Incremental Multi-view Support Vector Machine

Peng Zhou* Yi-Dong Shen† Liang Du‡ Fan Ye§

Abstract

Multi-view classification has received considerable attention in recent years. We observed that the existing multi-view classification methods learn a consensus result by collecting all views and thus have two critical limitations. First, it is not scalable. Second, in many applications views of data are available over time; it is infeasible to apply the existing multi-view learning methods to such streaming views. To address the two limitations, in this paper we propose a novel incremental multi-view SVM method, i.e., instead of processing all views simultaneously, we integrate them one by one in an incremental way. We first learn an initial model from the first view; next when a new view is available, we update the model and then apply it to learn a new consensus result. This incremental method is scalable and applicable to streaming views. We present a block coordinate descent algorithm whose convergence is theoretically guaranteed to optimize the induced objective function. Experimental results on several benchmark data sets further demonstrate the effectiveness of our method.

1 Introduction

Many real-world data sets are represented in multiple views. For example, images on the web may have two views: visual information and textual tags; and multilingual data sets have multiple representations in different languages. In recent years, different multi-view learning methods, such as [18, 13, 25, 15, 10, 30], have been proposed to improve the classification accuracy by making full use of the coherence of different views.

In particular, the support vector machine (SVM)[2] is a kernelized classification methodology of machine learning by utilizing the labeled samples to train a model, and has been widely applied in machine learning fields. Due to its effectiveness on classification tasks, many researches have extended SVM to multi-view setting [28, 4, 13, 6, 10, 7, 14]. For example, Farquhar et al. and Li et al. presented SVM methods for two-view data [4, 13]; Gehler et al. proposed a boosting approach for multi-view classification [6].

We observed that these existing multi-view SVM methods learn a consensus result by collecting all views and thus have two critical limitations. First, it is not scalable; with limited computational resources it would be difficult to collect and process a large collection of views together. Second, in many applications views of data are available over time; it is infeasible to apply the existing methods to such streaming views. For example, in dangerous gas detection system [23], a number of sensors have been deployed to detect the chemical gas and sample the field data once in a while. In this application, there are a number of gaseous substances which need to be classified and each gas substance is continuously monitored by several sensors. The sampled data at each time interval constitute a view, so the number of views increases over time. Obviously, none of the above mentioned multi-view SVM methods are applicable to such streaming views, because there would be an endless number of views available and it is too expensive, if not impossible, to store all historical views in a repository and ensemble all of them.

To address the two limitations, in this paper we propose a novel *incremental multi-view SVM* (IMSVM) method, i.e., instead of ensembling the collection of all views simultaneously, we integrate them one by one in an incremental way. The basic idea is that, assuming we have already learned a model from previous views, when a new view is available, instead of redoing multi-view learning on all views, we only update the model with the current view and learn a consensus result by applying the updated model. This method is scalable and suitable for streaming views.

In IMSVM, we keep a model which represents a collection of previous views. This model consists of a consensus kernel together with some necessary SVM parameters. Initially, we construct a Gaussian kernel from the first view and use it to learn the SVM parameters; next, when a new view is available, we apply it to update the kernel and SVM parameters jointly, for the reason that, on one hand, given a kernel matrix, we aim to learn SVM parameters which can classify the instances correctly; on the other hand, given the SVM parameters, we wish to learn a kernel matrix which can fit the current classifier better. Different from traditional kernel learning methods which explicitly combine kernels and optimize structured (e.g. linear, nonnegative, convex) compositions of the kernels [11, 27], we learn a non-parametric kernel in such a way that we can effectively enlarge the re-

*School of Computer Science and Technology, Anhui University, Hefei 230601, China. Email:$zhoupeng@ahu.edu.cn$

†The State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China. Email:$ydshen@ios.ac.cn$, corresponding author.

‡School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China. Email:$csliangdu@gmail.com$

§School of Computer Science and Technology, Anhui University, Hefei 230601, China. Email:$yfan@ahu.edu.cn$, corresponding author.

gion from which an optimal kernel can be chosen for classification, i.e., we have a great chance to learn a better kernel.

However, an obvious problem with non-parametric kernel learning is that we may need to relearn the model to obtain the kernel values of testing data. To handle this out-of-sample problem, we enforce a reconstruction property on the learned kernel to preserve the manifold structure of the data in the mapping kernel space, so that the kernel values of new data can be computed efficiently. Thanks to this reconstruction property, we can also effectively handle new data and missing data in new views.

We present a block coordinate descent algorithm to solve the objective function of IMSVM, which is theoretically guaranteed to converge. To demonstrate the effectiveness of our IMSVM, we conduct extensive experiments on benchmark data sets and compare IMSVM with the state-of-the-art multi-view SVM methods. The experimental results show that our method significantly outperforms the baseline methods; not only is it scalable, it also has lower classification error rates than the state-of-the-art methods.

2 Related Work

In recent years, SVM [2] has been extended to multi-view settings many times. For example, Farquhar et al. proposed an SVM method for two-view data by combining the kernel canonical correlation analysis (KCCA) [8] and SVM [4]. Li et al. presented a two-view transductive SVM model which takes advantage of both the abundant amount of unlabeled data and their multiple representations to improve the performance [13].

Besides the above SVM methods for two-view data, many SVM methods have been extended to the data which contain more than two views. For example, Gehler et al. proposed a boosting approach for multi-view classification by ensembling the classification results in each view [6]; Sun et al. and Xie et al. presented multi-view SVMs for semi-supervised learning which integrated the manifold regularization into their SVM formulations [19, 24]; Huang et al. provided a multi-view L2- SVM method which utilized the core vector machine (CVM) [21] for the purpose of efficiency [10]. Houthuys et al. proposed a least squares SVM which combines the benefits from both early and late fusion of multiple views [9]. Tang et al. presented a multi-view SVM method using the privileged information [20].

Since SVM is a kernelized method, some multiple kernel learning methods can be used for multi-view data. Kloft et al. presented an SVM based multiple kernel learning method whose kernel combination coefficients $\boldsymbol{\mu}$ are optimized under the constraint $\|\boldsymbol{\mu}\|_p \leq 1$ with $p \geq 1$ [11]. Xu et al. presented a hinge loss soft margin multiple kernel SVM method whose kernel combination coefficients are optimized by solving a hinge loss [27].

All above multi-view SVM methods and SVM based multiple kernel learning methods learn a consensus result by applying an ensemble algorithm over the collection of all views. So they have two limitations: they are not scalable and hard to handle streaming views data.

3 Preliminaries about Single-view Multi-class SVM

In this section, we provide some background knowledge of single-view multi-class SVM. Throughout the paper, we use boldface uppercase and lowercase letters to denote matrices and vectors, respectively, unless otherwise specified. We use lowercase letters to denote the scalars. The (i, j)-th element of a matrix $\mathbf{A}$ is denoted by A_{ij} and the i-th column of a matrix $\mathbf{A}$ is denoted by $\mathbf{A_i}$.

