Incremental-Decremental Methods for Multi-view Discriminant Analysis

Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Saroj Snehal Shivagunde

Under the supervision of

Dr. Vijaya Saradhi Vedula

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati - 781039 Assam India

March, 2022

Dedicated to

Everyone

Acknowledgements

I wish to thank Dr. Vijaya V. Saradhi for his help and motivation. His valuable guidance is the reason I am at this stage of my research journey. He has always been very kind and supportive of me, in fact, of all of his students. He never imposed his ideas on us and always encouraged us to discuss things, giving us the freedom we might have needed.

I would also like to thank my Doctoral Committee members- Prof. Prabin K. Bora, Dr. Arijit Sur, Dr. Rashmi Dutta Baruah, and Prof. Hemangee Kapoor for their support, insightful comments and questions that led to the betterment of my work.

Lastly, I express my deep gratitude to my family and friends, who always supported me through the rain or the shine on this path that I chose.

March 21, 2022 Saroj Snehal Shivagunde

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.
- I am fully aware that my thesis supervisor is not in a position to check for any possible instance of plagiarism within this submitted work.

March 21, 2022 Saroj Snehal Shivagunde

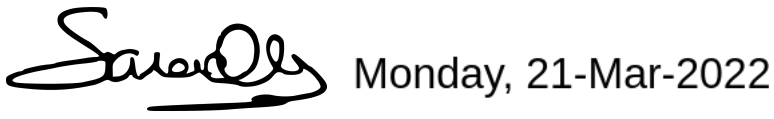
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Dr. Vijaya Saradhi Vedula
Associate Professor
Email : saradhi@iitg.ac.in
Phone : +91-361-258-2356

Certificate

This is to certify that this thesis entitled "**Incremental-Decremental Methods for Multi-view Discriminant Analysis**" submitted by **Saroj Snehal Shivagunde**, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by her under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: March 21, 2022
Place: Guwahati



Dr. Vijaya Saradhi Vedula
(Thesis Supervisor)

Abstract

In recent years, the use of multi-view data has attracted much attention from the machine learning community. Multiple views of an object are obtained using different sensors and contain different features that describe the object. Such complementary nature of the information contributed by multiple views leads to improved performance of the trained model. Many multi-view learning algorithms have been proposed to make use of multi-view data. However, these algorithms predominantly belong to the batch learning paradigm. Batch learning methods need all the training data at the start of the training process. These methods cannot incorporate new data once the training has been completed. If such a situation occurs, these methods must discard the trained model and retrain on the updated dataset, thereby proving expensive in terms of training time and memory.

Incremental methods solve this problem by updating the trained model after each increment without needing the historical data. Many incremental counterparts of single-view learning algorithms have been proposed in recent years. Some methods also support decremental unlearning of data when a subset of existing data is deleted. However, there are only a handful of incremental algorithms for multi-view data. The increment can be of two types in a multi-view context- data sample increment and view increment.

We present four multi-view methods in this thesis. Three of which are incremental methods equipped to update a trained model without needing the historical data, and one is a batch method. Two of the incremental methods are data sample incremental methods. One supports incremental learning for 1D multi-view data, and the other supports incremental learning and decremental unlearning for 2D multi-view data. The third method is a view incremental multi-view method that supports the addition and deletion of views. While formulating the 2D incremental method, we also formulated a 2D multi-view batch method due to the absence of a multi-view batch method for 2D data. All of these methods are based on Multi-view Discriminant Analysis (MvDA).

The first method is Multi-view Incremental Discriminant Analysis (MvIDA), which updates a trained model to incorporate new data samples. MvIDA requires only the old model and newly added data to update the model. Depending on the nature of increments, MvIDA is presented as two cases, sequential MvIDA and chunk MvIDA. Both of the cases are equipped to handle data samples from previously unseen classes. The experiments conducted on three widely used 1D multi-view datasets show that through order independence and faster construction of the optimal discriminant subspace, MvIDA addresses the issues faced by MvDA in an incremental setting.

The second method is 2D Multi-view Discriminant Analysis (2DMvDA), a 2D multi-view classification method based on discriminant analysis. It uses the 2D image matrices directly instead of extracting 1D features from them ensuring the preservation of spatial information and reduction in the sizes of scatter matrices. This leads to better classification accuracy and a considerable reduction in the computational cost. The experiments carried out on four image-based multi-view datasets show that, using less time and memory, 2DMvDA achieves a classification accuracy at par or better than its 1D and single-view counterparts.

We present an incremental version of 2DMvDA as 2D Multi-view Incremental Decremental Discriminant Analysis (2DMvIDDA) that provides a way to incorporate new data samples or discard the old ones without needing the historical data. The updates are done using just the old model

and the data samples to be added or deleted. We also present 2DMvIDDA as an umbrella method that can be transformed into other methods that are based on discriminant analysis, such as MvDA, MvIDA, 2DMvDA, 2DLDA (2D Linear Discriminant Analysis), and ILDA (Incremental Linear Discriminant Analysis). Through the experiments on four image-based multi-view datasets, we show that the proposed method is order-independent and converges to the same discriminant subspace as 2DMvDA and builds a better model than other relevant methods with less time and memory.

Lastly, we present View Incremental Decremental Multi-view Discriminant Analysis (VIDMvDA) that updates a learned model without retraining when new views are added or existing views are deleted. VIDMvDA is presented in two forms: incremental learning and decremental unlearning. It provides closed-form solutions to update the within-class and the between-class scatter, and it can also be used for 2D data by changing only one parameter. We prove that using significantly less training time and memory, VIDMvDA constructs a similar discriminant subspace and classification accuracy as its batch counterpart.

Contents

List of Figures

List of Algorithms

List of Tables

1

Introduction

We are currently living in the *information age*, owing to the technological advances that enabled us to gather and analyze a lot of data. This explosion of data gave rise to the digital companies that strive to provide their clients a digital alternative to almost every aspect of their lives. We enjoy the luxury of these digital options ranging from virtual meetings to online shopping and from digital banking to automated medical diagnosis and fitness trackers. Various devices that support these technologies keep generating a lot of raw data that can be turned into usable knowledge. The knowledge, thus obtained, is the key for digital companies to provide their clients with a better experience in virtual or real life. Hence, the need to transform this data into usable knowledge increases.

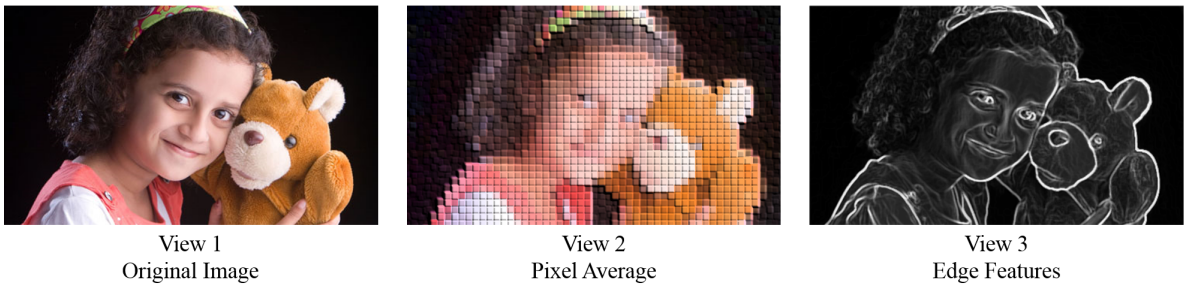
Analysis of such a vast amount of data calls for algorithms that produce results in a very short time using less memory and can improve themselves by including newer data and getting rid of obsolete data. This thesis aims to fulfill this need in part by introducing four multi-view methods in the supervised learning domain. These methods are based on discriminant analysis and produce the same or even better results with significantly less time and memory requirements than the traditional machine learning algorithms. Multi-view learning, 2D methods, and incremental learning paradigms form the basis of the proposed methods. In this chapter, we look at the features of these paradigms that provide motivation and help in designing the methods presented in this thesis.

Multi-view Learning

Data acquired by different sources related by a common latent aspect is known as multi-view data. A view of an object is defined as a set of features that store information about a certain aspect of that object. The data samples within a view are assumed to be independent and identically distributed (IID). Multiple views of an object provide information about multiple aspects of the same object. The views of data are generated- (i) by observing the object using a single type of sensor from multiple perspectives, or (ii) by using multiple sensors of different types, or (iii) by obtaining independent sets of features from existing views. Fig. 1.1 shows examples of each of these three cases. Fig. 1.1a presents an example of case-(i) of multi-view data where each view is captured by a similar instrument, here- a camera. Each view has the same number of dimensions. All the views complement each other by capturing the same person from different angles. Another example of case-(i) is [1], where a photograph of a person captured in the visual spectrum is one view, and a photograph of the same person captured in the infra-red spectrum is another view. Here, each

(a) Views obtained from a single sensor.

(b) Views obtained from different sensors.



(c) Views derived from an existing view.

Figure 1.1: Examples of multi-view data.

view is a photograph, so their features are comparable. Both the views complement each other by capturing the same person in a different spectrum. Multiple views in fig. 1.1b represent case-(ii) as they are obtained from different sensors belonging to different domains, namely- audio, image, and text. As these views have different representations and features, they are not directly comparable in their original form. A similar example of case-(ii) is [2], which has a multi-view dataset for speech recognition with speech signals as one view and patterns of lip motion as another view. These views cannot be compared without transforming them into features that share a common subspace. Fig. 1.1c shows an example of case-(iii). The first view is a photograph of a person captured using a color camera. View-2 and view-3 are obtained from the first view by performing pixel average and edge detection operations, respectively. These views are also not directly comparable in their original form as they have different representations.

Traditional machine learning used only one view of the underlying objects. The use of multiple views improves the performance of learned models because each view contributes some information about the underlying objects [3, 4]. Fig. 1.2 shows an example that highlights the advantage of

View1

	Feature-Shape	Label
	Heart	Peach
	Heart	Peach
	Heart	Apple
	Heart	Apple
	Heart	Cherry
	Heart	Cherry
	Round	Orange
	Round	Orange
	Round	Grapefruit
	Round	Grapefruit
	Round	Sweet-lime
	Round	Sweet-lime

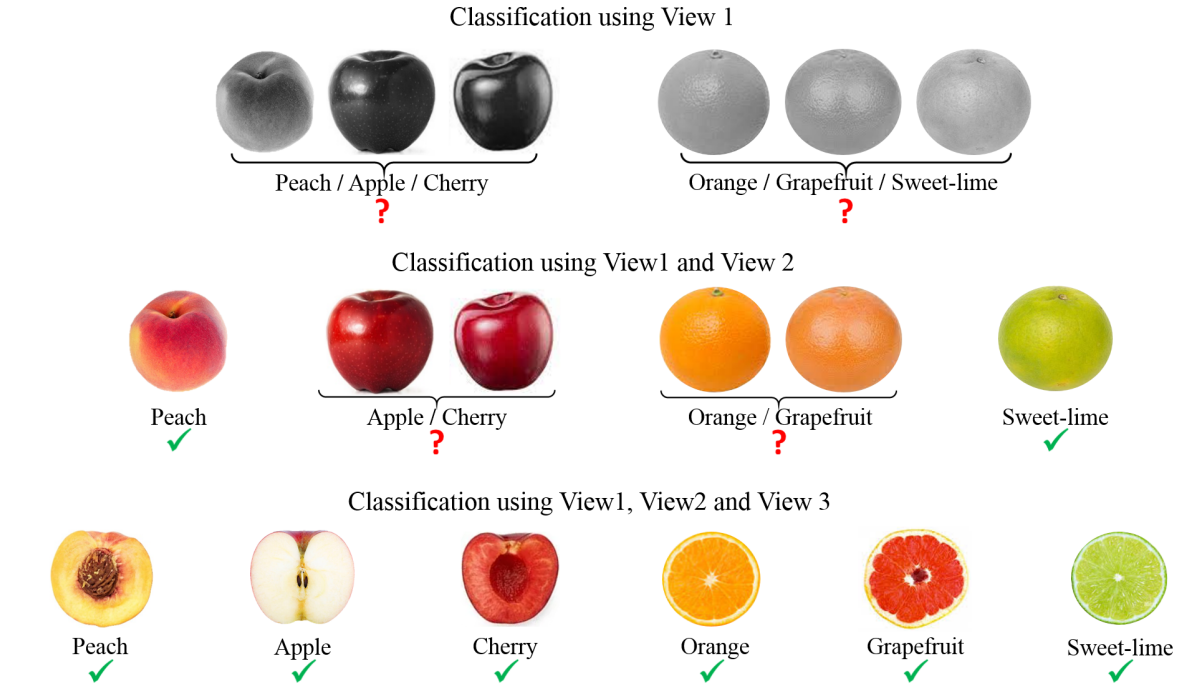
View2

	Feature-Color	Label
	Peach	Peach
	Peach	Peach
	Red	Apple
	Red	Apple
	Red	Cherry
	Red	Cherry
	Orange	Orange
	Orange	Orange
	Orange	Grapefruit
	Orange	Grapefruit
	Green	Sweet-lime
	Green	Sweet-lime

View3

	Feature-Core	Feature-Seeds	Label
	Yellow	Big	Peach
	Yellow	Big	Peach
	White	Small	Apple
	White	Small	Apple
	Red	Big	Cherry
	Red	Big	Cherry
	Orange	Small	Orange
	Orange	Small	Orange
	Red	Small	Grapefruit
	Red	Small	Grapefruit
	Green	Small	Sweet-lime
	Green	Small	Sweet-lime

(a) Training dataset that has three views



(b) Classification performance on test set with gradually increasing views

Figure 1.2: An example to demonstrate the advantages of having multiple views. As the number of views increases, each view adds new complementary information. The learning algorithm leverages this information to improve its performance on each test data sample.

using multiple views of data. Here, we see a training dataset with three views- the shape of each fruit is one view, second is the color of a fruit, and the third one is the cross-section of each fruit. Three things to note here are- (i) Each view contains the features of every fruit in the training dataset, (ii) all views agree on the label of each fruit, (iii) Each view may have a different number of features in it. These three are the fundamental assumptions of multi-view learning. Given this dataset of fruits, the task is to learn to classify fruits in the test dataset. We see in Fig. 1.2b, a model trained only on the first view has learned to distinguish between a heart-shaped and a round-shaped fruit, but it cannot distinguish between the fruits of the same shape as there are not enough features in that view that can teach the model to do so. We see three heart-shaped fruits and three round-shaped fruits in the test data. The information obtained from this view gives our model the ability to predict any label from ('peach', 'apple', 'cherry') for group-I and any label from ('orange', 'grapefruit', 'sweet-lime') for group-II. However, if we add a second view that describes the color of fruit and train a model on both the views, this model can see the facts that were not accessible using only the first view. Now, it can separate peach-colored 'peach' from group-I and green-colored 'sweet-lime' from group-II. However, it cannot distinguish between red-colored 'apple' and 'cherry', or orange-colored 'orange' and 'grapefruit'. When the third view that describes the pulp's color is added, each test sample can be labeled correctly because this view has more discriminatory information than the previous ones. This example demonstrates the benefit of using multiple views of the data that provide more object features that complement each other.

