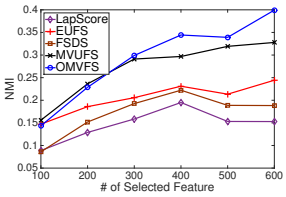
(a) ACC for FOX



(b) NMI for FOX

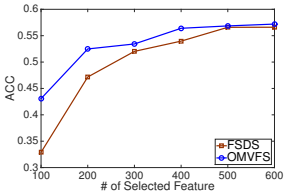


(c) ACC for CNN

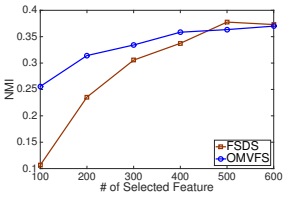


(d) NMI for CNN

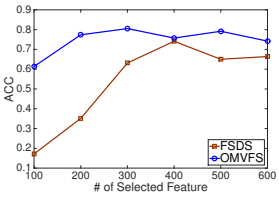
Fig. 2: Performance with different # features on small datasets.



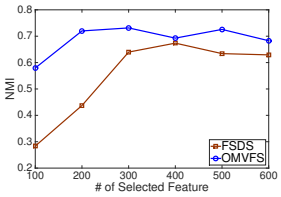
(a) ACC for Reuters



(b) NMI for Reuters



(c) ACC for YouTube



(d) NMI for YouTube

Fig. 3: Performance with different # features on large datasets.

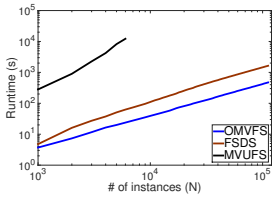
From the Fig. 2, we can observe that as the number of selected feature increases, the performance will increase for most of the cases. For both datasets, the two multi-view feature selection methods, MVUFS (off-line) and OMVFS (online), outperform the single view feature selection methods. This observation suggests that combining different views would benefit the feature selection for multi-view data.

For the FOX data, we can observe from Fig. 2a and Fig. 2b that the performance of OMVFS is very close and even better than that of the best off-line method. FSDS and EUFS performance better than LapScore. Although FSDS may suffer from low performance when including only 100 features, its performance will increase dramatically when including more than 200 features. This may indicate that the online FSDS selects less discriminative features as the top features, however, more discriminative features will be included if we increase the size of the feature set. For the CNN data, we can see that the proposed OMVFS has similar performance as MVUFS when the feature size is less than 300. However, OMVFS outperforms MVUFS if more than 300 features are included. Comparing the statics of CNN and FOX data, we can find that CNN has higher feature dimensions and more classes than FOX. The proposed OMVFS may select more discriminative features in data with higher dimensions than MVUFS.

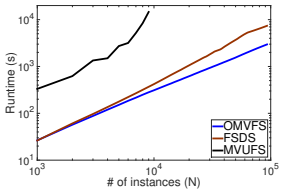
From the results on FOX and CNN, we can conclude that OMVFS outperforms all the single view feature selection methods. It also get close or even better performance than the most advanced off-line multi-view feature selection methods.

B. Result on Large Datasets

Since the proposed OMVFS is designed for large-scale multi-view datasets, we test OMVFS on the two largest public available multi-view datasets. We also compare OMVFS with the most recent online single view feature selection method, FSDS. The results on both datasets are reported in Fig. 3. However, due to the extremely high runtime and memory consumption, LapScore, EUFS and MVUFS cannot be applied to the two large datasets.



(a) Runtime on Reuters



(b) Runtime on YouTube

Fig. 4: Runtime v.s. data size on Reuters and YouTube Data.

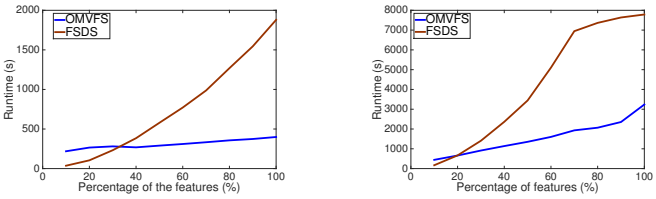
From Fig. 3, it can be easily observed that the proposed OMVFS methods achieves better performance than FSDS in most cases. Especially when the number of selected feature is small (*i.e.*, less than 300), OMVFS is much better than FSDS. For example, on YouTube data, when we only select 100 features, the NMI for OMVFS is about 0.58, while the NMI for FSDS is only 0.28. This observation suggests that unlike FSDS, which tends to include less discriminative features in the top selected feature set, the proposed OMVFS selects more discriminative features even in a small set of selected features.

C. Scalability Comparison

In order to show the scalability of the proposed OMVFS method, we run OMVFS on the Reuters and YouTube data under different data sizes (number of instances) and different feature dimensions in Fig. 4 and Fig. 5, respectively.

MVUFS and FSDS are selected for the runtime comparison, since MVUFS is the only one that deals with multi-view data and FSDS is the only one that handles large-scale/streaming data. We run FSDS on each view and report the total runtime.