Let $D = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)\}$ be a single-view data set of n training instances. Each instance $\mathbf{x_i}$ is a d-dimension vector from the domain $\mathcal{X} \subseteq \mathbb{R}^d$ and each label y_i is an integer from the set $\mathcal{Y} = \{1, 2, ..., c\}$ where c is the number of classes. A multi-class classifier is a function $C : \mathcal{X} \to \mathcal{Y}$ that maps an instance $\mathbf{x_i}$ to an element $y_i \in \mathcal{Y}$.

According to [3], one formulation of multi-class SVM is as follows:

$$\begin{aligned} \text{(3.1)} \quad \min_{\mathbf{A},\xi} \quad & \frac{\beta}{2}\|\mathbf{A}\|_F^2 + \sum_{i=1}^{n} \xi_i \\ s.t. \quad & \forall i \in \{1, ..., n\}, j \in \{1, ..., c\} : \\ & \mathbf{A_{y_i}}^T \mathbf{x_i} + \delta_{y_i,j} - \mathbf{A_j}^T \mathbf{x_i} \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{aligned}$$

where $\mathbf{A_j}$ is the j-th column of the $d \times c$ weight matrix $\mathbf{A}$, ξ_i is a slack variable, β is a balancing parameter, and $\delta_{i,j}$ is a Kronecker symbol, i.e., $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ otherwise.

In the training phase, we learn $\mathbf{A}$ by optimizing Eq.(3.1). In the testing phase, given a testing instance $\mathbf{x}$, we predict the label of $\mathbf{x}$ as follows:

$$\text{(3.2)} \qquad C(\mathbf{x}) = \underset{j=1}{\overset{c}{\operatorname{argmax}}}\, \mathbf{A_j}^T \mathbf{x}.$$

To handle the non-linear data, we need to apply kernel trick to this multi-class SVM (Eq.(3.1)). Introducing the dual variables $\boldsymbol{\tau} \in \mathbb{R}^{c \times n}$ and the kernel function $K(\cdot, \cdot)$ that satisfies Mercer's conditions, the dual program of Eq.(3.1) using kernel functions is:

$$\begin{aligned} \text{(3.3)} \quad \min_{\boldsymbol{\tau}} \quad & \frac{1}{2}\sum_{i,j=1}^{n} K_{ij} \boldsymbol{\tau_i}^T \boldsymbol{\tau_j} - \beta \sum_{i=1}^{n} \boldsymbol{\tau_i}^T \mathbf{1}_{y_i} \\ s.t. \quad & \forall i \in \{1, ..., n\} : \quad \boldsymbol{\tau_i} \leq \mathbf{1}_{y_i}, \quad \boldsymbol{\tau_i}^T \mathbf{1} = 0. \end{aligned}$$

where K_{ij} is the (i, j)-th element of the Gram Matrix $\mathbf{K}$ of the kernel function $K(\cdot, \cdot)$, $\boldsymbol{\tau_i}$ is the i-th column of $\boldsymbol{\tau}$, $\mathbf{1}_{y_i}$ is a vector whose components are all zeros except for the y_i-th component which is equal to one and $\mathbf{1}$ is a vector whose components are all one.

In the training phase, we learn the dual variables $\boldsymbol{\tau}$ by optimizing Eq.(3.3). Eq.(3.3) can be efficiently optimized by a sequential minimal optimization method, i.e., in each iteration we choose one column in $\boldsymbol{\tau}$ to be optimized. The detailed algorithm can be found in Algorithm 3 in [3].

After obtaining $\boldsymbol{\tau}$, we can use the following equation to predict the label of a testing instance $\mathbf{x}$:

$$C(\mathbf{x}) = \operatorname*{argmax}_{i=1}^{c} \sum_{j=1}^{n} \tau_{ij} K(\mathbf{x}, \mathbf{x_j}). \tag{3.4}$$

where τ_{ij} is the (i, j)-th element of matrix $\boldsymbol{\tau}$.

4 Incremental Multi-view SVM

In this section, we will introduce the formulation of our IMSVM and provide a block coordinate descent algorithm to solve the optimization problem.

The basic idea is that, we learn an initial model from the first view, and when a new view is available, we apply it to update the existing model, and learn a new consensus result from the updated model. Since our method extends the multi-class SVM introduced in the previous section into multi-view data, the model contains a consensus kernel matrix $\mathbf{K}$ and the SVM dual variables $\boldsymbol{\tau}$. Note that in the dual program of the original multi-class SVM (Eq.(3.3)), the kernel matrix $\mathbf{K}$ is given as an input and the target is only to learn the model parameter $\boldsymbol{\tau}$. However, in our multi-view setting, since we need to integrate the kernel matrices incrementally, the kernel matrix is also a parameter we need to learn. On one hand, given a kernel matrix, we aim to learn an SVM model $\boldsymbol{\tau}$ to classify the instances correctly; on the other hand, given the SVM model, we wish to learn a kernel matrix $\mathbf{K}$ which can fit the current classifier better. So in our method, we learn the kernel and the SVM model jointly. In the next subsection, we will introduce how to learn the consensus kernel and dual variables jointly.

4.1 Formulation When handling the first view ($t = 1$), we learn a standard single-view SVM classifier. In more details, we first generate the Gaussian kernel of this view and then use Eq.(3.3) to learn $\boldsymbol{\tau}$. In the following, we focus on the case $t > 1$.

When we handle the t-th view ($t > 1$), we have the learned consensus kernel $\mathbf{K}^{(t-1)}$ of the first $t-1$ views and the Gaussian kernel of the current view $\mathbf{K}_c$ on hand. The target is to learn the consensus kernel $\mathbf{K}^{(t)}$ of all the t views. Intuitively, we expect the model to be stable in the sense that the learned kernel $\mathbf{K}^{(t)}$ should be as close as possible to $\mathbf{K}^{(t-1)}$. This can be achieved by minimizing a smooth term $\left\|\mathbf{K}^{(t)} - \mathbf{K}^{(t-1)}\right\|_F^2$.

Moreover, since the learned kernel $\mathbf{K}^{(t)}$ integrates the current kernel $\mathbf{K}_c$ into $\mathbf{K}^{(t-1)}$, $\mathbf{K}^{(t)}$ is expected to be as close as possible to the current kernel $\mathbf{K}_c$, i.e., the difference $\left\|\mathbf{K}^{(t)} - \mathbf{K}_c\right\|_F^2$ should be minimized. Thus we get the following kernel integration formula:

$$\begin{aligned} \min_{\mathbf{K}^{(t)}} \quad & \|\mathbf{K}^{(t)} - \mathbf{K}^{(t-1)}\|_F^2 + \lambda_2 \|\mathbf{K}^{(t)} - \mathbf{K}_c\|_F^2 \\ s.t. \quad & \mathbf{K}^{(t)} \succeq 0, \quad \mathbf{K}^{(t)} = \mathbf{K}^{(t)T}. \end{aligned} \tag{4.5}$$

where the constraints guarantee that $\mathbf{K}^{(t)}$ is a valid kernel matrix, i.e., it is symmetric and positive semi-definite.