Due to their complementary nature, multiple views provide distinct information about the data, which is leveraged by the learning algorithms [3, 4]. Hence, multi-view learning has been gaining attention in the recent past. Several multi-view algorithms have been developed over the years in different machine learning paradigms, such as supervised [5, 6, 7, 8], semi-supervised [9, 10], active [11, 12], unsupervised [13, 14, 15] and transfer learning [16, 17]. These efforts demonstrate improvements in the performance of the algorithms due to the use of multiple views.

Multi-view data is used in various ways in different paradigms. Supervised methods use labeled data to train a model, and in multi-view learning, all views of an object in the training dataset agree with each other on its label. We can see the object-label pair as an input-output pair on which the model trains and, based on this training, produces output for the test objects. In the semi-supervised or active learning domain, labeled data is scarce, and the model has fewer data samples to train itself. Though the views agree on the labels of training data samples, due to the scarcity of ideal object-label pairs, they may produce different labels for test data samples. These conditions are often leveraged to expand the pool of labeled datasets by querying the labels of such data samples. Co-training[18] and methods based on it [19, 10] train separate models on different views and use these models alternately to label a few data samples that were labeled by other models. Then, the data samples whose labels were most agreed upon are included in the training set to improve the models further. The views of data can even be thought of as being *weak* or *strong*. In the above example (Fig. 1.2), we see that view-1 and view-2 are not capable of classifying all the fruits correctly on their own. Such views are termed as *weak views*. On the other hand, view-3 can perfectly classify all the data samples by itself. Such views are termed as *strong views*. Co-Testing [20] and methods based on it [21], make use of *weak* and *strong* views for learning a better model. These methods deems a data sample to be most informative if the views disagree on its label. It is based on the assumption that if the views disagree, then the data sample must be in the class-boundary region.

Though beneficial for learning, multi-view data poses two challenges- its size and incompatibility of the views. When we have multiple views, the cumulative number of features of a data sample increases, as each view contributes some information about the data. As a result, learning on multi-view data demands more time and storage. If the views are obtained from different sources,

they are incomparable as they belong to different feature spaces. The multi-view algorithms can be divided into three categories based on how they compare views. The first category of algorithms trains separate classifiers on each view while only considering the within-view information. This method is called late fusion, and some methods that use this technique are [22, 23]. At the time of classification, any form of polling may be used. It can be traditional polling that finalizes the label supported by the majority of the views, or it can be a weighted polling policy. A detailed study of weighted voting-based methods is given in [24]. Some methods use the strong views for voting, and the weak views are only used as tie-breakers. Algorithms in the second category consider within-view and between-view information at the time of training [25, 26, 14]. This technique is called early fusion, where information from all of the views is either simply concatenated or is integrated using some method, and only one classifier is trained using this information. In this case, the test data has to be transformed accordingly to suit the classifier. Some efforts to combine both fusion types can also be seen in the literature. These methods train weak classifiers on each view while using some parameters to gain information from other views without actually integrating them. Some of the methods that use combined fusion are [14, 27].

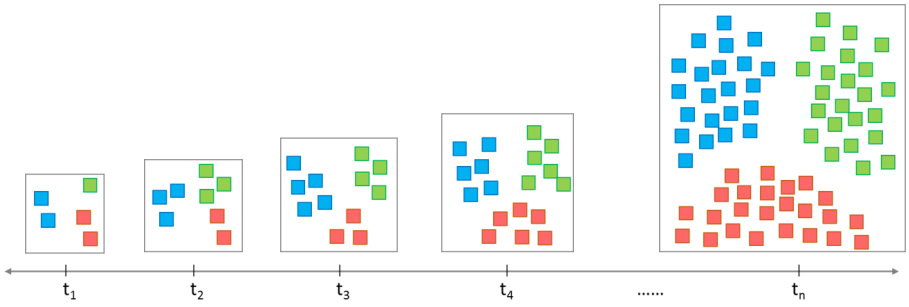
Some multi-view learning algorithms [5, 6] employ techniques such as discriminant analysis that can address both the challenges of multi-view data by constructing a common latent subspace from the data. Discriminant analysis is a supervised method that reduces dimensionality by removing redundant or dependent features of the data, thereby addressing the first challenge of size. The common subspace, which allows the comparison of information from different views, addresses the second challenge of incompatibility of views.

Multi-view learning algorithms proposed in the literature are predominantly batch learning algorithms. These algorithms require all of the data to be available at the start of the learning process. If additional data is introduced later, these algorithms must forget the trained model and re-learn all the historical data. As a result, learning time and memory demands grow with every addition in the data, making these algorithms unsuitable for such incremental data. To train efficiently on the incremental data, we need methods that can support the inclusion of new data without discarding the old model and retraining on the updated dataset. The incremental learning paradigm provides such support to learn from new data without retraining, which is even more desirable in the context of multi-view learning due to the large size of typical multi-view datasets.

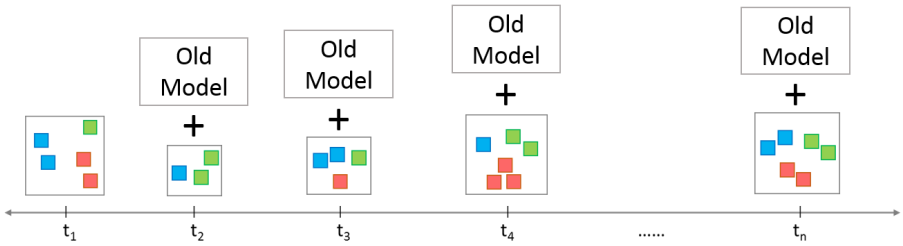
Incremental Learning

Data is said to be incremental when new data samples are added to the existing dataset gradually over time. To include these new data samples, batch methods train on all the previous data samples after every addition. Incremental learning helps us alleviate this problem by updating the model without training on all the previous data samples. With incremental learning, we can even add data samples that belong to a previously unseen class. Fig. 1.3 shows the difference between batch methods and incremental methods. As time passes, more and more data samples get added to the dataset and the batch methods have to train on all these data samples. Hence, the cost of training increases with time. On the other hand, incremental methods only need the old model and new data samples to update the model. Hence, the training of a model is faster and requires less memory.

The primary purpose of incremental learning is to include the latest data in the existing model without needing the old data or forgetting it. This way, the model retains all the valuable information from old data without needing those data samples for training in subsequent time-frames. Hence, incremental learning techniques are used where training data is obtained gradually over time. Another use case for incremental learning is when the memory requirements of data are



(a) Training with Batch Methods



(b) Training with Incremental Methods

Figure 1.3: Batch methods vs. Incremental methods: (a) Batch methods have to retrain on all the historical data after every addition. (b) Incremental methods only need the old model and new data samples to update the model after every addition.

beyond the limits of the computational system. In such a case, the training data can be divided into smaller subsets that the system can support and then the model is trained on these subsets incrementally.

The increment in single-view data may occur in two types-first adds one data sample at a time, termed as *sequential increment* and the other adds a group of data samples at a time, termed as *chunk increment*. The number of data samples added in one increment may or may not be the same as that in the previous increment. Incremental methods for single-view data include methods based on LDA [28, 29], PCA [30, 31] as well as SVM [32]. Some of the extensions [33, 34, 35, 36] of SVM are presented as incremental-decremental methods that allow addition as well as deletion of data samples. However, when used on multi-view data, the single-view methods treat all the views as one and lose the discriminating information each view offers. Hence, these methods are not equipped to handle multi-view data.

In the multi-view context, the increment can be of two types - the first is *data sample increment*, where the number of data samples gets updated with time, but the number of views remains unchanged. The new data samples are added to all the existing views simultaneously. We see the applications of this type of increment in tasks such as video surveillance [37] and autonomous driving [38] where new training samples are added from all the cameras or other sensors. The other type of increment is *view increment*, where the number of data samples is fixed, and the number of views is updated over time as new views of every existing data sample are added. This type of increment is seen when information from sensors, cameras, or other information capture tools that were previously not included is added to gather new views of data in the recognition or detection systems mentioned above.

The incremental learning paradigm is still in its early stages of development. Most of the

works report the advantages and the results of these methods. Due to the lack of comparative or analytical studies, the challenges faced by the incremental methods have not been studied extensively. Nevertheless, we find two major challenges listed in the literature- First is termed as *catastrophic forgetting* which refers to the situation where the learning algorithm forgets the older data while learning from the new samples. The other challenge is the *order of update*, where the order of addition or deletion of data samples changes the learned model considerably. Discriminant analysis-based methods do not face these problems as the scatter matrices are computed by adding the scatter of each data sample together, leading to the ability to remember all data samples and order-invariance. Another challenge is faced by the methods that retain some of the older data samples. These methods must devise a strategy to select the data samples that are most informative as there is often an imposition on the number of data samples a model can retain in their original form.

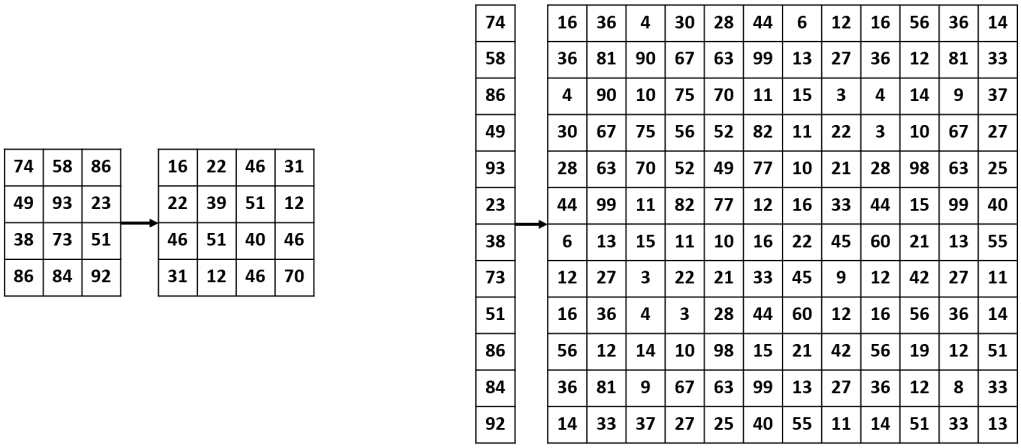
In real life, we see many applications that are of incremental nature and use newer data to produce up-to-date results. One such system to use incremental learning is an autonomous driving system [38] that collects data and incrementally learns to recognize different objects from every new encounter. Though many incremental learners keep expanding their knowledge-base by learning new information, others may choose to forget some of the older data samples for better performance. For example, the recommendation engine [39] of the online video platforms collects the watch history and recommends more videos related to the latest favorites of the user. Here, the system may choose to forget some or all of older data as it has become obsolete due to the changed preferences of the user.

2D Learning

2D learning is designed for image-based datasets and is about changing the use of data for training. Traditionally, useful features from these images are extracted to train on image-based data for various computer vision tasks. Such features are often one-dimensional and specialize in capturing a particular aspect of the image, such as histograms, corners, or blobs. If the features are 2D, they must be reshaped into 1D vectors before training.

While these features are useful, they have some disadvantages. The first disadvantage is the one-dimensional nature of such features. These features, being one-dimensional, are less suitable for capturing the structural or spatial information that is better stored in a 2D image matrix. Hence, the vectorization defeats the purpose of having 2D data or 2D features. Secondly, these specialized features may not carry all the information that is valuable to the learning task. One has to use a combination of many features to gather the required information from an image. Lastly, as the size of one-dimensional features is often large, we have to face the problem of singularity. For example, if the size of the images in a dataset is $p \times q$, then the size of the feature vector will be pq. If one wishes to use these features, the size of the scatter matrix will be $pq \times pq$, which is very large compared to the small number of data samples in a typical image processing dataset. This issue becomes even more prominent with the emergence of new technology, as high-definition images lead to even larger feature vectors.

To address these issues of 1D methods, one can directly use the original image matrix. The original matrix form stores the spatial information better as it is inherently two-dimensional. The original 2D image has all the information within itself, so we do not need to extract any other features. It also tackles the singularity issue as the number of dimensions of the scatter matrix is significantly less than that when computed with the one-dimensional features. Continuing with the example above, when the matrix form of an image of size $r \times c$ is used directly, the size of the scatter matrix will be $r \times r$ instead of $rc \times rc$. Fig. 1.4 shows how the use of 2D data in its



(a) Scatter matrix using 2D data

(b) Scatter matrix using 1D data

Figure 1.4: Difference between the sizes of scatter matrices- (a) 2D data of size 4×3 produces a scatter matrix of size 4×4. (b) The vectorized form of the same input matrix produces scatter matrix of size 12×12.

original form produces a smaller scatter matrix than the 1D data. Another benefit of 2D data is that its use eliminates the efforts put into feature extraction. These qualities of original 2D image data make it a better candidate for training a classification model. To use it for training, we need methods designed for the 2D data.