Due to the high runtime and memory consumption, we only report the runtime for MVUFS under data size 10^4. From Fig. 4, we can observe that the two online methods, FSDS and OMVFS, are much faster than the off-line MVUFS and the runtime of the two online methods are linear with respect to the data size. It takes about 3 seconds for FSDS and OMVFS on data with size 1000 on Reuters data, while MVUFS takes about 300 seconds. Like the FSDS method, the proposed OMVFS only uses the current data chunks and all the historical data is aggregated. OMVFS considers all the views



(a) Runtime on Reuters (b) Runtime on YouTube

Fig. 5: Runtime v.s. feature dimension on Reuters and YouTube Data.

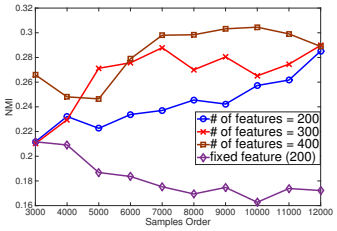


Fig. 6: Concept drift test across the data stream for OMVFS

simultaneously while FSDS can only work on single view. Although the two online methods have similar runtime with small data, OMVFS uses less time as the data size increases. Further, it can be found that OMVFS is about 100 times faster than MVUFS on both datasets.

Due to the high memory usage and slow runtime of MVUFS, we only report the runtime for OMVFS and FSDS in Fig. 5. From Fig. 5, we can observe that although FSDS is faster when we only sample small number of features (10 % or 20%), the runtime of OMVFS grows much slower than FSDS as the dimension of features increases. For Reuters data, the runtime of OMVFS is only 480 seconds when the number of features increases to 100%. Thus, OMVFS is faster when the feature dimension becomes really large.

D. Stability under Concept Drift

It is well-known that online/streaming algorithms are generally sensitive to the order of data, or concept drift [12]. To test the performance of OMVFS in such scenarios, we use 12,000 instances from Reuters data. We randomly create unbalanced data with two dominant classes and randomly change the dominant classes every 3,000 instances in the data stream. OMVFS is then applied to the data stream with concept drifts. As the data came in, we report the performance of OMVFS with different sizes of feature set in Fig. 6. We also compare against a scheme where a static feature subset (with 200 features) is used without adapting to concept drift. This static feature subset is determined by OMVFS using only the first 3,000 instances in the data stream.

From Fig. 6, we can observe that the performance of static feature subset is quite close to OMVFS at the beginning. However, as new data come in and concept drift becomes more prominent, the performance of static feature subset decreases. On the other side, the performance of the proposed OMVFS will not decrease as more data come in. Instead, the performance increases as OMVFS combines the information in the new data with the aggregated information from previous data. We can also observe that in general, the more features we select, the better performance OMVFS would achieve.

V. Conclusions

In this paper, we present possibly the first attempt to solve the online unsupervised multi-view feature selection problem. The proposed method OMVFS integrates information from different views to learn a consensus clustering indicator matrix, and further embeds the feature selection into the graph regularized clustering algorithm. By using the buffering technique, OMVFS can reduce the computational and storage cost while taking advantage of the structure information. The buffering will also help OMVFS capture the concept drift in the data streams. Extensive experiments conducted on four datasets demonstrate the effectiveness and efficiency of the proposed OMVFS algorithm. Without sacrificing performance, the proposed OMVFS is about 100 times faster than the best off-line multi-view feature selection method.

Acknowledgment

This work is supported in part by NSF through grant IIS-1526499, NSFC through grants 61472089, 61503253, 61672357, and NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

[1] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.
[2] W. Shao, X. Shi, and P. Yu, "Clustering on multiple incomplete datasets via collective kernel learning," in *ICDM*, 2013.
[3] G. H. John, R. Kohavi, K. Pfleger *et al.*, "Irrelevant features and the subset selection problem," in *ICML*, 1994.
[4] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *ICML*, 2007.
[5] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *KDD*, 2010.
[6] S. Wang, J. Tang, and H. Liu, "Embedded unsupervised feature selection." in *AAAI*, 2015.
[7] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis." in *AAAI*, 2012.
[8] M. Qian and C. Zhai, "Robust unsupervised feature selection." in *IJCAI*, 2013.
[9] H. Wang, F. Nie, and H. Huang, "Multi-view clustering and feature learning via structured sparsity," in *ICML*, 2013.
[10] J. Tang, X. Hu, H. Gao, and H. Liu, "Unsupervised feature selection for multi-view data in social media." in *SDM*, 2013.
[11] M. Qian and C. Zhai, "Unsupervised feature selection for multi-view clustering on text-image web news data," in *CIKM*, 2014.
[12] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *KDD*, 2003.
[13] S. Wang, J. Tang, and H. Liu, "Embedded unsupervised feature selection," in *AAAI*, 2015.
[14] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001.
[15] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *TPAMI*, vol. 33, no. 8, pp. 1548–1560, 2011.
[16] L. Bottou, "Online learning and stochastic approximations," *On-line learning in neural networks*, vol. 17, no. 9, p. 142.
[17] W. Shao, L. He, C.-T. Lu, X. Wei, and P. S. Yu, "Online unsupervised multi-view feature selection," *CORR arXiv:1609.08286*, 2016.
[18] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *NIPS*, 2005.
[19] H. Huang, S. Yoo, and S. P. Kasiviswanathan, "Unsupervised feature selection on data streams," in *CIKM*, 2015.
[20] M. Qian and C. Zhai, "Unsupervised feature selection for multi-view clustering on text-image web news data," in *CIKM*, 2014.
[21] S. Bickel and T. Scheffer, "Multi-view clustering," in *ICDM*, 2004.