Combining Eq.(3.3) and Eq.(4.5), we obtain a unified formula which learns the SVM parameters and the consensus kernel jointly. However, in this formula, it is hard to handle the out-of-sample problem. More specifically, when we obtain a new testing instance $\mathbf{x_{test}}$, we need the kernel values $K^{(t)}(\mathbf{x_{test}}, \mathbf{x_1}), ..., K^{(t)}(\mathbf{x_{test}}, \mathbf{x_n})$, where $K^{(t)}(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function whose Gram matrix is the learned $\mathbf{K}^{(t)}$. So we should relearn $\mathbf{K}^{(t)}$ with $\mathbf{x_{test}}$. Since Eq.(4.5) contains positive semi-definite constraint, the relearning process may involve eigenvalue decomposition. Obviously, it is unpractical.

To tackle this problem, we impose a reconstruction property on the learned kernel $\mathbf{K}^{(t)}$ to make sure that the manifold structure of the data can be preserved in the mapping kernel space, so that we can learn the kernel values of new testing instances efficiently.

In more details, according to the definition of the kernel function, for any instances $\mathbf{x_i}$ and $\mathbf{x_j}$, $K^{(t)}(\mathbf{x_i}, \mathbf{x_j}) = \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle$ where $\phi : \mathcal{X} \to \mathcal{H}$ is a map function and $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS), and $\langle \cdot, \cdot \rangle$ is the inner product. For an instance $\mathbf{x_i}$, we choose a k-element subset $\{\mathbf{x_{i_1}}, \mathbf{x_{i_2}}, ..., \mathbf{x_{i_k}}\}$ from the training set as landmarks. Then we use these k landmarks to reconstruct $\mathbf{x_i}$, i.e., we learn the linear coefficient $\mathbf{w_i}$ which satisfies $\mathbf{x_i} \approx \sum_{j=1}^{k} w_{ij} \mathbf{x_{i_j}}$ where w_{ij} is the j-th element of $\mathbf{w_i}$. Here, we choose the landmarks and learn $\mathbf{w_i}$ by a similar way in Locally Linear Embedding (LLE) [17].

In more details, for an instance $\mathbf{x_i}$, we choose its k-NN points $\{\mathbf{x_{i_1}}, ..., \mathbf{x_{i_k}}\}$ as its landmark points. Here we follow the assumption of LLE [17]: for neighboring points, Euclidean distance provides a good approximation to real distance. Therefore, finding k-NN points by Euclidean distance is feasible. Then we learn $\mathbf{w_i}$ as follows:

$$\min_{\mathbf{w_i}} \left\| \mathbf{x_i} - \sum_{j=1}^{k} w_{ij} \mathbf{x_{i_j}} \right\|_2^2 + \frac{\alpha}{2} \|\mathbf{w_i}\|_2^2, \quad s.t. \sum_{j=1}^{k} w_{ij} = 1. \tag{4.6}$$

The first term is to minimize the linear reconstruction error, the second term is an ℓ_2 regularization term and the constraint is to normalize w_{ij} so that they sum up to 1. Taking the constraint into the objective function, we obtain

$$\left\| \mathbf{x_i} - \sum_{j=1}^{k} w_{ij} \mathbf{x_{i_j}} \right\|_2^2 + \frac{\alpha}{2} \|\mathbf{w_i}\|_2^2 = \sum_{i=1}^{k} \sum_{l=1}^{k} Q_{jl} w_{ij} w_{il} + \frac{\alpha}{2} \|\mathbf{w_i}\|_2^2$$

where $Q_{jl} = (\mathbf{x_i} - \mathbf{x_{i_j}})^T(\mathbf{x_i} - \mathbf{x_{i_l}})$. Let us define a $k \times k$ matrix $\mathbf{Q}$ whose (j,l)-th element is Q_{jl} and define a $k \times k$ matrix $\mathbf{R}$ as $\mathbf{R} = (\mathbf{Q} + \alpha\mathbf{I})^{-1}$ where $\mathbf{I}$ is an identity matrix. It is easy to obtain the optima of Eq.(4.6):

$$w_{ij} = \frac{\sum_{l=1}^{k} R_{jl}}{\sum_{p=1}^{k}\sum_{q=1}^{k} R_{pq}}. \tag{4.7}$$

where R_{pq} is the (p,q)-th element in $\mathbf{R}$. Here, for simplicity, we empirically set $\alpha = 10^{-3} \times tr(\mathbf{Q})$.

Note that although we involve matrix inverse when constructing $\mathbf{R}$, since the size of $\mathbf{R}$ is $k \times k$ and k is often very small in practice (we fix $k = 10$ in our experiments), the time complexity is $O(k^3)$, which is acceptable.

After obtaining the reconstruction coefficients of all data points, to preserve the manifold structure in the mapping space, the mapped values should also follow the reconstruction coefficients, i.e., $\phi(\mathbf{x_i}) \approx \sum_{j=1}^{k} w_{ij}\phi(\mathbf{x_{i_j}})$. Based on this condition, we can easily compute the kernel value of any two instances as following.

According to Eq.(4.7), we construct an $n \times n$ sparse matrix $\mathbf{W}^{(t)}$ for the t-th view, whose (i,j)-th element $W_{ij}^{(t)}$ is computed as Eq.(4.7) if $\mathbf{x_j}$ is a landmark of $\mathbf{x_i}$ in the t-th view, and is 0 otherwise. Then for any $K_{pq}^{(t)}$, we have

$$\begin{aligned} K_{pq}^{(t)} &= K^{(t)}(\mathbf{x_p}, \mathbf{x_q}) = \langle \phi(\mathbf{x_p}), \phi(\mathbf{x_q}) \rangle \\ &\approx \left\langle \sum_{i=1}^{n} W_{pi}^{(t)}\phi(\mathbf{x_i}), \sum_{j=1}^{n} W_{qj}^{(t)}\phi(\mathbf{x_j}) \right\rangle \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} W_{pi}^{(t)}W_{qj}^{(t)} \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} W_{pi}^{(t)}W_{qj}^{(t)} K_{ij}^{(t)} \end{aligned} \tag{4.8}$$

If $\mathbf{x_p}$ and $\mathbf{x_q}$ are not in the training set, we can also compute $K^{(t)}(\mathbf{x_p}, \mathbf{x_q})$ in a similar way. We only need to find the k-NN points of $\mathbf{x_p}$ and $\mathbf{x_q}$ in the training set, compute $W_{pi}^{(t)}$ and $W_{qj}^{(t)}$ by Eq.(4.7) and then compute $K^{(t)}(\mathbf{x_p}, \mathbf{x_q})$ by Eq.(4.8).