2D methods are fairly restricted in terms of the domains where they can be used. However, their impact cannot be ignored. Natural 2D data is used primarily in computer vision and image processing applications. Though conceptualized as early as in 1991 [40], these methods are just now finding their footing in the literature. 2D methods play a significant role for tasks such as face [41], gait [42], and character [43] recognition, among others. However, these are all single-view methods, and the multi-view paradigm has not yet benefited from the advantages these methods offer. The main goal of using 2D data is to reduce computational memory. Hence, methods that deal with multi-view data that can be very large will benefit from 2D methods.

1.1 Contributions of the Thesis

In this thesis, we present four methods that fill four niches in multi-view learning. We introduce incremental-decremental learning to the multi-view domain with our first method that supports the inclusion and removal of data samples. Then, we formulate a multi-view batch learning method for 2D data. This method is then extended to support incremental learning and decremental unlearning of data samples in the 2D multi-view domain in the third method. Lastly, we present a method to incrementally learn or decrementally unlearn the views of data. All of these methods belong to the supervised learning domain and are based on Multi-view Discriminant Analysis (MvDA) [5]. These methods use incremental learning and 2D learning paradigms to achieve their goals. Fig. 1.5 shows where the presented methods stand in the multi-view learning paradigm.

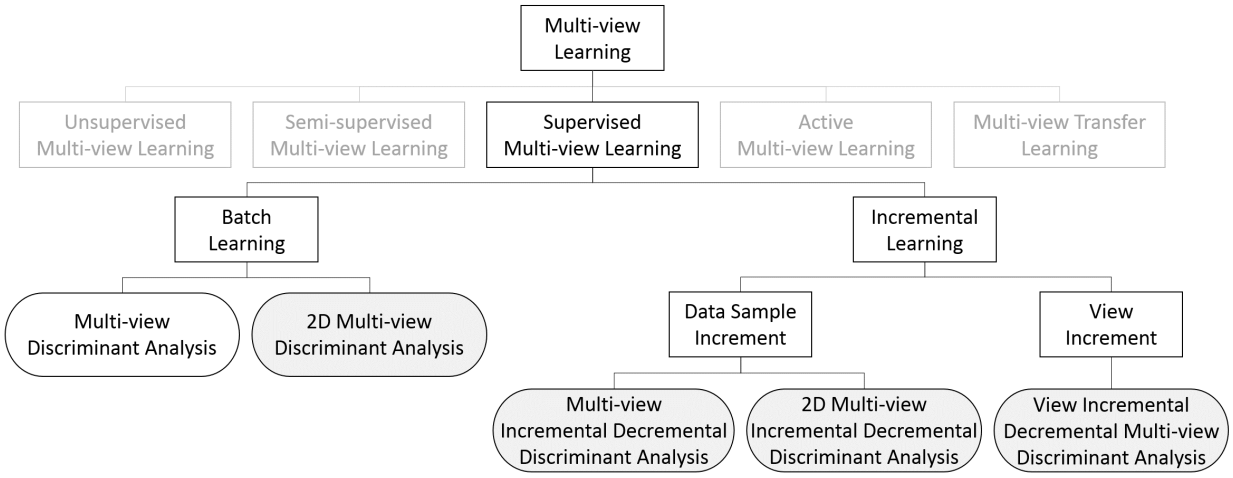


Figure 1.5: The placement of presented methods in the multi-view learning paradigm.

1.1.1 Contribution 1: An Incremental Decremental Method for 1D Multi-view Data

Multi-view batch methods have proved their utility by producing better results than single-view methods. However, when working with incremental data, these methods need to retrain from scratch after every addition or deletion in the dataset. We need incremental methods for training efficiently on the incremental data. The following research questions are investigated in this regard.

RQ1: Can an incremental method, without using old data samples, produce an identical model to that of its batch counterpart MvDA?

RQ2: Is this incremental method invariant to the order of addition of new data samples?

RQ3: Can this incremental method reduce training time?

RQ4: Can this incremental method reduce memory requirements?

RQ5: Is this multi-view incremental method more advantageous than a single-view incremental method in terms of classification accuracy and training time?

Solution: We present Multi-view Incremental Decremental Discriminant Analysis (MvIDDA), which supports the addition/deletion of data samples without using the old data samples. MvIDDA offers a closed-form solution to update the existing model using only the data samples to be added/deleted. It considers all the views and jointly solves the linear transform to find the same optimal common discriminant space as a batch method.

Results: The experiments carried out on three 1D multi-view datasets show that the proposed method indeed finds the same subspace as its batch counterpart. Moreover, the produced model does not depend on the order or the number of data samples in each update. MvIDDA is also shown to perform the updates in less time and memory than MvDA due to its incremental nature. Incremental Linear Discriminant Analysis (ILDA) is a single-view incremental method based on discriminant analysis. The comparison of MvIDDA with ILDA shows that using a multi-view incremental method on multi-view data produces better results than those

obtained using a single-view incremental method. This comparison also shows that MvIDDA requires less training time than ILDA.

1.1.2 Contribution 2: A Batch Method for 2D Multi-view Data

Image-based data is best represented in its original 2D form, as this form preserves spatial information. However, due to the lack of methods that support the use of 2D multi-view data, it must be converted in the vectorized form before it is used for training. To tackle this, we need improved multi-view methods that can train on the 2D form eliminating the need for vectorization. The following research questions are investigated in order to fulfill this need.

RQ1: Can a 2D multi-view method build better classification model than a 1D multi-view method or a 2D single-view method?

RQ2: Can the use of 2D matrices reduce training time?

RQ3: Can the use of 2D matrices reduce memory requirements?

Solution: We present 2D Multi-view Discriminant Analysis (2DMvDA), which can use the 2D multi-view data without having to vectorize it before training. Like MvIDDA, this method also solves the linear transforms of every view together and forms an optimal common discriminant subspace. This method also forms the basis for its incremental version presented in the following contribution.

Results: The experiments carried out on four 2D datasets show that the proposed method benefits from the direct use of 2D data in two ways: (i) it generates significantly smaller scatter matrices, leading to reduced time and memory costs. (ii) it preserves spatial information, leading to improved classification accuracy. The comparison of training time and memory requirements shows that the proposed method needs less time and memory than the 1D multi-view method (MvDA) and a 2D single-view method (2DLDA).

1.1.3 Contribution 3: An Incremental Decremental Method for 2D Multi-view Data

Though MvIDDA has been proposed for multi-view incremental data, it does not support 2D data. For 2D incremental data, we need a method that can train on the original matrix form of the data. Also, to facilitate the removal of the obsolete data samples from the model, we need a method that supports decremental unlearning.

RQ1: Can the 2D incremental-decremental method, without using old data samples, produce an identical model to that of its batch counterpart, 2DMvDA?

RQ2: Is this method invariant to the order of addition or deletion of data samples?

RQ3: Is this method more advantageous than other discriminant analysis-based methods?

Solution: We present a method, 2D Multi-view Incremental Decremental Discriminant Analysis (2DMvIDDA), which supports the addition and deletion of data samples at any point in time. It provides closed-form solutions to update the scatter matrices without retraining after each addition/deletion. This method can also be transformed into other methods based

on discriminant analysis, such as MvIDDA, 2DMvDA, MvDA, 2DLDA, and ILDA, with minor changes to the formulation.

Results: The experiments show that 2DMvIDDA produces the same discriminant subspace as its batch counterpart 2DMvDA and is order-independent. The experiments to compare 2DMvIDDA with other methods based on discriminant analysis are carried out in two parts. The first set compares these methods on rescaled versions of four 2D datasets to show that 2DMvIDDA attains the best classification accuracy among all these methods using considerably less time and memory owing to its use of all three paradigms. The rescaled versions are used here because the 1D methods could not train on the original datasets due to their huge memory requirements. The second set of experiments trains the model using only the 2D methods- 2DMvIDDA, 2DMvDA, and 2DLDA on the original versions of the same four datasets to show the vast difference in memory requirements of 1D vs. 2D methods.

1.1.4 Contribution 4: A View Incremental Decremental Method for Multi-view Data

Increment/decrement in multi-view context is of two types: data sample increment and view increment. MvIDDA and 2DMvIDDA belong to the *data sample increment* category. However, to support the addition or deletion of views of data, we need a view incremental method.

RQ1: Can the incremental-decremental method, without using old views, produce an identical model to that of its batch counterpart MvDA?

RQ2: Is this method invariant to the order of addition or deletion of data samples?

RQ3: Can this method perform as well as the batch method in terms of classification accuracy?

RQ4: Can this method reduce training time?

RQ5: Can this method reduce memory requirements?

Solution: We present View Incremental Decremental Multi-view Discriminant Analysis (VIDMvDA), which supports incremental learning and decremental unlearning of the views of data without retraining. VIDMvDA jointly solves multiple linear transforms to produce the same optimal common discriminant subspace as its batch counterpart MvDA. Though presented for 1D data, VIDMvDA can also be used as a 2D method.

Results: The experiments on three 1D datasets show that the proposed method finds the same or sometimes better common subspace than its batch counterpart, MvDA. It is proven to be order-independent and has the same or slightly better classification accuracy than MvDA. VIDMvDA also updates the model to include or exclude the views using less time and memory than MvDA.

1.2 Organization of the Thesis

The thesis is comprised of eight chapters. The chapter-wise organization of the thesis is given below:

Chapter 2 reviews multi-view batch methods and single-view incremental methods. It also takes a look at the methods that emerged out of a combination of multi-view and incremental learning,

namely data-incremental and view-incremental multi-view methods. We then present single-view batch methods for 2D data, as 2D learning is a relatively new paradigm and does not have any multi-view or incremental methods. In this chapter, we also briefly go through the workings of a recent multi-view batch method, Multi-view Discriminant Analysis (MvDA), which serves as an inspiration for our methods presented in this thesis. MvDA is, as the name suggests, based on discriminant analysis.

Chapter 3 introduces the terminology and notations used by the methods in this thesis and states the assumptions made by these methods. Then, it describes the 1D and 2D datasets used for experiments that measure the performance of presented methods.

Chapter 4 presents the formulations of our first contribution, Multi-view Incremental Decremental Discriminant Analysis (MvIDDA), and the experimental results thereof. MvIDDA is an incremental decremental classification method based on discriminant analysis for multi-view data.

Chapter 5 presents a classification method for 2D multi-view data. This batch method, 2D Multi-view Discriminant Analysis (2DMvDA), serves as a basis to introduce an incremental method in the context of 2D multi-view data in chapter 6.

Chapter 6 presents 2D Incremental Decremental Discriminant Analysis (2DMvIDDA), an incremental-decremental classification method for 2D multi-view data. This method can be converted into MvDA, ILDA, 2DLDA, MvIDDA, and 2DMvDA.

Chapter 7 presents an incremental-decremental method that supports view-wise addition or deletion. This method, View Incremental Decremental Multi-view Discriminant Analysis (VID-MvDA), incrementally learns or decrementally unlearns the views of data.

Chapter 8 highlights the conclusions and summarizes the contributions made. New avenues for future research have also been discussed.

2

Literature Survey

In this chapter, we take a look at recent developments in the paradigms relevant to the methods presented in this thesis, namely- Multi-view batch learning, single-view incremental learning, multi-view incremental learning, and 2D methods. We shall first take a look at the discriminant analysis-based methods for each of these paradigms as the thesis focuses on discriminant analysis, and then we will go through other methods pertaining to these paradigms. At the end of this chapter, we will briefly review Multi-view Discriminant Analysis (MvDA) as this method serves as an inspiration for the methods presented in this thesis.

2.1 Multi-view Batch Methods

Multi-view methods are proposed in nearly all of the machine learning paradigms. Out of those, we will first briefly discuss the methods that are based on discriminant analysis and then discuss other noteworthy multi-view methods.

2.1.1 Discriminant Analysis-based Methods

Multi-view discriminant analysis (MvDA) [5] is a method based on discriminant analysis, which jointly solves multiple linear transforms to compute the optimal projection matrix. This method forms the basis of the methods presented in this thesis. KMvDA [44] is a kernelized version of MvDA, which uses random Fourier features to approximate Gaussian kernels in large-scale learning. The use of kernels enables the construction of a better subspace by projecting the data onto a non-linear higher-dimensional space. Another method is local feature-based multi-view discriminant analysis (FMDA) [45], which employs the local feature descriptor (LFD) matrices and the representation matrices. Representation matrices are comprised of only the non-redundant linear coefficients of the LFDs for each view. The dimensionality is reduced further by using discriminant analysis on these matrices, and the singularity problem is also addressed to some extent. Shu et al. proposed a two-step approach based on MvDA in [46]. They use the Hilbert-Schmidt Independence Criterion (HSIC) to capture the intra-view class information in the first step and CCA to capture the inter-view correlation in the second step. A multi-view method based on Uncorrelated LDA is presented in [6]. This method is a multi-view version of ULDA that uses the constraint of *S-orthogonality* to find uncorrelated projection vectors, leading to better subspace construction. The same paper also presents two non-linear versions of MULDA that use Kernel CCA and Kernel DCCA, respectively, to project the data onto the higher dimensions. Based on the observation that

many of the supervised and unsupervised feature extraction methods follow a similar structure that can be solved as a generalized eigenvalue problem, Sharma et al. [7] present Generalized Multiview Analysis (GMA). This method is a generalized supervised framework and can obtain a linear latent subspace when coupled with other methods like LDA or Marginal Fisher Analysis (MFA). It can also be kernelized to obtain a non-linear subspace.