Thus to make the learned $\mathbf{K}^{(t)}$ have this reconstruction property, we minimize the following term based on Eq.(4.8):

$$\begin{aligned} \min_{K^{(t)}} \quad & \sum_{p=1}^{n}\sum_{q=1}^{n} \left(K_{pq}^{(t)} - \sum_{i=1}^{n}\sum_{j=1}^{n} W_{pi}^{(t)}W_{qj}^{(t)}K_{ij}^{(t)} \right)^2 \\ s.t. \quad & \mathbf{K}^{(t)} \succeq 0, \quad \mathbf{K}^{(t)} = \mathbf{K}^{(t)T}. \end{aligned} \tag{4.9}$$

which leads to the following formulation:

$$\begin{aligned} \min_{K^{(t)}} \quad & \left\| \mathbf{K}^{(t)} - \mathbf{W}^{(t)}\mathbf{K}^{(t)}\mathbf{W}^{(t)T} \right\|_F^2 \\ s.t. \quad & \mathbf{K}^{(t)} \succeq 0, \quad \mathbf{K}^{(t)} = \mathbf{K}^{(t)T}. \end{aligned} \tag{4.10}$$

Combining Eq.(3.3), Eq.(4.5) and Eq.(4.10), we get the final objective function of our Incremental Multi-view SVM:

$$\begin{aligned} \min_{\mathbf{K}^{(t)},\boldsymbol{\tau}} \quad & \lambda_1 \left(\frac{1}{2}\sum_{i,j=1}^{n} K_{ij}^{(t)} \boldsymbol{\tau_i}^T\boldsymbol{\tau_j} - \beta\sum_{i=1}^{n} \boldsymbol{\tau_i}^T\mathbf{1}_{y_i} \right) \\ & + \left(\left\| \mathbf{K}^{(t)} - \mathbf{K}^{(t-1)} \right\|_F^2 + \lambda_2 \left\| \mathbf{K}^{(t)} - \mathbf{K}_c \right\|_F^2 \right) \\ & + \lambda_3 \left\| \mathbf{K}^{(t)} - \mathbf{W}^{(t)}\mathbf{K}^{(t)}\mathbf{W}^{(t)T} \right\|_F^2 \\ s.t. \quad & \forall i \in \{1,...,n\}: \quad \boldsymbol{\tau_i} \leq \mathbf{1}_{y_i}, \quad \boldsymbol{\tau_i}^T\mathbf{1} = 0, \\ & \mathbf{K}^{(t)} \succeq 0, \quad \mathbf{K}^{(t)} = \mathbf{K}^{(t)T}. \end{aligned} \tag{4.11}$$

where λ_1 and λ_3 are balancing parameters.

Note that in Eq.(4.11), instead of explicitly combining kernels and optimizing structured compositions of the kernels as traditional methods [11, 27], which may limit their capacity of fitting diverse patterns in real complex applications, we learn a non-parametric kernel which aims to learn a positive semi-definite kernel matrix directly from the data such that the learned kernel can be as flexible as possible to fit the complex data. In other words, we can effectively enlarge the region from which an optimal kernel can be chosen, and learn a more suitable kernel for classification.

4.2 Optimization Since Eq.(4.11) contains variables $\mathbf{K}^{(t)}$ and $\boldsymbol{\tau}$, we present a block coordinate descent scheme to optimize it. In particular, we optimize the objective function with respect to one variable while fixing the other variables.

4.2.1 Optimize $\boldsymbol{\tau}$ by Fixing $\mathbf{K}^{(t)}$ When $\mathbf{K}^{(t)}$ is fixed, Eq.(4.11) degenerates into Eq.(3.3) which is a standard single-view multi-class SVM. It can be solved efficiently by Algorithm 3 in [3] and it is easy to verify that the time complexity of this algorithm is $O(ncs)$ where n is the number of instances, c is the number of classes and s is the number of iterations.

4.2.2 Optimize $\mathbf{K}^{(t)}$ by Fixing $\boldsymbol{\tau}$ When $\boldsymbol{\tau}$ is fixed, we rewrite Eq.(4.11) as following,

$$\begin{aligned} \min_{\mathbf{K}^{(t)}} \quad & \left\| \mathbf{K}^{(t)} - \mathbf{K}^{(t-1)} \right\|_F^2 + \lambda_2 \left\| \mathbf{K}^{(t)} - \mathbf{K}_c \right\|_F^2 \\ & + \lambda_3 \left\| \mathbf{K}^{(t)} - \mathbf{W}^{(t)}\mathbf{K}^{(t)}\mathbf{W}^{(t)T} \right\|_F^2 + \frac{\lambda_1}{2} tr(\mathbf{K}^{(t)}\boldsymbol{\tau}^T\boldsymbol{\tau}) \\ s.t. \quad & \mathbf{K}^{(t)} \succeq 0, \quad \mathbf{K}^{(t)} = \mathbf{K}^{(t)T}. \end{aligned} \tag{4.12}$$

Since Eq.(4.12) involves positive semi-definite constraint, it is hard to find the closed-form solution. To solve it efficiently, we make a low rank approximation on kernels. Moreover, according to [29], a low rank kernel often improves the learning performance. We assume that the rank of $\mathbf{K}^{(t)}$ is r ($r \ll n$), then we can rewrite $\mathbf{K}^{(t)} = \mathbf{H}^{(t)T}\mathbf{H}^{(t)}$ where $\mathbf{H}^{(t)} \in \mathbb{R}^{r\times n}$ since $\mathbf{K}^{(t)}$ is symmetric and positive semi-definite. For the current kernel $\mathbf{K}_c$, there are many methods to obtain its low rank approximation. For example, if n is small, we can directly compute the Singular Value Decomposition (SVD) of $\mathbf{K}_c$ and set all the singular values to 0 except for the largest r singular values. If n is large, in which case computing the SVD directly is very time-consuming, we can use random Fourier features method [16] to approximate the Gaussian kernel. In more details, we first sample $r/2$ vectors $\omega_1, ..., \omega_{r/2}$ from zero means and $1/\sigma$ variances Gaussian distribution, where σ is the bandwidth parameter in the original Gaussian kernel. Then we construct $\mathbf{H}_c \in \mathbf{R}^{r\times n}$ which satisfies $\mathbf{H}_c^T\mathbf{H}_c \approx \mathbf{K}_c$ as follows:

$$\mathbf{H}_c = [h(x_1)^T, ..., h(x_n)^T] \tag{4.13}$$

$$\text{where} \quad h(x) = \frac{\sqrt{2}}{\sqrt{r}}[\cos(\omega_1^T x), ..., \cos(\omega_{r/2}^T x), \sin(\omega_1^T x), ..., \sin(\omega_{r/2}^T x)] \tag{4.14}$$

Taking $\mathbf{H}^{(t)}$ and $\mathbf{H}_c$ into Eq.(4.12), we need to minimize the following formula:

(4.15)
$$\begin{aligned}\min_{\mathbf{H}^{(t)}} \mathcal{J} = &\left\|\mathbf{H}^{(t)T}\mathbf{H}^{(t)} - \mathbf{H}^{(t-1)T}\mathbf{H}^{(t-1)}\right\|_F^2 \\ &+\frac{\lambda_1}{2}\|tr(\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\boldsymbol{\tau}^T\boldsymbol{\tau}) + \lambda_2\left\|\mathbf{H}^{(t)T}\mathbf{H}^{(t)} - \mathbf{H}_c^T\mathbf{H}_c\right\|_F^2 \\ &+\lambda_3\left\|\mathbf{H}^{(t)T}\mathbf{H}^{(t)} - \mathbf{W}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\right\|_F^2\end{aligned}$$

Note that Eq.(4.15) is an unconstrained optimization problem and we can solve it by the standard Quasi-Newton method. To apply the Quasi-Newton method, we first compute the partial derivative of $\mathcal{J}$ w.r.t. $\mathbf{H}^{(t)}$:

(4.16)
$$\begin{aligned}\frac{\partial\mathcal{J}}{\partial\mathbf{H}^{(t)}} =&(4+4\lambda_2+4\lambda_3)\mathbf{H}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)} - 4\mathbf{H}^{(t)}\mathbf{H}^{(t-1)T}\mathbf{H}^{(t-1)} \\ &-4\lambda_2\mathbf{H}^{(t)}\mathbf{H}_c^T\mathbf{H}_c - 4\lambda_3\mathbf{H}^{(t)}\mathbf{W}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)T} \\ &-4\lambda_3\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)} + 2\lambda_1\mathbf{H}^{(t)}\boldsymbol{\tau}^T\boldsymbol{\tau} \\ &+4\lambda_3\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{W}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{W}^{(t)}\end{aligned}$$

Since we use low rank representation $\mathbf{H}^{(t)}$ instead of kernel matrix $\mathbf{K}^{(t)}$, we reduce the space complexity from $O(n^2)$ to $O(nr + nc + nk)$, where k is the k-NN number in $\mathbf{W}^{(t)}$ and $r, c, k \ll n$. Eq.(4.16) only involves matrix multiplications. The time complexity of computing $\mathbf{H}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}$, $\mathbf{H}^{(t)}\mathbf{H}^{(t-1)T}\mathbf{H}^{(t-1)}$, and $\mathbf{H}^{(t)}\mathbf{H}_c^T\mathbf{H}_c$ is all $O(nr^2)$; the time complexity of computing $\mathbf{H}^{(t)}\boldsymbol{\tau}^T\boldsymbol{\tau}$ is $O(nrc)$. Since $\mathbf{W}^{(t)}$ is a sparse matrix and each row of it contains $O(k)$ non-zero elements, the time complexity of computing $\mathbf{H}^{(t)}\mathbf{W}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)T}$, $\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)}$ and $\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{W}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)T}\mathbf{W}^{(t)}$ is all $O(nr^2 + nrk)$.

According to [3], optimizing $\boldsymbol{\tau}$ can make the objective function Eq.(3.3) decrease monotonically. In addition, Quasi-Newton method can also decrease Eq.(4.15). Therefore, whether optimizing $\boldsymbol{\tau}$ or $\mathbf{K}^{(t)}$ decreases the objective function Eq.(4.11) monotonically and the objective is lower bounded. So this block coordinate descent algorithm converges. Algorithm 1 summarizes the whole process.

Algorithm 1 Training phase of IMSVM

```
Input: A multi-view data set $D = \{(\mathcal{X}^{(1)}, \mathcal{Y}), ..., (\mathcal{X}^{(v)}, \mathcal{Y})\}$,
    parameters $\lambda_1$, $\lambda_2$, $\lambda_3$, $\beta$, and rank $r$ of $\mathbf{H}^{(t)}$.
Output: low rank approximation $\mathbf{H}^{(t)}$, SVM parameter $\boldsymbol{\tau}$
 1: Construct $\mathbf{H}^{(1)}$ of the Gaussian kernel of the first view.
 2: Optimize the initial $\boldsymbol{\tau}$ by Algorithm 3 in [3].
 3: for $t = 2, 3, \cdots, v$ do
 4:     Construct the low rank approximation $\mathbf{H}_c$ of the Gaussian
        kernel of the $t$-th view.
 5:     Construct $\mathbf{W}^{(t)}$ by Eq.(4.7).
 6:     while not converge do
 7:         Optimize $\boldsymbol{\tau}$ by Algorithm 3 in [3].
 8:         Optimize $\mathbf{H}^{(t)}$ by Quasi-Newton method.
 9:     end while
10: end for
```

4.3 Testing Phase

Algorithm 1 shows the process of training phase, and in this subsection, we present how to classify a data point in the testing set. When we get a testing point $\mathbf{x_{test}}$ in the t-th view, we need the kernel values $\mathbf{k_{test}}^{(t)} = [K^{(t)}(\mathbf{x_{test}}, \mathbf{x_1}), ..., K^{(t)}(\mathbf{x_{test}}, \mathbf{x_n})]$. Since we impose the reconstruction property on the learned kernels, we can compute these kernel values efficiently. In more details, we first find the k-NN points[1] of $\mathbf{x_{test}}$ in the training set and learn the sparse reconstruction weights $\mathbf{W_{test}}^{(t)} \in \mathbb{R}^{1\times n}$ by Eq.(4.7). Then we take $\mathbf{W_{test}}^{(t)}$ into Eq.(4.11) and obtain:

(4.17)
$$\begin{aligned}\min_{\mathbf{k_{test}}^{(t)}} \quad &\left\|\mathbf{k_{test}}^{(t)} - \mathbf{k_{test}}^{(t-1)}\right\|_2^2 + \lambda_2\left\|\mathbf{k_{test}}^{(t)} - \mathbf{k_{c,test}}\right\|_2^2 \\ &+\lambda_3\left\|\mathbf{k_{test}}^{(t)} - \mathbf{W_{test}}^{(t)}\mathbf{K}^{(t)}\mathbf{W}^{(t)}\right\|_2^2\end{aligned}$$

where $\mathbf{k_{c,test}} = [K_c(\mathbf{x_{test}}, \mathbf{x_1}), ..., K_c(\mathbf{x_{test}}, \mathbf{x_n})]$ is the kernel values of the current view. Since $\mathbf{x_{test}}$ is not in the training set, we omit the SVM loss of Eq.(4.11) here. If the testing data $\mathbf{x_{test}}$ is new in this view, i.e., it did not appear in previous views, we can easily omit the first term $\left\|\mathbf{k_{test}}^{(t)} - \mathbf{k_{test}}^{(t-1)}\right\|_2^2$. However, in traditional multi-

[1]If the number of instances or dimensions is too large, so that it is difficult to find the exact k-NN points, we can use some fast nearest neighbor algorithm to obtain the approximate k-NN points, e.g. [1].

view SVM methods which collect all views to learn a classifier, it is hard to handle this missing data case.

Eq.(4.17) has a closed-form solution:

(4.18)
$$\mathbf{k_{test}}^{(t)} = \frac{1}{1+\lambda_2+\lambda_3}\left(\mathbf{k_{test}}^{(t-1)} + \lambda_2 \mathbf{k_{c,test}} + \lambda_3 \mathbf{W_{test}}^{(t)} \mathbf{K}^{(t)} \mathbf{W}^{(t)}\right)$$

After obtaining $\mathbf{k_{test}}^{(t)}$, take it into our classifier:

$$C(\mathbf{x_{test}}) = \underset{i=1}{\overset{c}{\arg\max}} \sum_{j=1}^{n} \tau_{ij} K^{(t)}(\mathbf{x_{test}}, \mathbf{x_j}) \tag{4.19}$$

Since the number of support vectors l is often very small, $\boldsymbol{\tau}$ is sparse and we just need to compute $K^{(t)}(\mathbf{x_{test}}, \mathbf{x_j})$ where $\mathbf{x_j}$ is a support vector. So the time complexity of Eq.(4.19) is $O(lc)$. Note that in Eq.(4.18), $\mathbf{W_{test}}^{(t)}\mathbf{K}^{(t)}\mathbf{W}^{(t)} = \mathbf{W_{test}}^{(t)}\mathbf{H}^{(t)T}\mathbf{H}^{(t)}\mathbf{W}^{(t)}$, in which $\mathbf{H}^{(t)}\mathbf{W}^{(t)}$ which costs $O(nkr)$ time can be calculated beforehand and only calculated once. $\mathbf{W_{test}}^{(t)}\mathbf{H}^{(t)T}$ costs $O(kr)$ time and multiplying $\mathbf{W_{test}}^{(t)}\mathbf{H}^{(t)T}$ by $\mathbf{H}^{(t)}\mathbf{W}$ costs $O(rl)$ time since we just need to use the support vectors in Eq.(4.19). Thus although we learn a non-parametric kernel in our method, the testing time is comparable with the standard SVM.

4.4 Handling New Data and Missing Data

In the streaming view setting, it often happens that there are a small quantity of new data or missing data in the new view. In traditional multi-view SVM methods which collect all views for classification, it is hard to handle this incomplete data setting. Fortunately, since our method learns $\mathbf{H}^{(t)}$ which can be regarded as an embedding of instances in the kernel space and imposes the reconstruction property on kernels, this problem can be solved naturally.

The essential problem is that when optimizing Eq.(4.15), $\mathbf{H}^{(t-1)}$ and $\mathbf{H}_c$ are not aligned, i.e., some instances appear in $\mathbf{H}^{(t-1)}$ while are absence in $\mathbf{H}_c$ and vice versa. A natural way to solve it is filling $\mathbf{H}^{(t-1)}$ and $\mathbf{H}_c$ before optimizing Eq.(4.15). Since we enforce the reconstruction property on kernels, the filling step is easy. For an instance $\mathbf{x}$ which is in the $\mathbf{H}^{(t-1)}$ while is absence in $\mathbf{H}_c$, we denote $\mathbf{h_x}^{(t-1)}$ as the corresponding column of $\mathbf{x}$ in $\mathbf{H}^{(t-1)}$. We first find its k-NN instances in the common instances (appearing in $\mathbf{H}^{(t-1)}$ and $\mathbf{H}_c$ both), and learn the linear coefficient by Eq.(4.7) such that $\mathbf{h_x}^{(t-1)} \approx \Sigma_{j=1}^{k} w_j \mathbf{h_j}^{(t-1)}$ where $\mathbf{h_j}^{(t-1)}$ is a k-NN instance of $\mathbf{x}$ represented in $\mathbf{H}^{(t-1)}$ and w_j is the linear coefficient. Then in $\mathbf{H}_c$, we reconstruct the corresponding instance by $\mathbf{h_x}^c = \Sigma_{j=1}^{k} w_j \mathbf{h_j}^c$ where $\mathbf{h_x}^c$ is the estimated $\mathbf{x}$ in $\mathbf{H}_c$ and $\mathbf{h_j}^c$ is the k-NN instance of $\mathbf{x}$ represented in $\mathbf{H}_c$. For the instances that appear in $\mathbf{H}_c$ while absence in $\mathbf{H}^{(t-1)}$, we can handle them similarly.

5 Experience

In this section, we empirically evaluate the effectiveness of the proposed IMSVM on benchmark multi-view data sets.

Table 1: Description of the data sets.

	#instances	#features	#classes
UCI Digit	2000	216,76,64,6,240,47	10
Corel	3400	64, 9, 128, 10, 8, 104, 15	34
AwA	30475	4096, 2688, 2000, 252, 2000, 2000, 2000, 40960	50
Gas Sensor	17922	72 features × 100 views	11

5.1 Data Sets

We use benchmark multi-view data sets to evaluate the effectiveness of our method, including UCI Digit data set[2] [22], Corel data set[3][5], and Animal with Attributes (AwA) data set[4][12]. Moreover, to evaluate the effectiveness of our method on streaming view date set, we conduct the experiences on Gas sensor data set [5] [23]. This data set contains time-series measurement recordings collected from an array of 72 metal-oxide gas sensors utilized in the detection of dangerous chemical gaseous substances. Since the data set is a time-series recording, we sample 100 views from 100 time points, and in each view the recordings of the 72 sensors are the features. By this way, we obtain a 100-view data set. The important statistics of these data sets are summarized in Table 1.

5.2 Baseline Methods

To evaluate the effectiveness of our method, we compare it with the following methods:

- **Single-view SVM** [3], which is a standard single-view multi-class SVM introduced in Section 3.
- **Lp-boost SVM** [6], which is a multi-view SVM using boosting approach to mix multiple views.
- **LpMKL** [11], which is a multiple kernel SVM method whose kernel combination coefficients $\boldsymbol{\mu}$ are optimized under the constraint $\|\boldsymbol{\mu}\|_p \leq 1$ with $p \geq 1$.
- **SMMKL** [27], which is a hinge loss soft margin multiple kernel SVM method whose kernel combination coefficients are optimized by solving a hinge loss.
- **MvCVM** [10], which is a multi-view SVM utilizing CVM method [21].

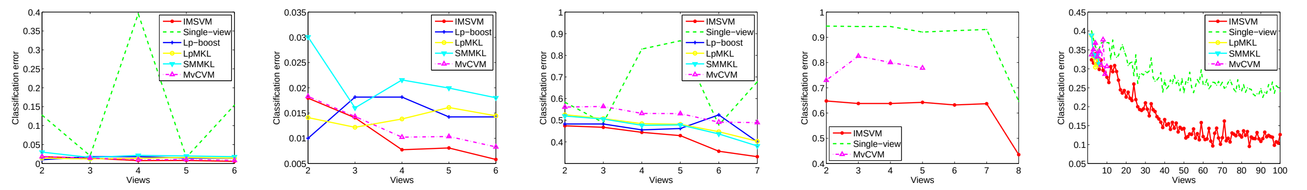
Note that although LpMKL and SMMKL are multiple kernel learning methods, they use SVM loss function, so we regard them as SVM based methods and compare with them. All methods, including our method and all compared methods, use multiple Gaussian kernels as input data. Single-view

[2]`http://archive.ics.uci.edu/ml/datasets/Multiple+Features`

[3]`http://www.cs.virginia.edu/%7exj3a/research/CBIR/Download.htm`

[4]`http://attributes.kyb.tuebingen.mpg.de/`

[5]`http://archive.ics.uci.edu/ml/datasets/Gas+sensor+arrays+in+open+sampling+settings`



(a) Error rates on UCI Digit (b) Error rates on UCI Digit (without Single-view SVM) (c) Error rates on Corel (d) Error rates on AwA (e) Error rates on Gas Sensor

Figure 1: Error rates on multi-view data sets.