2.1.1.1 Multi-view Discriminant Analysis (MvDA)

As stated earlier, Multi-view Discriminant Analysis (MvDA) [5] forms a basis of the methods presented in this thesis. Hence, we take a look at the workings of MvDA. This method uses discriminant analysis to project the data samples onto a common subspace. Discriminant analysis works by finding projection vectors that bring the data samples from one class closer together while separating different classes from each other in the discriminant subspace. To apply this principle to multi-view data, MvDA needs to jointly find the projection vectors for all the views. It also needs to keep together the data samples that belong to the same class across all the views. This is achieved by a joint optimization function given as-

$$\mathbf{W}^{opt} = \left(\mathbf{W}_1^{opt}, \cdots, \mathbf{W}_v^{opt}\right) = \underset{(\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W})}{argmax} \frac{Tr\left(\mathbf{S}_B\right)}{Tr\left(\mathbf{S}_W^y\right)} \tag{2.1}$$

Here, v is the number of views. This function maximizes the between-class scatter ($\mathbf{S}_B$) while minimizing the within-class scatter ($\mathbf{S}_W$) in the projected space. The optimal projection matrix $\mathbf{W}^{opt}$ projects data samples so that they are better discriminable in the projection subspace. To find an analytical solution, the function in Eq. (2.1) is reformulated as a ratio-trace function. To achieve this, both the scatter matrices in projected space are defined in the original space as in Eq. (2.2) and Eq. (2.3).

$$\mathbf{S}_W = \begin{bmatrix}\mathbf{W}_1^T & \mathbf{W}_2^T & \cdots & \mathbf{W}_v^T\end{bmatrix} \begin{bmatrix}\mathbf{S}_{11} & \mathbf{S}_{12} & \cdots & \mathbf{S}_{1v} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \cdots & \mathbf{S}_{2v} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{v1} & \mathbf{S}_{v2} & \cdots & \mathbf{S}_{vv}\end{bmatrix} \begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_v\end{bmatrix} = \mathbf{W}^T\mathbf{S}\mathbf{W} \tag{2.2}$$

similarly,

$$\mathbf{S}_B = \begin{bmatrix}\mathbf{W}_1^T & \mathbf{W}_2^T & \cdots & \mathbf{W}_v^T\end{bmatrix} \begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{12} & \cdots & \mathbf{D}_{1v} \\ \mathbf{D}_{21} & \mathbf{D}_{22} & \cdots & \mathbf{D}_{2v} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{v1} & \mathbf{D}_{v2} & \cdots & \mathbf{D}_{vv}\end{bmatrix} \begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_v\end{bmatrix} = \mathbf{W}^T\mathbf{D}\mathbf{W} \tag{2.3}$$

And the optimization function in Eq.(2.1) is transformed as,

$$\mathbf{W}^{opt} = \underset{(\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_v)}{argmax} \frac{Tr\left(\mathbf{W}^T\mathbf{D}\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\mathbf{S}\mathbf{W}\right)} = \underset{(\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_v)}{argmax} Tr\left(\frac{\mathbf{W}^T\mathbf{D}\mathbf{W}}{\mathbf{W}^T\mathbf{S}\mathbf{W}}\right) \tag{2.4}$$

Generalized eigenvalue decomposition method is used to find $\mathbf{W}^{opt}$,

$$\mathbf{D}\mathbf{W}^{opt} = \lambda\mathbf{S}\mathbf{W}^{opt} \tag{2.5}$$

The projection matrix thus found is used to project the test data samples onto the common discriminant subspace and are assigned appropriate class labels.

2.1.2 Other Multi-view Batch Methods

Multi-View Multi-Feature Learning (MVMFL) [8] describes each view by multiple features to convey different information. It sees the features as the projections of latent features extracted from a more discriminant space. The label information guides the extracted latent features belonging to one class to follow the same Gaussian distribution for better classification. Huang et al. present a robust multi-view method based on Lp-norm minimization and Ls-norm maximization in [47]. This method provides a multi-view counterpart for classical GEPSVM methods. The robustness is achieved by using Lp-norm and Ls-norm to calculate the distances from positive and negative points to the hyperplane, which minimizes the effect of the outliers. Another multi-view method based on uncorrelated constraints (MULPP) is presented in [48]. This method is based on the locality preserving projection. MULPP aims to obtain multiple projections and minimize the redundancy of low-dimensional features. To achieve this, it first finds the primary projections of each view and then adds uncorrelated constraints to find more non-redundant low-dimensional projections.

Though these methods make efficient use of the multi-view data to build better classification models, they have been developed for batch learning and do not handle the incremental data.

2.2 Single-view Incremental Methods

The incremental learning paradigm has seen developments in the past few years, presenting many incremental or online versions of learning algorithms. This section briefly goes over some of the single-view incremental methods based on discriminant analysis and other learning algorithms.

2.2.1 Discriminant Analysis-based Methods

Pang et al. [28] proposed an incremental algorithm for LDA that updates the discriminant eigenspace as and when the new data is presented. This method works even when the data samples belonging to an entirely new class are added. GSVD-ILDA [49] is another incremental method based on LDA. This method uses generalized SVD to find the projection matrix in full space. Complete LDA or CLDA, a version of traditional LDA, addresses the problem of singularity of the within-class scatter matrix. A modified CLDA algorithm, along with its incremental version, is presented in [50]. Chu et al. presented an incremental version of LDA/QR in [51], which is faster than LDA, as it uses QR factorization of the data matrix before solving a lower triangular linear system. Liu et al. present an incremental version of discriminant analysis in [52] as kernel null-space discriminant analysis (IKNDA). Along with the discriminant analysis, IKNDA also employs deep neural networks for better performance in novelty detection. Some methods [33, 53, 54, 36] support incremental learning and decremental unlearning too. Incremental decremental SVM (IDSVM) [33] supports the addition or deletion of one data sample at a time without retraining. Karasuyama et al. extended IDSVM in [53] enabling the addition and deletion of multiple data samples at a time. Interleaved IDSVM [54] makes it possible to learn and unlearn at the same time while reducing the computational complexity of the method, thereby reducing the training time by a margin of 60%-70%. Another SVM-based method presented in [36] offers a boosting algorithm for non-linear SVM that supports incremental learning and decremental unlearning.

2.2.2 Other Single-view Incremental Methods

Incremental SVM [55] is one of the earliest incremental versions of SVM. It incrementally learns the classifier while only remembering certain useful support vectors. Older support vectors are

also gradually discarded using the *least recently used* policy and newer support vectors are added. Incremental PCA presented in [56] is used for visual learning. This method is an incremental version of PCA and retains only the learned coefficients and can discard the data after incrementally updating the model. SVDU-IPCA [30] is a specialized method for face recognition based on IPCA that bounds the approximation error to be significantly less. This is achieved by using singular value decomposition (SVD) updating. Incremental PCA-LDA [57] is a combination of two methods as it computes the principal components incrementally without computing the covariance matrix and finds the linear discriminant directions to separate the classes from each other using LDA.

2.3 Multi-view Incremental Methods

The benefits of incremental methods in the single-view context were sure to give rise to their multi-view counterparts. In the multi-view context, however, along with the *data sample increment* we also have *view increment*. We take a look at the methods belonging to both of these categories.

2.3.1 Data Sample Increment

Incremental multi-view passive-aggressive active learning algorithm (IMPAA) [58] supports the *data sample increment*. This classification method is based on active learning and is designed for polarimetric synthetic aperture radar (PolSAR) data. IMPAA assumes increments in the number of data samples and works with two views of data. The data samples on whose labels these two views do not agree are seen as informative data samples. Labels of these data samples are queried and then used to improve the model. One more incremental method [59] to classify the PolSAR data combines semi-supervised and active learning. The method has two phases of learning- one uses active learning to select some informative samples with the help of multi-view information and randomized rule, the next phase employs semi-supervised learning to update the classifier. Both of these methods are equipped to work with the data samples from previously unseen classes. However, these methods do not present a way to add new views of data samples or delete existing data samples or views.

2.3.2 View Increment

Zhou et al. proposed a method based on SVM in [60] which supports the *view increment*. This method, incremental multi-view support vector machine (IMSVM), assumes an increment in the number of views instead of data samples. When a new view is encountered, IMSVM updates the trained model to include information from this view. The complementing information from different views is used for the betterment of the model. Incremental multi-view spectral clustering (IMSC) [61] also supports increment in the number of views. This clustering method learns a consensus kernel from the new views and the base kernels when a new view is added. The base kernels are also updated simultaneously. As IMSVM and IMSC are based on the increment in the number of views, these methods only support the addition of new views of the existing data samples. These methods are not equipped to handle the addition of new data samples, as in the case of IMPAA.

2.4 2D Batch Methods

Eigenfaces [40] and Fisherfaces [62] were the very first methods to use the original image matrix directly. Since then, many methods have been proposed to make use of the 2D data because of the

benefits it has to offer. We present 2D methods based on discriminant analysis and other traditional machine learning algorithms.

2.4.1 Discriminant Analysis-based Methods

2D Linear Discriminant Analysis (2DLDA) [63] is a classification method based on discriminant analysis, which suggests the use of fisher linear projection criterion to overcome singularity. A modified version of 2DLDA was presented in [64] which uses weighted between-class scatter to separate the classes further. Wang et al. presented a formulation of 2DLDA for non-linear data [65], which uses a specially designed convolutional neural network (CNN), facilitating the use of non-linear data. Another 2DLDA-based method is presented in [66]. This method eliminates the outlier and the small sample size problem by using bilateral Lp-norm criterion. Some methods [67, 68] use fuzzy sets in the feature extraction process to improve the performance. A membership degree matrix is computed using fuzzy k-NN, which is then used for classification by these methods. Another method based on 2DLDA is Fuzzy 2DLDA [69], which uses sub-image and random sampling. This method divides the original image into sub-images to make 2DLDA more robust to facial expressions, occlusions, or illumination. In the next step, local features are extracted from sub-images, and the fuzzy 2DLDA algorithm is applied to randomly-sampled row vectors of these sub-images. Cost-sensitive Dual-Bidirectional LDA (CB^{2}LDA) [70] employs the bidirectional LDA along with the misclassification costs to improve classification results. Each misclassification has an associated cost, which is leveraged during the classification phase in this method.

2.4.2 Other Single-view 2D Methods

2DPCA [71], as the name suggests, is a 2D version of traditional PCA. It constructs the image covariance matrix directly from the original images. Eigenvectors of the covariance matrix comprise the feature vectors, which are then used for classification. 2DPCA was further adopted and modified by Kong et al. [72]. They proposed two methods based on 2DPCA in their paper- one is bilateral-projection-based 2DPCA (B2DPCA), and the other is Kernel-based 2DPCA (K2DPCA). B2DPCA constructs two projection directions simultaneously and projects the row and column vectors of the image matrix onto two different subspaces. This way, the image can be represented by fewer coefficients than 2DPCA. K2DPCA is the kernelized version of the 2DPCA. It facilitates the modeling of non-linear structures versus the linear projection technique of the 2DPCA. Another method based on 2DPCA was proposed in [73] by the name of F2DPCA. This method uses the F-norm minimization to make the method robust to outliers. The distance in the attribute domain is computed using F-norm, and the summation over different data points uses the 1-norm. F2DPCA is also robust to the rotation of the images. Angle-2DPCA [74] uses the F-norm along with the relationship between reconstruction error and variance in the criterion function, making the method more robust to the outliers.

3

Notations, Assumptions and Datasets

This chapter introduces the notations and definitions used throughout the thesis. The methods presented in later chapters use the same notations as given here. We also state the common premises about datasets assumed by these methods. Towards the end of this chapter, we list the details of datasets used for experiments.

3.1 Terminologies and Notations

We use lowercase letters to denote a constant (e.g.- n, n_{ij}) and boldface lowercase letters for vectors (e.g.- $\mathbf{x}_{ijk}$). A boldface capital letter (e.g.- $\mathbf{S}_{jr}$) denotes a matrix, and a set is represented using calligraphic capital letter (e.g.- $\mathcal{X}$). The notations used in this thesis are listed in Table 3.1.

As the methods presented in this thesis use either 1D or 2D data samples, the representation varies according to the dimensionality of data. This difference is presented here explicitly. However, at the subsequent mentions of the data samples in this chapter, only the 1D notations are used. Let us denote a multi-view dataset as $\mathcal{X}$. If it is a 1D dataset, it is defined as

$$\mathcal{X} = \{\mathbf{x}_{ijk} | i = 1, \cdots, c; \;\; j = 1, \cdots, v; \;\; k = 1, \cdots, n_{ij}\}$$

Here, each vector $\mathbf{x}_{ijk}$ is the k^{th} data sample from j^{th} view of i^{th} class and n_{ij} is the number of data samples in j^{th} view of i^{th} class.

If $\mathcal{X}$ is a 2D dataset, each of its k^{th} data sample from i^{th} class of j^{th} view is denoted as $\mathbf{X}_{ijk}$. The dataset is defined as

$$\mathcal{X} = \{\mathbf{X}_{ijk} | i = 1, \cdots, c; \;\; j = 1, \cdots, v; \;\; k = 1, \cdots, n_{ij}\}$$

We denote the number of classes with c and the number of views with v. The size of each data sample is $p_j \times q$. The value of p_j may vary across the views, but it is the same for all data samples within a view. The value of q is constant for all the data samples across all the views. For 1D datasets $q = 1$.

Every data sample from the original space is projected into a common discriminant subspace using the projection matrix $\mathbf{W} = \left[\mathbf{W}_1^T \mathbf{W}_2^T \cdots \mathbf{W}_v^T\right]^T$, where each $\mathbf{W}_j$ is a $d \times p_j$ matrix that is used to project jth view. The data samples thus projected are denoted as

$$\mathcal{Y} = \left\{\mathbf{y}_{ijk} = \mathbf{W}_j^T \mathbf{x}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, n_{ij}\right\}$$

Table 3.1: Table shows how the entities are denoted with different notations for existing dataset ($\mathcal{X}$), added/deleted subset ($\bar{\mathcal{X}}$) and the updated dataset after addition/deletion ($\mathcal{X}'$).