SVM is a weak baseline as it only uses the current view; the other four methods are strong baselines as they make use of all available views. Since our method is in an incremental scheme, it only uses the current view to update the model and learn a new result with the updated model.

5.3 Experimental Setup In our experiments, on all data sets, we start incremental multi-view learning with the second view. When the t-th view ($t \geq 2$) is available, we run our method and the baseline methods and report the classification error rates on the testing set as [26] did. We randomly select $3/4$ instances as the training set and the rest $1/4$ instances as the testing set. We repeat this process 5 times and report the average results.

We fix the rank of kernel matrix $r = 100$ and the k-NN parameter $k = 10$ on all data sets. We fix $\lambda_3 = 0.1$, $\beta = 0.1$ and tune λ_1 and λ_2 by 5-fold cross validation in the range $[10^{-3}, 10^3]$. We tune the parameters of the baseline methods as suggested in their papers.

All experiments are conducted using Matlab on a PC computer with Windows 7, 3.4GHz CPU and 32GB memory.

5.4 Experimental Results Figure 1 shows the classification error rates of our method and the baseline methods on the benchmark data sets. We see that the error rates of our method keep decreasing with the increasing of the number of views on all data sets. This indicates that using a new view to update the current model can indeed improve the performance of the model.

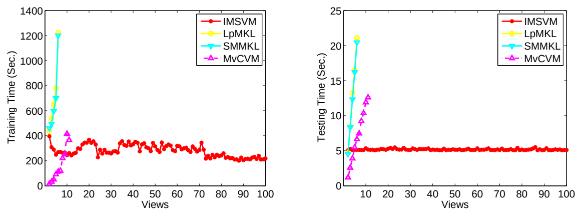
The classification results also clearly demonstrate that our method significantly outperforms the single-view SVM. Note that on the UCI Digit data set (Figure 1(a)), since the performance of single-view SVM on the fourth view is too bad, it is hard to distinguish the results of our method and the other baseline methods. We redisplay the results of our method and the baseline methods except the single-view SVM on Figure 1(b). We can also see that, on most data sets, our method can even outperform the state-of-the-art multi-view SVM methods which collect all views for classification. The major reason is that our method learns a non-parametric kernel in which way we enlarge the region from which an optimal kernel can be chosen, and have a great chance to learn a more suitable kernel.

Note that on AwA data set, we only show the results of our method, Single-view SVM and MvCVM because Lp-boost SVM, LpMKL, and SMMKL run out of memory. These methods need to handle kernel matrices which need $O(n^2)$ space and will fail when n is large. MvCVM can only handle 5 views on this data set, and run out of memory when the 6-th view is available. Although MvCVM uses sampling technique to handle the case that n is large, the space complexity is square in the number of views v. On Gas Sensor data set, Lp-boost SVM cannot run a result in an acceptable time since it needs to solve a quadratic programming which contains $O(n+v)$ variables and $O(cn+v)$ constraints. Moreover, with the increasing of the views, our method always works. However, LpMKL, SMMKL and MvCVM can only work on the first 6, 6, and 11 views respectively. When one more view comes, they run out of memory. It demonstrates that those methods which collect all views to learn a consensus result are not scalable. Since our method is in an incremental scheme, the memory it uses is independent of the number of views, and it can always work no matter how many views there are.

5.5 Running Time on Streaming Views Figure 2 shows the running time on the streaming views data set Gas Sensor. In this setting, the views are available one by one and we show the training and testing time of our method and the other baseline methods when a new view arrives.

As introduced in previous subsection, LpMKL and S-MMKL can only work on the first 6 views and MvCVM can only work on the first 11 views. Since our method is an incremental method, i.e., when a new view arrives, we just need to handle the current view, the running time of our method is relatively stable with the number of views. Therefore, when the number of views is large, our method is significantly faster than LpMKL and SMMKL. Since MvCVM uses CVM to speed up the method, it is fast when the number of views is small. However, when the number of views increases, it also slows down significantly and when handling more than 10 views, it is slower than ours in training phase.

In the testing phase, the running time on the whole testing set of our method is stable in about 5 seconds, while the time of other methods increases with the increasing of the number of views. When the number of views is large, our method is also significantly faster than the baseline methods in the testing phase.



(a) Training time (b) Testing time (on all testing instances)

Figure 2: Running time on Gas Sensor data set.

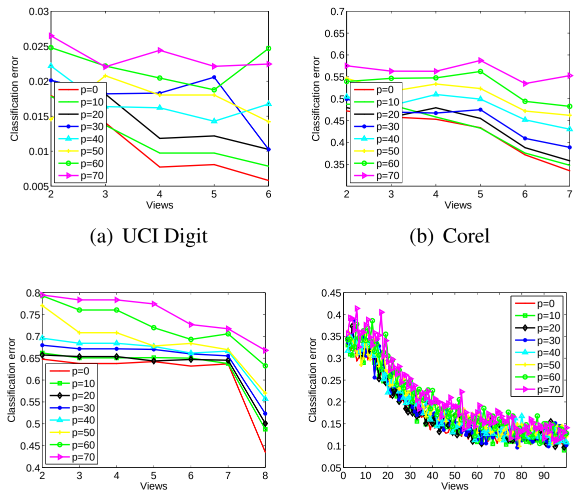
5.6 Evaluation on Sensitivity to the Order of Views An important property of our IMSVM is that it is not sensitive to the order of views. We empirically evaluate this property by randomly shuffling the views of each data set 5 times. Table 2 shows the results under 5 different order of views. It is easy to check that the 5 results in each row are similar and most results are better than the result of the second best method, meaning that our incremental method produces similar results under whatever order of views.

Table 2: Classification error rates of IMSVM under 5 different order of views.