Description	Existing	Added/Deleted	Updated
Data sample of 1D dataset	$\mathbf{x}_{ijk}$	$\bar{\mathbf{x}}_{ijk}$	$\mathbf{x}'_{ijk}$
Data sample of 2D dataset	$\mathbf{X}_{ijk}$	$\bar{\mathbf{X}}_{ijk}$	$\mathbf{X}'_{ijk}$
Size of 1D data samples	$p_j \times 1$	-	-
Size of 2D data samples	$p_j \times q$	-	-
No. of classes	c	$\bar{c}$	c'
Set of classes	$\mathcal{C}$	$\bar{\mathcal{C}}$	$\mathcal{C}'$
No. of views	v	$\bar{v}$	v'
Set of views	$\mathcal{V}$	$\bar{\mathcal{V}}$	$\mathcal{V}'$
No. of data samples per class per view	n_{ij}	$\bar{n}_{ij}$	n'_{ij}
No. of data samples per class	n_i	$\bar{n}_i$	n'_i
No. of total data samples	n	$\bar{n}$	n'
Mean per class per view of 1D dataset	$\mathbf{m}_{ij}^{(\mathbf{x})}$	$\bar{\mathbf{m}}_{ij}^{(\mathbf{x})}$	$\mathbf{m}'_{ij}{}^{(\mathbf{x})}$
Class mean of 1D dataset	$\mathbf{m}_i$	$\bar{\mathbf{m}}_i$	$\mathbf{m}'_i$
Total mean of 1D dataset	$\mathbf{m}$	$\bar{\mathbf{m}}$	$\mathbf{m}'$
Mean per class per view of 2D dataset	$\mathbf{M}_{ij}^{(\mathbf{x})}$	$\bar{\mathbf{M}}_{ij}^{(\mathbf{x})}$	$\mathbf{M}'_{ij}{}^{(\mathbf{x})}$
Class mean of 2D dataset	$\mathbf{M}_i$	$\bar{\mathbf{M}}_i$	$\mathbf{M}'_i$
Total mean of 2D dataset	$\mathbf{M}$	$\bar{\mathbf{M}}$	$\mathbf{M}'$
Within-class scatter in projected space	$\mathbf{S}_W$	-	$\mathbf{S}'_W$
Within-class scatter in original space	$\mathbf{S}$	-	$\mathbf{S}'$
Between-class scatter in projected space	$\mathbf{S}_B$	-	$\mathbf{S}'_B$
Between-class scatter in original space	$\mathbf{D}$	-	$\mathbf{D}'$
Projection matrix	$\mathbf{W}$	-	-
No. of projection vectors	d	-	-
Projected 1D data samples	$\mathbf{y}_{ijk}$	$\bar{\mathbf{y}}_{ijk}$	$\mathbf{y}'_{ijk}$
Projected 2D data samples	$\mathbf{Y}_{ijk}$	$\bar{\mathbf{Y}}_{ijk}$	$\mathbf{Y}'_{ijk}$

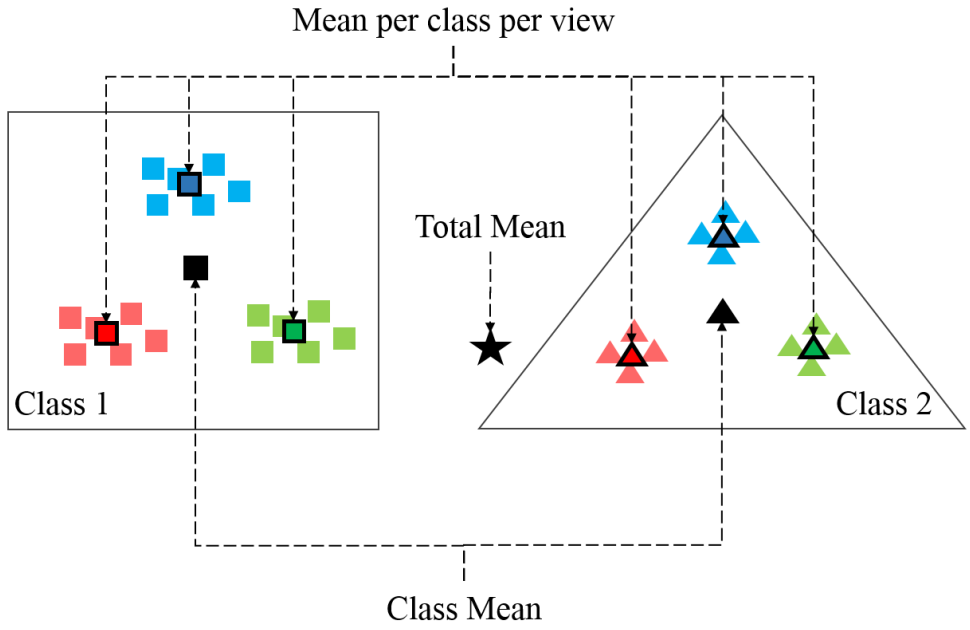


Figure 3.1: The figure shows the three means pictorially. Three views are shown with three colors- blue, red and green. Two classes, Class1 and Class2, are depicted with squares and triangles, respectively.

Each $\mathbf{y}_{ijk}$ is of size $d \times 1$, where d is the number of projection vectors. Each projected 2D data sample is denoted as $\mathbf{Y}_{ijk}$ and is of size $d \times q$.

There are three data sample counts- number of data samples per class per view (n_{ij}), number of data samples per class (n_i), and the total number of data samples (n). Similarly, we also have three corresponding means- mean per class per view ($\mathbf{m}_{ij}^{(\mathbf{x})}$), class mean ($\mathbf{m}_i$) and the total mean ($\mathbf{m}$). Table 3.2 lists these entities along with their equations and interrelations. The mean per class per view is denoted with a superscript ($\mathbf{x}$) because it is computed in the original space. However, the other two means are computed in the projected space because their computations involve data samples from all the views that are not comparable in the original space.

The increments in the dataset can be of two types: (i) data sample increment- where the number of views remains the same and only the number of data samples changes over time, or (ii) view increment- where the number of data samples remains the same and the number of views changes over time. The increase or decrease in the number of data samples can be sequential (one-by-one) or in chunks (groups of data samples). A set of data samples to be added/deleted is denoted by $\bar{\mathcal{X}}$. If any data sample from $\bar{\mathcal{X}}$ belongs to a new class, this class is denoted by N, and if it belongs to an already existing class, we denote its class by E. The data sample count or the means related to $\bar{\mathcal{X}}$ are denoted with a bar symbol (e.g., $\bar{m}$) over them and those related to the updated dataset ($\mathcal{X}'$) are denoted with a prime symbol over them (e.g., m').

3.2 Assumptions

We first state three fundamental assumptions made by all of the four methods.

A) The label of the new data sample is made available at the time of its inclusion.

Table 3.2: Notations and Formulae

Notations	Formulae
n_{ij}	-
n_i	$\sum_{j=1}^{v} n_{ij}$
n	$\sum_{i=1}^{c} n_i$
$\mathbf{m}_{ij}^{(\mathbf{x})}$	$\frac{1}{n_{ij}} \sum_{k=1}^{n_{ij}} \mathbf{x}_{ijk}$
$\mathbf{m}_i$	$\frac{1}{n_i} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \mathbf{y}_{ijk}$
$\mathbf{m}$	$\frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \mathbf{y}_{ijk}$

- As all the four methods belong to the supervised learning domain, the class label of a data sample is a non-negotiable requirement. As the model is updated as soon as new data is added, without a label, the data sample will be of no use to the algorithm and will be discarded.

B) Each data sample is present in all of the v views.

- This is one of the fundamental assumptions of multi-view learning. All of the views are assumed to contain the information of each data sample in the dataset. In the real world, this may not always be true. Hence, some efforts are being made towards approximating the missing view information [75, 76].

C) All of the views agree with each other on the labels of the training data samples.

- This is another fundamental assumption of multi-view learning and is very intuitive too. As all the views give information about the same object to solve the same problem, they must agree on the class to which the given sample belongs. If the views disagree on the label, the data sample cannot be used for training, as supervised learning is based on the labeled data.

Other assumptions made by data sample incremental/decremental methods-

a) At the start of the learning process, we may or may not have any existing data samples, i.e., $n \geqslant 0$.

- All of the incremental/decremental methods presented in this thesis support learning from scratch (cold-start) or learning on top of the already existing model (warm-start) without any change in the formulations. Hence, these methods work as well when there is no previous model as they work when updating a previously existing model.

b) New data samples may belong to an already existing class or a new class.

- All of these methods support the addition of new data samples to already existing classes or to a previously unseen class in any increment. MvIDDA presents these cases as different specialized formulations. However, both the cases can be taken care of by one generalized formulation as demonstrated with 2DMvIDDA formulations.

c) In the case of addition, all the views of new data samples are added at once to the dataset.

- This assumption is the same as Assumption (B), but written separately in the incremental context because the learning method expects that all the data belonging to a data sample be available at the time of training. So, if one or more views of a new data sample are missing, it cannot be included in the updated model as it violates the fundamental assumption of multi-view learning.

d) In the case of deletion, all views of the data samples to be deleted are removed from the dataset.

- Same as in the case of addition, if we wish to remove a data sample, it has to be removed completely. Information from all the views of a data sample belongs to that data sample. Hence, all of it is removed when deleting that data sample.

Other assumptions made by view incremental/decremental methods-

a) At the start of the learning process, one or more views of data samples may have already been presented.

- Similar to the first assumption of *data sample increment* above, this assumption states that the learning method is capable of learning a new model when no data was present previously, as well as updating an already existing model.

b) In the case of addition, new views of all the existing data samples are added at once.

- As Assumption-B states, a data sample must be present in all the views. Hence, whenever a new view is added, it must contain the information of all the data samples that exist in the previous views. If any data sample is missing in a new view, the model cannot accept that view for training.

c) In the case of deletion, the whole view is deleted at once.

- Same as above, a view cannot be partially removed as it will violate the fundamental assumption of multi-view learning. If the whole view is deleted, the remaining views still confirm to the assumption that each data sample is present in all the views.

3.3 Datasets

In this section, we first present the details of 1D datasets used for experiments on the 1D methods in chapter 4 and 7. We then present the details of 2D datasets used by the 2D methods in chapter 5 and 6.

3.3.1 1D Datasets

We have used three widely used multi-view datasets for the experiments. All datasets are based on images and their features. Each feature of these images comprises a different view.

3.3.1.1 Handwritten Digits Dataset

Handwritten Digits Dataset [77] is from the UCI Machine Learning Repository. This dataset contains features extracted from 2000 images of handwritten digits from Dutch utility maps. There are ten classes, one corresponding to each digit from 0 to 9, and each class has 200 images. Each set of features constitutes one view. Details of the description of views and their corresponding dimensions are given in Table 3.3.

Table 3.3: Details of Handwritten Digits Dataset

View#	Description of View	Dimension
1	Profile correlations	216
2	Fourier coefficients	76
3	Karhunen-Love coefficients	64
4	Morphological features	6
5	Pixel averages in 2 x 3 windows	240
6	Zernike moments	47

3.3.1.2 Caltech-7 Dataset

Caltech-7 is a subset of Caltech-101 dataset [78]. This data set contains a total of 1000 images of 7 distinct objects, namely, Faces (435), Motorbikes (324), Dollar-bill (52), Garfield (34), Snoopy (35), Stop-sign (64), and Windsor-chair (56). Numbers in the brackets denote the cardinality of each class. Each image has a size of around 300×200 pixels. Six views are composed of six different features extracted from the images. Description of the views and their dimensions are given in Table 3.4.

Table 3.4: Details of Caltech-7 Dataset

View#	Description of View	Dimension
1	Cenhist Features(CEN)	254
2	Gabor Wavelet Features(GAB)	48
3	Gist Features(GIS)	512
4	Histogram of Oriented Gradients(HOG)	1984
5	Local Binary Pattern(LBP)	928
6	Wavelet Moments(WAV)	40

3.3.1.3 AwA Dataset

AwA (Animals with Attributes) is a large dataset [79] containing a total of 30475 images of 50 animals. The cardinality of each of the 50 classes is different and ranges from 92 to 1168. This dataset has six image features extracted from the original images. Each feature is considered as one view. Description of the views and their dimensions are given in Table 3.5.

Table 3.5: Details of AwA Dataset

View#	Description of View	Dimension
1	Color Histogram(CQ)	2688
2	Local Self-similarity Features(LSS)	2000
3	Pyramid HOG(PHOG)	252
4	Scale-Invariant Feature Transform(SIFT)	2000
5	Color SIFT(RGBSIFT)	2000
6	Speeded-Up Robust Features(SURF)	2000

Table 3.6: Details of IMPART Face Dataset

View#	Description of View	Dimensions	Rescaled to
1	Photographs from angle 0°	1920×1080	54×96
2	Photographs from angle 30°	1920×1080	54×96
3	Photographs from angle 45°	1920×1080	54×96
4	Photographs from angle $-30°$	1920×1080	54×96
5	Photographs from angle $-45°$	1920×1080	54×96

3.3.2 2D Datasets

For experiments involving the 2D methods, we have used four widely used multi-view image datasets: the IMPART face dataset [80], the MSSpoof dataset [1], the Stereo face dataset [81] and the ORL dataset [82]. The first three datasets are based on images of faces taken from different angles. Photographs of the participants taken from one angle compose one view in each of these datasets. As the ORL dataset contains photographs of the participants only from the front view, we have extracted 2D features from these images to make up two more views. Detailed information about all datasets is provided below. Due to the higher memory requirements of 1D methods, we have rescaled the images from original datasets and used them for the experiments that include such methods. However, for experiments that compare only the 2D methods, we have used original datasets to show the full strength of the 2D methods.

3.3.2.1 IMPART Face Dataset

IMPART Face Dataset [80] was created within the EU FP7 IMPART project. This dataset contains photographs of 10 participants - seven males and three females. Each person corresponds to one class. Five different views of the participants are captured from five different angles. The description of views and their corresponding dimensions are given in Table 3.6. Each view contains six photographs of each participant that correspond to six facial expressions: neutral, anger, fear, happiness, sadness, and surprise.

3.3.2.2 MSSpoof Dataset

We have taken a subset of the MSSpoof dataset [1]. This subset contains photographs of 15 participants, hence, 15 classes. Out of two views, one contains photographs in the visual spectrum, and the other contains photographs in the infra-red spectrum. The number of photographs per view per person is 28. Description of the views and their dimensions are given in Table 3.7.