Data sets	Different view orders					Second best
	1	2	3	4	5	
UCI Digit	0.0058	0.0081	0.0097	0.0081	0.0077	0.0083
Corel	0.3352	0.3320	0.3361	0.3379	0.3344	0.3811
AwA	0.4352	0.4422	0.4131	0.4665	0.4623	-
Gas Sensor	0.1263	0.1192	0.1556	0.1443	0.12437	-

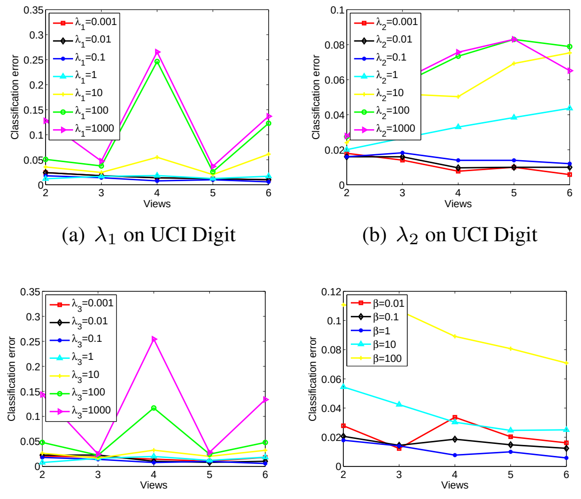
5.7 Handling New Data and Missing Data We evaluate the ability of our method to handle new data and missing data. Given a data set, in the first view, we randomly remove $p\%$ of data as missing data. Then, in following each view, we add $p/(v-1)\%$ of data as new data and simultaneously remove $p/(v-1)\%$ of data as missing data, where v is the number of views. Thus in each view, the ratio of missing data is $p\%$. We show the classification results in Figure 3.

In Figure 3, we show the results of missing data from 10% to 70%. Here $p = 0$ means that the data set is complete without missing data. As expected, the performance is degraded with the increasing of missing data. Despite all this, the performance of our method on the data whose ratio of missing data is lower than 30% is close to it on the complete data. It demonstrates that our method can handle the case with a small quantity of missing and new data well.



(a) UCI Digit (b) Corel (c) AwA (d) Gas Sensor

Figure 3: Classification results in the incomplete data.



(a) λ_1 on UCI Digit (b) λ_2 on UCI Digit (c) λ_3 on UCI Digit (d) β on UCI Digit

Figure 4: Classification error w.r.t. λ_1, λ_2, λ_3, and β on UCI Digit data set.

5.8 Parameter Study We test different parameter settings for IMSVM to see the performance variation. We tune λ_1, λ_2 and λ_3 in $[10^{-3}, 10^3]$ and tune β in $[10^{-2}, 10^2]$. Figure 4 shows the results on the UCI Digit data set, and the results on other data sets are similar. From Figure 4, we see that the performance is stable across a wide range of the parameters.

6 Conclusion

We presented a novel incremental multi-view SVM method (IMSVM). In IMSVM, we keep a consensus kernel together with some SVM parameters as a model, and when a new view is available, we update the model and apply it to learn a consensus result. This essentially differs from the existing multi-view SVM methods and is scalable and ready to handle streaming views data. In order to obtain a more suitable kernel, we learn a non-parametric kernel instead of explicitly combining all kernels. Although learning non-parametric kernel leads to the out-of-sample problem, we handle it by enforcing a reconstruction property on the learned kernels. Experimental results shows its effectiveness; not only is it scalable, it also has better classification performance.

7 Acknowledgments

This work is supported by the National Natural Science Fund of China grants 61806003, and 61502289, the China National 973 program 2014CB340301, and the Key Natural Science Project of Anhui Provincial Education Department KJ2018A0010.

References

[1] Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *JMLR*, 10(Sep):1989–2012, 2009.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[3] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[4] Jason Farquhar, David Hardoon, Hongying Meng, John S Shawe-taylor, and Sandor Szedmak. Two view learning: Svm-2k, theory and practice. In *NIPS*, pages 355–362, 2005.

[5] James C French, James VS Watson, Xiangyu Jin, and WN Martin. Integrating multiple multi-channel cbir systems. In *Workshop on MIS*. Citeseer, 2003.

[6] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009.

[7] Zhenfeng Gu, Zhao Zhang, Jiabao Sun, and Bing Li. Robust image recognition by l1-norm twin-projection support vector machine. *Neurocomputing*, 223:1–11, 2017.

[8] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[9] Lynn Houthuys, Rocco Langone, and Johan A.K. Suykens. Multi-view least squares support vector machines classification. *Neurocomputing*, 282:78 – 88, 2018.

[10] Chengquan Huang, Fu-lai Chung, and Shitong Wang. Multi-view l2-svm and its multi-view core vector machine. *Neural Networks*, 75:110–125, 2016.

[11] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Lp-norm multiple kernel learning. *Journal of Machine Learning Research*, 12(Mar):953–997, 2011.

[12] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958. IEEE, 2009.

[13] Guangxia Li, Steven CH Hoi, and Kuiyu Chang. Two-view transductive support vector machines. In *SDM*, pages 235–244. SIAM, 2010.

[14] Vasileios Mygdalis, Anastasios Tefas, and Ioannis Pitas. Exploiting multiplex data relationships in support vector machines. *Pattern Recognition*, 85:70–77, 2019.

[15] Minh Ha Quang, Loris Bazzani, and Vittorio Murino. A unifying framework for vector-valued manifold regularization and multi-view learning. In *ICML*, pages 100–108, 2013.

[16] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[17] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[18] Vikas Sindhwani and David S Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *ICML*, pages 976–983, 2008.

[19] Shiliang Sun. Multi-view laplacian support vector machines. In *International Conference on Advanced Data Mining and Applications*, pages 209–222. Springer, 2011.

[20] Jingjing Tang, Yingjie Tian, Xiaohui Liu, Dewei Li, Jia Lv, and Gang Kou. Improved multi-view privileged support vector machine. *Neural Networks*, 106:96 – 109, 2018.

[21] IW-H Tsang, JT-Y Kwok, and Jacek M Zurada. Generalized core vector machines. *IEEE Transactions on Neural Networks*, 17(5):1126–1140, 2006.

[22] M Van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

[23] Alexander Vergara, Jordi Fonollosa, Jonas Mahiques, Marco Trincavelli, Nikolai Rulkov, and Ramón Huerta. On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines. *Sensors and Actuators B: Chemical*, 185:462–477, 2013.

[24] Xijiong Xie and Shiliang Sun. Multi-view laplacian twin support vector machines. *Applied Intelligence*, 41(4):1059–1068, 2014.

[25] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv: Learning*, 2013.

[26] Chang Xu, Dacheng Tao, and Chao Xu. Streaming view learning. *arXiv: Machine Learning*, 2016.

[27] Xinxing Xu, Ivor W Tsang, and Dong Xu. Soft margin multiple kernel learning. *IEEE TNNLS*, 24(5):749–761, 2013.

[28] Zhao Zhang and Tommy W S Chow. Maximum margin multisurface support tensor machines with application to image classification and segmentation. *Expert Systems With Applications*, 39(1):849–860, 2012.

[29] Peng Zhou, Liang Du, Lei Shi, Hanmo Wang, and Yi-Dong Shen. Recovery of corrupted multiple kernels for clustering. In *IJCAI*, pages 4105–4111, 2015.

[30] Yao Zhou and Jingrui He. A randomized approach for crowdsourcing in the presence of multiple views. In *ICDM*, pages 685–694, 2017.