Table 3.7: Details of MSSpoof Dataset

View#	Description of View	Dimensions	Rescaled to
1	VIS spectrum photographs	1024×1280	64×80
2	NIR spectrum photographs	1024×1280	64×80

3.3.2.3 Stereo Face Dataset

Stereo Face Dataset [81] contains photographs of 100 participants (55 males and 45 females). Two cameras are used to provide two views, i.e., the angles of capture. Each camera captures eight photographs per person in different poses. Details of views and their dimensions are given in Table 3.8.

Table 3.8: Details of Stereo Face Dataset

View#	Description of View	Dimensions	Rescaled to
1	Photographs by Camera-1	480×640	96×128
2	Photographs by Camera-2	480×640	96×128

3.3.2.4 ORL Dataset

ORL [82] is a face dataset created in 1992 consisting of 400 images. This dataset contains ten photographs of each participant taken from different angles over two years. There are 40 such participants, hence 40 classes. We extracted the FFT [83] and the Canny [84] features from these images and used them as two views along with the original images as the first view. Details of views are given in Table 3.9.

Table 3.9: Details of ORL Dataset

View#	Description of View	Dimensions	Rescaled to
1	Original Images	112×92	56×46
2	FFT	112×92	56×46
3	Canny Features	112×92	56×46

4

Multi-view Incremental Decremental Discriminant Analysis

We present an incremental decremental classification method for 1D multi-view data in this chapter. The batch methods discard an existing model when new data samples are added to the dataset. These methods learn a new model on the updated dataset consisting of old and new data samples. So, to add one new data sample to an existing dataset of a thousand data samples, a batch method will retrain on a thousand and one data samples spending a lot of time and memory. To alleviate these limitations, we propose Multi-view Incremental Decremental Discriminant Analysis (MvIDDA) that supports the addition and deletion of data samples without retraining the model from scratch. It also supports the addition of data samples from existing as well as previously unseen classes. MvIDDA, like its batch counterpart MvDA, is based on discriminant analysis. However, unlike MvDA, it is an incremental method that does not require retraining on all the historical data. It only needs the data samples to be added/removed and the existing model for update and hence, can learn the same model as MvDA using less training time and memory.

Fig. 4.1 shows an example of sequential increment and chunk increment. In sequential increments, only one data sample is added at a time. This sample is present in all the views. Each view is shown in a different color. In chunk increment, a group of data samples is added at a time. This chunk may have data samples from one or more existing classes or even new classes. Different classes are shown in different shapes in the figure. The same is the case with decrements in the data.

We first present the incremental and decremental formulations of MvIDDA, and then the experimental setup and results are presented. We have compared MvIDDA with its batch multi-view counterpart MvDA and single-view incremental counterpart Incremental Linear Discriminant Analysis (ILDA). We conclude this chapter by summarizing the features of the proposed method.

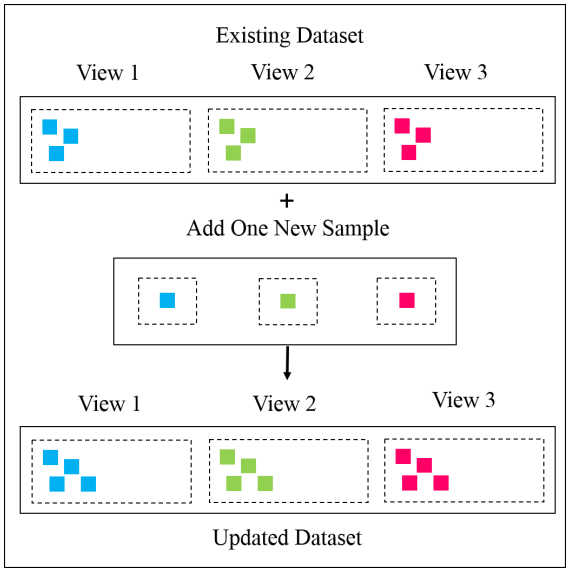
4.1 Methodology

4.1.1 Addition of Data Samples

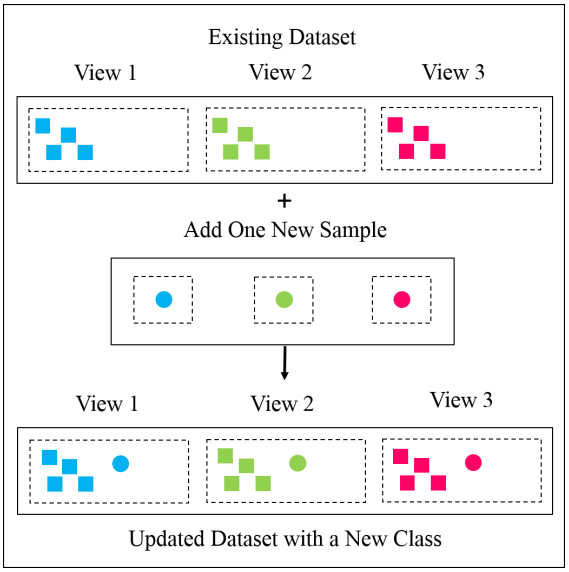
The incremental formulation of MvIDDA has four different cases depending on the nature of an incoming data stream and the class labels of new data samples.

- **Sequential increment and Existing class**- To be used when only one new sample is added at a time and it belongs to one of the already existing classes (Refer fig. 4.1a).

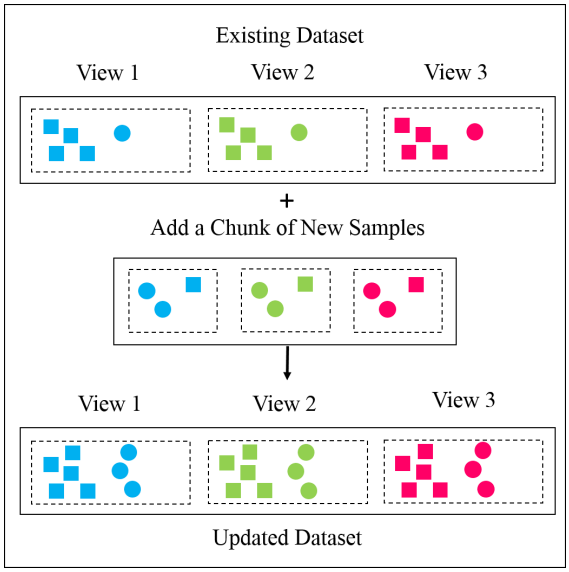
- **Sequential increment and New class**- To be used when only one new sample is added at a time and it belongs to a new class (Refer fig. 4.1b).



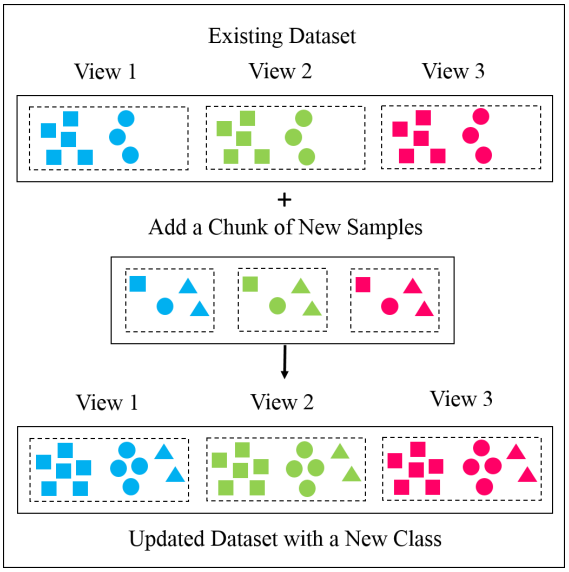
(a) Sequential Increment and Existing Class



(b) Sequential Increment and New Class



(c) Chunk Increment and Existing Class



(d) Chunk Increment and New Class

Figure 4.1: Cases of Increment: (a) A new data sample is added to every view. It belongs to an existing class. (b) A new data sample is added to every view. It belongs to a new class. (c) A chunk of new data samples is added. None of the new data samples belong to a new class. (d) A chunk of new data samples is added. Some of the new data samples belong to a new class.

- **Chunk increment and Existing class**- To be used when a group of new samples is added at a time and none of the new samples belongs to a new class (Refer fig. 4.1c).
- **Chunk increment and New class**- To be used when a group of new samples is added at a time and some of the new samples belong to new classes (Refer fig. 4.1d).

Each of the four cases of MvIDDA follows four steps that incrementally update- (i) number of data samples, (ii) means, (iii) within-class scatter matrix, and (iv) between-class scatter matrix, respectively. Once these entities are updated, the new scatter matrices are used to find optimal projection vectors using Eq.(2.5). Algorithm 1 presents the workings of incremental MvIDDA.

4.1.1.1 Sequential Increment and Existing Class

Let us suppose only one data sample is added to the base set at a time and it belongs to an already existing class E. Let us denote this data sample by $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ and $\bar{\mathbf{x}}_j \in \mathbb{R}^{p_j}$. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_j = \mathbf{W}_j^T \bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples- As the class already exists in $\mathcal{C}$, we have to update the number of total data samples (n), the number of data samples in class E (n_E), and the number of data samples per class per view of class E (n_{Ej}). Here, n_{ij} and n_i for other classes $i \in \mathcal{C} - \{E\}$ do not change.

$$n'_{Ej} = n_{Ej} + 1, \;\; n'_E = n_E + v \text{ and } \;\; n' = n + v$$

B) Updating the means- As the new data sample has been added to class E, we update the means per class per view (${\mathbf{m}_{Ej}}^{(x)}$) and class mean ($\mathbf{m}_E$) of class E. The means of remaining classes do not need updating as these classes were not updated in the increment. We also update the total mean ($\mathbf{m}$).

$${\mathbf{m}'_{Ej}}^{(x)} = \frac{n_{Ej}{\mathbf{m}_{Ej}}^{(x)} + \bar{\mathbf{x}}_j}{n_{Ej} + 1} \tag{4.1}$$

$$\mathbf{m}'_E = \frac{n_E \mathbf{m}_E + v\bar{\mathbf{m}}_E}{n_E + v} = \frac{n_E \mathbf{m}_E + \sum_{j=1}^{v} \bar{\mathbf{y}}_j}{n_E + v} \tag{4.2}$$

$$\mathbf{m}' = \frac{n\mathbf{m} + v\bar{\mathbf{m}}}{n + v} = \frac{n\mathbf{m} + \sum_{j=1}^{v} \bar{\mathbf{y}}_j}{n + v} \tag{4.3}$$

C) Updating the within-class scatter matrix- As the within-class scatter of classes other than E does not change, we only update the scatter of class E. The scatter of the rest of the classes does not need recomputing.

$$\begin{aligned} \mathbf{S}'_W &= \sum_{i=1, i\neq E}^{c} \mathbf{S}_{W_i} + \mathbf{S}'_{W_E} \\ &= \mathbf{S}_W + \frac{v\, n_E}{(n_E + v)} (\bar{\mathbf{m}}_E - \mathbf{m}_E)(\bar{\mathbf{m}}_E - \mathbf{m}_E)^T + \sum_{j=1}^{v} \left(\bar{\mathbf{y}}_j - \mathbf{m}_E\right)\left(\bar{\mathbf{y}}_j - \mathbf{m}_E\right)^T \end{aligned} \tag{4.4}$$

Algorithm 1 MvIDDA algorithm

Input :
- A trained model ϕ that consists of:
  - Data sample counts: n, n_i, n_{ij}
  - Means: $\mathbf{m}$, $\mathbf{m}_i$, $\mathbf{m}_{ij}$
  - Scatter matrices: $\mathbf{S}$, $\mathbf{D}$
  - Set of class labels of existing data samples: $\mathcal{C}$
- Set of new data samples: $\bar{\mathcal{X}}$
- Number of new data samples: $\bar{n}$
- Set of class labels of new data samples: $\bar{\mathcal{C}}$

while new data is encountered **do**
  if $\bar{n} == 1$ **then**
    if $\bar{\mathcal{C}} \in \mathcal{C}$ **then**
      Update n, n_i, n_{ij}
      Update $\mathbf{m}$, $\mathbf{m}_i$, $\mathbf{m}_{ij}$ using Eq.(4.29)-(4.31)
      Update $\mathbf{S}$ using Eq.(4.33)
      Update $\mathbf{D}$ using Eq.(4.35)
    else
      Update n, n_i, n_{ij}
      Update $\mathbf{m}$, $\mathbf{m}_i$, $\mathbf{m}_{ij}$ using Eq.(4.8)-(4.10)
      Update $\mathbf{S}$ using Eq.(4.12)
      Update $\mathbf{D}$ using Eq.(4.14)
      $\mathcal{C} = \mathcal{C} \cup \bar{\mathcal{C}}$
    end if
  else
    if $\bar{\mathcal{C}} \in \mathcal{C}$ **then**
      Update n, n_i, n_{ij}
      Update $\mathbf{m}$, $\mathbf{m}_i$, $\mathbf{m}_{ij}$ using Eq.(4.15)-(4.17)
      Update $\mathbf{S}$ using Eq.(4.19)
      Update $\mathbf{D}$ using Eq.(4.21)
    else
      Update n, n_i, n_{ij}
      Update $\mathbf{m}$, $\mathbf{m}_i$, $\mathbf{m}_{ij}$ using Eq.(4.36)-(4.38)
      Update $\mathbf{S}$ using Eq.(4.40)
      Update $\mathbf{D}$ using Eq.(4.42)
      $\mathcal{C} = \mathcal{C} \cup \bar{\mathcal{C}}$
    end if
  end if
  Compute the projection matrix $\mathbf{W}^{opt}$ using Eq.(2.5)
end while
Output :
- Updated model ϕ
- The projection matrix $\mathbf{W}^{opt}$

and the reformulation of above equation is,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jr} + \frac{n_E}{v(n_E+v)} \left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)^T + \bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T - \frac{1}{v}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T & j = r \\ \mathbf{S}_{jr} + \frac{n_E}{v(n_E+v)} \left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_r - \mathbf{m}_{Er}{}^{(x)}\right)^T - \frac{1}{v}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_r^T & j \neq r \end{cases} \tag{4.5}$$

For detailed derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ refer to Appendix-B.

D) Updating the between-class scatter matrix- The between-class scatter depends on the total mean and class means. Hence, it has to be recalculated as the total mean and the class mean of class E are updated after the increment.

$$\begin{aligned} \mathbf{S}'_B &= \sum_{i=1, i\neq E}^{c} n_i(\mathbf{m}_i - \mathbf{m}')(\mathbf{m}_i - \mathbf{m}')^T + n'_E(\mathbf{m}'_E - \mathbf{m}')(\mathbf{m}'_E - \mathbf{m}')^T \\ &= \sum_{i=1}^{c} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \end{aligned} \tag{4.6}$$

and the reformulated between-class scatter is,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c} \frac{n'_{ij}\, n'_{ir}}{n'_i}\, \mathbf{m}'_{ij}{}^{(x)} \mathbf{m}'_{ir}{}^{(x)^T} - \frac{1}{n'} \left(\sum_{i=1}^{c} n'_{ij}\, \mathbf{m}'_{ij}{}^{(x)}\right)\left(\sum_{i=1}^{c} n'_{ir}\, \mathbf{m}'_{ir}{}^{(x)}\right)^T \tag{4.7}$$

For detailed derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ refer to Appendix-A.

4.1.1.2 Sequential Increment and New Class

The second case is when the newly added data sample belongs to a new class N. Let us denote this data sample by $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ and $\bar{\mathbf{x}}_j \in \mathbb{R}^{p_j}$. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_j = \mathbf{W}_j^T \bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples- As this is the first data sample in its class, we initialize the number of data samples per class per view for this new class as, $n_{Nj} = 1$. Also, the number of data samples per class, $n_N = v$ and updated total number of data samples, $n' = n + v$. Both, n_{ij} and n_i for $i \in \mathcal{C}$ do not change.

B) Updating the means-

$$\mathbf{m}'_{Nj}{}^{(x)} = \bar{\mathbf{m}}_{Nj}^{(x)} = \bar{\mathbf{x}}_j \tag{4.8}$$

$$\mathbf{m}'_N = \bar{\mathbf{m}}_N = \frac{1}{v}\sum_{j=1}^{v} \mathbf{y}'_j \tag{4.9}$$

$$\mathbf{m}' = \frac{n\mathbf{m} + v\bar{\mathbf{m}}}{n+v} = \frac{n\mathbf{m} + \sum_{j=1}^{v} \bar{\mathbf{y}}_j}{n+v} \tag{4.10}$$

C) Updating the within-class scatter matrix- We only add the within-class scatter of the new class to the old $\mathbf{S}_W$, as the within-class scatter of existing classes does not change.

$$\mathbf{S}'_W = \mathbf{S}_W + \sum_{j=1}^{v} (\bar{\mathbf{y}}_j - \bar{\mathbf{m}}_N)(\bar{\mathbf{y}}_j - \bar{\mathbf{m}}_N)^T \tag{4.11}$$

and the reformulation of above equation is,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jr} + \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_j^T - \frac{1}{v^2} \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_j^T & j = r \\ \mathbf{S}_{jr} - \frac{1}{v^2} \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_r^T & j \neq r \end{cases} \tag{4.12}$$

For detailed derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ refer to Appendix-C.

D) Updating the between-class scatter matrix- Like before, the between-class scatter has to be recalculated as,

$$\begin{aligned} \mathbf{S}'_B &= \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m}')(\mathbf{m}_i - \mathbf{m}')^T + v(\bar{\mathbf{m}}_N - \mathbf{m}')(\bar{\mathbf{m}}_N - \mathbf{m}')^T \\ &= \sum_{i=1}^{c+1} n'_i (\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \end{aligned} \tag{4.13}$$

and the reformulated between-class scatter is,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c+1} \frac{n'_{ij} \, n'_{ir}}{n'_i} \, \mathbf{m}'^{(x)}_{ij} \mathbf{m}'^{(x)T}_{ir} - \frac{1}{n'} \left(\sum_{i=1}^{c+1} n'_{ij} \, \mathbf{m}'^{(x)}_{ij} \right) \left(\sum_{i=1}^{c+1} n'_{ir} \, \mathbf{m}'^{(x)}_{ir} \right)^T \tag{4.14}$$

After the completion of all four steps, we add the new class in the set of existing classes ($\mathcal{C}$).

4.1.1.3 Chunk Increment and Existing Class

For the first case of chunk increment, $\bar{n}_i$ of the $\bar{n}$ new data samples belong to class $i \in \mathcal{C}$, meaning, none of the new data samples belong to a class that has not yet been introduced. We denote a chunk of new data samples by, $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ and $\bar{\mathbf{x}}_{ijk} \in \mathbb{R}^{p_j}$. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_{ijk} = \mathbf{W}_j^T \bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples-

$$n'_{ij} = n_{ij} + \bar{n}_{ij}, \;\; n'_i = n_i + \bar{n}_i \text{ and } \;\; n' = n + \bar{n}$$

for $i = 1, \cdots, c$ and $j = 1, \cdots, v$.

B) Updating the means-

$$\mathbf{m}'_{ij}{}^{(x)} = \frac{n_{ij}\mathbf{m}_{ij}{}^{(x)} + \bar{n}_{ij}\bar{\mathbf{m}}_{ij}{}^{(x)}}{n_{ij} + \bar{n}_{ij}} = \frac{n_{ij}\mathbf{m}_{ij}{}^{(x)} + \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}}{n_{ij} + \bar{n}_{ij}} \tag{4.15}$$

$$\mathbf{m}'_i = \frac{n_i\mathbf{m}_i + \bar{n}_i\bar{\mathbf{m}}_i}{n_i + \bar{n}_i} = \frac{n_i\mathbf{m}_i + \sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n_i + \bar{n}_i} \tag{4.16}$$

$$\mathbf{m}' = \frac{n\mathbf{m} + \bar{n}\bar{\mathbf{m}}}{n + \bar{n}} = \frac{n\mathbf{m} + \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n + \bar{n}} \tag{4.17}$$

C) Updating the within-class scatter matrix- The within-class scatter of only the classes to which new data samples were added is updated. Hence, we have,

$$\mathbf{S}'_W = \mathbf{S}_W + \sum_{i=1}^{c}\left[\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}}\left(\bar{\mathbf{y}}_{ijk}\bar{\mathbf{y}}_{ijk}^T - \bar{\mathbf{y}}_{ijk}\mathbf{m}_i{}^T\right) + \frac{\bar{n}_i n_i}{(n_i + \bar{n}_i)}\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T\right] \tag{4.18}$$

and the reformulation of above equation is,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jj} + \sum\limits_{i=1}^{c}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i (n_i + \bar{n}_i)^2}\mathbf{a}\mathbf{a}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\mathbf{m}_{ij}^{(x)}\mathbf{m}_{ij}^{(x)^T} + \sum\limits_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T\right] & j = r \\ \mathbf{S}_{jr} + \sum\limits_{i=1}^{c}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i (n_i + \bar{n}_i)^2}\mathbf{a}\mathbf{b}^T - \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\mathbf{m}_{ij}^{(x)}\mathbf{m}_{ir}^{(x)^T}\right] & j \neq r \end{cases} \tag{4.19}$$

Where,
$\mathbf{a} = \bar{n}_{ij}n_i\bar{\mathbf{m}}_{ij}^{(x)} - \bar{n}_i n_{ij}\mathbf{m}_{ij}^{(x)}$ and $\mathbf{b} = \bar{n}_{ir}n_i\bar{\mathbf{m}}_{ir}^{(x)} - \bar{n}_i n_{ir}\mathbf{m}_{ir}^{(x)}$
For detailed derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ refer to Appendix-D.

D) Updating the between-class scatter matrix- The between-class scatter is recalculated as,

$$\mathbf{S}'_B = \sum_{i=1}^{c} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \tag{4.20}$$

and the reformulated between-class scatter is,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c}\frac{n'_{ij}\,n'_{ir}}{n'_i}\,\mathbf{m}'_{ij}{}^{(x)}\mathbf{m}'_{ir}{}^{(x)^T} - \frac{1}{n'}\left(\sum_{i=1}^{c} n'_{ij}\,\mathbf{m}'_{ij}{}^{(x)}\right)\left(\sum_{i=1}^{c} n'_{ir}\,\mathbf{m}'_{ir}{}^{(x)}\right)^T \tag{4.21}$$

4.1.1.4 Chunk Increment and New Class

In second case of chunk increment, we assume that out of $\bar{n}$ new data samples, some samples belong to new classes. The number of new classes may be one or more. Here, we denote a chunk of new data samples by, $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c'; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ and $\bar{\mathbf{x}}_{ijk} \in \mathbb{R}^{p_j}$. Here, c' is the number of classes after adding the new classes. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_{ijk} = \mathbf{W}_j^T\bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c'; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples-

$$n'_{ij} = n_{ij} + \bar{n}_{ij}, \;\; n'_i = n_i + \bar{n}_i \text{ and } \;\; n' = n + \bar{n}$$

for $i = 1, \cdots, c'$ and $j = 1, \cdots, v$.

B) Updating the means-

$$\mathbf{m'}_{ij}^{(x)} = \frac{n_{ij}\mathbf{m}_{ij}^{(x)} + \bar{n}_{ij}\bar{\mathbf{m}}_{ij}^{(x)}}{n_{ij} + \bar{n}_{ij}} = \frac{n_{ij}\mathbf{m}_{ij}^{(x)} + \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}}{n_{ij} + \bar{n}_{ij}} \tag{4.22}$$

$$\mathbf{m'}_i = \frac{n_i\mathbf{m}_i + \bar{n}_i\bar{\mathbf{m}}_i}{n_i + \bar{n}_i} = \frac{n_i\mathbf{m}_i + \sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n_i + \bar{n}_i} \tag{4.23}$$

$$\mathbf{m'} = \frac{n\mathbf{m} + \bar{n}\bar{\mathbf{m}}}{n + \bar{n}} = \frac{n\mathbf{m} + \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n + \bar{n}} \tag{4.24}$$

C) Updating the within-class scatter matrix- The within-class scatter of the existing and the new classes to which new data samples were added is updated. Hence, we have,

$$\mathbf{S'}_W = \mathbf{S}_W + \sum_{i=1}^{c'}\left[\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}}\left(\bar{\mathbf{y}}_{ijk}\bar{\mathbf{y}}_{ijk}^T - \bar{\mathbf{y}}_{ijk}\bar{\mathbf{m}}_i{}^T\right) + \frac{\bar{n}_i n_i}{(n_i + \bar{n}_i)}\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T\right] \tag{4.25}$$

and the reformulation of above equation is,

$$\mathbf{S'}_{jr} = \begin{cases} \mathbf{S}_{jj} + \sum\limits_{i=1}^{c'}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i (n_i + \bar{n}_i)^2}\mathbf{a}\mathbf{a}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ij}^{(x)^T} + \sum\limits_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T\right] & j = r \\ \mathbf{S}_{jr} + \sum\limits_{i=1}^{c'}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i (n_i + \bar{n}_i)^2}\mathbf{a}\mathbf{b}^T - \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ir}^{(x)^T}\right] & j \neq r \end{cases} \tag{4.26}$$

Where,
$\mathbf{a} = \bar{n}_{ij} n_i \bar{\mathbf{m}}_{ij}^{(x)} - \bar{n}_i n_{ij}\mathbf{m}_{ij}^{(x)}$ and $\mathbf{b} = \bar{n}_{ir} n_i \bar{\mathbf{m}}_{ir}^{(x)} - \bar{n}_i n_{ir}\mathbf{m}_{ir}^{(x)}$
The derivation of $\mathbf{S'}_W$ and $\mathbf{S'}_{jr}$ in this case is the same as in Appendix-D, if we set $c = c'$. Hence, no separate derivation is included.

D) Updating the between-class scatter matrix- The between-class scatter is recalculated as,

$$\mathbf{S'}_B = \sum_{i=1}^{c'} n'_i(\mathbf{m'}_i - \mathbf{m'})(\mathbf{m'}_i - \mathbf{m'})^T \tag{4.27}$$

and the reformulated between-class scatter is,

$$\mathbf{D'}_{jr} = \sum_{i=1}^{c'}\frac{n'_{ij}\, n'_{ir}}{n'_i}\,\mathbf{m'}_{ij}^{(x)}\mathbf{m'}_{ir}^{(x)^T} - \frac{1}{n'}\left(\sum_{i=1}^{c'} n'_{ij}\,\mathbf{m'}_{ij}^{(x)}\right)\left(\sum_{i=1}^{c'} n'_{ir}\,\mathbf{m'}_{ir}^{(x)}\right)^T \tag{4.28}$$

After the completion of all four steps, we add the new classes in the set of existing classes ($\mathcal{C}$).

It is possible to obtain the formulations of *chunk increment and existing class* if we set $c' = c$ in the formulations in Section 4.1.1.4. We can also obtain the sequential increment formulations for both cases by setting $l = 1$ in the two chunk increment formulations. As these formulations are inherently similar to the batch MvDA, we can derive the batch MvDA formulations from MvIDDA formulations. This is based on the base requirement of incremental methods that the discriminant spaces obtained by the incremental method must be identical to that of the batch method.

4.1.2 Deletion of Data Samples

The decremental formulation of MvIDDA has two cases:

- **Sequential decrement**- To be used when only one data sample is deleted at a time.
- **Chunk decrement**- To be used when a group of data samples is deleted at a time.

Both of these cases follow four steps that update- (i) number of data samples, (ii) means, (iii) within-class scatter matrix, and (iv) between-class scatter matrix, respectively. Once these entities are updated, the new scatter matrices are used to find optimal projection vectors using Eq.(2.5).

4.1.2.1 Sequential Increment

Let us suppose only one data sample is deleted from the existing set at a time and it belongs to class E. Let us denote this data sample by $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ and $\bar{\mathbf{x}}_j \in \mathbb{R}^{p_j}$. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_j = \mathbf{W}_j^T \bar{\mathbf{x}}_j | j = 1, \cdots, v\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples- We have to update the number of total data samples (n), the number of data samples in class E (n_E), and the number of data samples per class per view of class E (n_{Ej}) to reflect the deletion of this data sample. Here, n_{ij} and n_i for other classes $i \in \mathcal{C} - \{E\}$ do not change.

$$n'_{Ej} = n_{Ej} - 1, \;\; n'_E = n_E - v \text{ and } \;\; n' = n - v$$

B) Updating the means- As the data sample to be deleted belongs to class E, we update the means per class per view ($\mathbf{m}_{Ej}{}^{(x)}$) and class mean ($\mathbf{m}_E$) of class E. The means of remaining classes do not need updating as these classes were not updated during the decrement. We also update the total mean ($\mathbf{m}$).

$$\mathbf{m}'_{Ej}{}^{(x)} = \frac{n_{Ej}\mathbf{m}_{Ej}{}^{(x)} - \bar{\mathbf{x}}_j}{n_{Ej} - 1} \tag{4.29}$$

$$\mathbf{m}'_E = \frac{n_E\mathbf{m}_E - v\bar{\mathbf{m}}_E}{n_E - v} = \frac{n_E\mathbf{m}_E - \sum_{j=1}^{v}\bar{\mathbf{y}}_j}{n_E - v} \tag{4.30}$$

$$\mathbf{m}' = \frac{n\mathbf{m} - v\bar{\mathbf{m}}}{n - v} = \frac{n\mathbf{m} - \sum_{j=1}^{v}\bar{\mathbf{y}}_j}{n - v} \tag{4.31}$$

C) Updating the within-class scatter matrix- As the within-class scatter of classes other than E does not change, we only update the scatter of class E. The scatters of rest of the classes do not need recomputing.

$$\begin{aligned}\mathbf{S}'_W &= \sum_{i=1, i\neq E}^{c} \mathbf{S}_{W_i} + \mathbf{S}'_{W_E} \\ &= \mathbf{S}_W + \frac{v\ n_E}{(n_E - v)} (\bar{\mathbf{m}}_E - \mathbf{m}_E)(\bar{\mathbf{m}}_E - \mathbf{m}_E)^T - \sum_{j=1}^{v} \left(\bar{\mathbf{y}}_j - \mathbf{m}_E\right)\left(\bar{\mathbf{y}}_j - \mathbf{m}_E\right)^T\end{aligned} \tag{4.32}$$

and the reformulation of above equation is,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jr} + \frac{n_E}{v(n_E - v)} \left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)^T - \bar{\mathbf{x}}_j \bar{\mathbf{x}}_j^T + \frac{1}{v}\bar{\mathbf{x}}_j \bar{\mathbf{x}}_j^T & j = r \\ \mathbf{S}_{jr} + \frac{n_E}{v(n_E - v)} \left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_r - \mathbf{m}_{Er}{}^{(x)}\right)^T + \frac{1}{v}\bar{\mathbf{x}}_j \bar{\mathbf{x}}_r^T & j \neq r \end{cases} \tag{4.33}$$

The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ is similar to that presented in Appendix-B.

D) Updating the between-class scatter matrix- The between-class scatter depends on the total mean and class means. Hence, it has to be recalculated as the total mean and the class mean of class E are updated after the decrement.

$$\begin{aligned}\mathbf{S}'_B &= \sum_{i=1, i\neq E}^{c} n_i(\mathbf{m}_i - \mathbf{m}')(\mathbf{m}_i - \mathbf{m}')^T + n'_E(\mathbf{m}'_E - \mathbf{m}')(\mathbf{m}'_E - \mathbf{m}')^T \\ &= \sum_{i=1}^{c} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T\end{aligned} \tag{4.34}$$

and the reformulated between-class scatter is,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c} \frac{n'_{ij}\ n'_{ir}}{n'_i}\ \mathbf{m}'_{ij}{}^{(x)} \mathbf{m}'_{ir}{}^{(x)T} - \frac{1}{n'}\left(\sum_{i=1}^{c} n'_{ij}\ \mathbf{m}'_{ij}{}^{(x)}\right)\left(\sum_{i=1}^{c} n'_{ir}\ \mathbf{m}'_{ir}{}^{(x)}\right)^T \tag{4.35}$$

The derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ is similar to that presented in Appendix-A.

4.1.2.2 Chunk Decrement

We denote a chunk of data samples to be deleted by, $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c'; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ and $\bar{\mathbf{x}}_{ijk} \in \mathbb{R}^{p_j}$. Here, c' is the number of classes after deleting the classes from which all the data samples have been removed. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{y}}_{ijk} = \mathbf{W}_j^T \bar{\mathbf{x}}_{ijk} | i = 1, \cdots, c'; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ be the projection of $\bar{\mathcal{X}}$ onto the common discriminant space.

A) Updating the number of data samples-

$$n'_{ij} = n_{ij} - \bar{n}_{ij},\ \ n'_i = n_i - \bar{n}_i \text{ and } \ \ n' = n - \bar{n}$$

for $i = 1, \cdots, c'$ and $j = 1, \cdots, v$.

B) Updating the means-

$$\mathbf{m}'_{ij}{}^{(x)} = \frac{n_{ij}\mathbf{m}_{ij}{}^{(x)} - \bar{n}_{ij}\bar{\mathbf{m}}_{ij}{}^{(x)}}{n_{ij} - \bar{n}_{ij}} = \frac{n_{ij}\mathbf{m}_{ij}{}^{(x)} - \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}}{n_{ij} - \bar{n}_{ij}} \tag{4.36}$$

$$\mathbf{m}'_i = \frac{n_i\mathbf{m}_i - \bar{n}_i\bar{\mathbf{m}}_i}{n_i - \bar{n}_i} = \frac{n_i\mathbf{m}_i - \sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n_i - \bar{n}_i} \tag{4.37}$$

$$\mathbf{m}' = \frac{n\mathbf{m} - \bar{n}\bar{\mathbf{m}}}{n - \bar{n}} = \frac{n\mathbf{m} - \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}}{n - \bar{n}} \tag{4.38}$$

C) Updating the within-class scatter matrix- The within-class scatter of the existing and the new classes to which new data samples were added is updated. Hence, we have,

$$\mathbf{S}'_W = \mathbf{S}_W - \sum_{i=1}^{c'}\left[\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}}\left(\bar{\mathbf{y}}_{ijk}\bar{\mathbf{y}}_{ijk}^T - \bar{\mathbf{y}}_{ijk}\bar{\mathbf{m}}_i{}^T\right) - \frac{\bar{n}_i n_i}{(n_i - \bar{n}_i)}\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T\right] \tag{4.39}$$

and the reformulation of above equation is,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jj} - \sum\limits_{i=1}^{c'}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i(n_i+\bar{n}_i)^2}\mathbf{a}\mathbf{a}^T + \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ij}^{(x)^T} + \sum\limits_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T\right] & j = r \\ \mathbf{S}_{jr} - \sum\limits_{i=1}^{c'}\left[\frac{v\bar{n}_i + n_i}{\bar{n}_i n_i(n_i+\bar{n}_i)^2}\mathbf{a}\mathbf{b}^T + \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ir}^{(x)^T}\right] & j \neq r \end{cases} \tag{4.40}$$

Where,
$\mathbf{a} = \bar{n}_{ij}n_i\bar{\mathbf{m}}_{ij}^{(x)} - \bar{n}_i n_{ij}\mathbf{m}_{ij}^{(x)}$ and $\mathbf{b} = \bar{n}_{ir}n_i\bar{\mathbf{m}}_{ir}^{(x)} - \bar{n}_i n_{ir}\mathbf{m}_{ir}^{(x)}$
The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ in this case is similar to that presented in Appendix-D, if we set $c = c'$.

D) Updating the between-class scatter matrix- The between-class scatter is recalculated as,

$$\mathbf{S}'_B = \sum_{i=1}^{c'} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \tag{4.41}$$

and the reformulated between-class scatter is,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c'}\frac{n'_{ij}\ n'_{ir}}{n'_i}\ \mathbf{m}'_{ij}{}^{(x)}\mathbf{m}'_{ir}{}^{(x)^T} - \frac{1}{n'}\left(\sum_{i=1}^{c'} n'_{ij}\ \mathbf{m}'_{ij}{}^{(x)}\right)\left(\sum_{i=1}^{c'} n'_{ir}\ \mathbf{m}'_{ir}{}^{(x)}\right)^T \tag{4.42}$$

After the completion of all four steps, we update the number of classes to reflect the changes in the dataset.

4.2 Experiments and Results

4.2.1 Experimental Setup

We use three 1D datasets- Handwritten Digits, Caltech-7, and AwA- for our experiments. The datasets are divided randomly into two parts for the experiments. The first part contains 80% of the total data samples and is used to simulate the increments for training. The second part is the test set that contains 20% of the total data samples.

We train two separate models, ϕ_A and ϕ_B, using MvIDDA and batch MvDA method, respectively. For sequential increment/decrement, one training data sample is added/removed at a time. In the case of increment, MvIDDA updates ϕ_A to include the newly added sample using sequential increment formulations (refer Algorithm 1) and computes the new projection vectors. At the same time, batch MvDA discards ϕ_B to recompute the same on all of the $(n+1)$ samples together. Same is the case with decrements as well.

For chunk increment/decrement, MvIDDA handles the fixed as well as varying chunk sizes. However, to demonstrate the effect of the chunk size, we perform four separate experiments by taking chunks of 50, 100, 150, and 200 data samples, respectively. Similar to the Sequential increment, MvIDDA updates ϕ_A using chunk increment/decrement formulations (refer Algorithm 1) to include $\bar{n}$ new data samples in each update. At the same time, batch MvDA recalculates ϕ_B altogether for all of the $(n+\bar{n})$ samples, where $\bar{n}$ is the chunk size.

At the end of the training process we obtain the projection matrices, $\mathbf{W}_A^{opt}$ and $\mathbf{W}_B^{opt}$ from the final models ϕ_A and ϕ_B respectively. These projection matrices are used to project each test sample $\mathbf{y}^{Test}$ onto the common discriminant subspace. We compute the distance of $\mathbf{y}^{Test}$ from each class mean in the projected space as follows.

$$d(\mathbf{y}^{Test}, \mathbf{m}_i) = \left\| \mathbf{y}^{Test} - \mathbf{m}_i \right\|_2 \tag{4.43}$$

for $i = 1, 2, \cdots, c$.
$\mathbf{y}^{Test}$ is then assigned the label of the class to which it is found to be closest. i.e.,

$$label(\mathbf{y}^{Test}) = \underset{i}{argmin} \;\; d(\mathbf{y}^{Test}, \mathbf{m}_i) \tag{4.44}$$

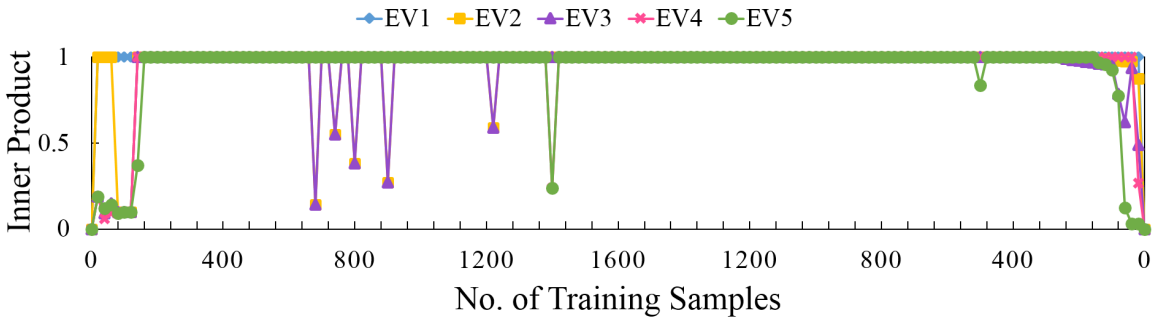
4.2.2 Results

This section presents the results of experiments designed in order to answer the research questions posed in chapter 1.

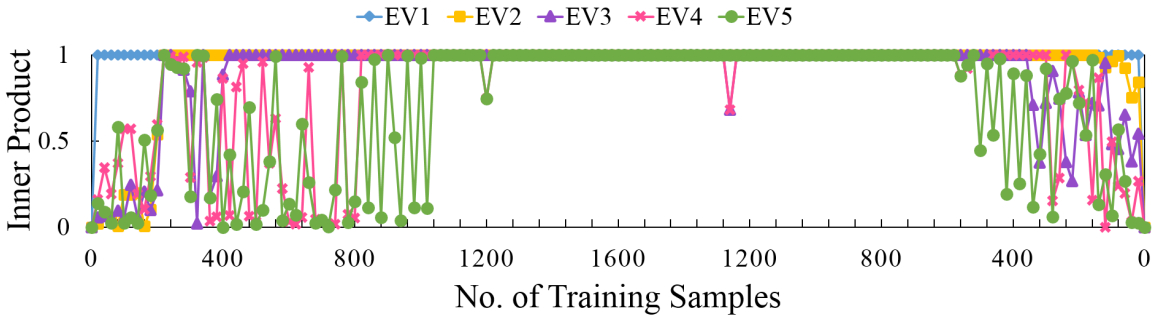
4.2.2.1 Discriminative Subspace

- **RQ1 :** Can MvIDDA, without using old data samples, produce an identical model to its batch counterpart MvDA?

- **Experiment :** We trained two models with MvIDDA and batch MvDA, respectively. If the projection vectors obtained from both methods are in agreement with each other, the subspace, and in turn, the classification accuracy is also the same. Hence, we computed the inner product of the first five eigenvectors obtained from both methods. As the eigenvectors are of unit length, we say two eigenvectors are equal when their inner product is 1.



(a) Handwritten digits dataset : Sequential increment and decrement



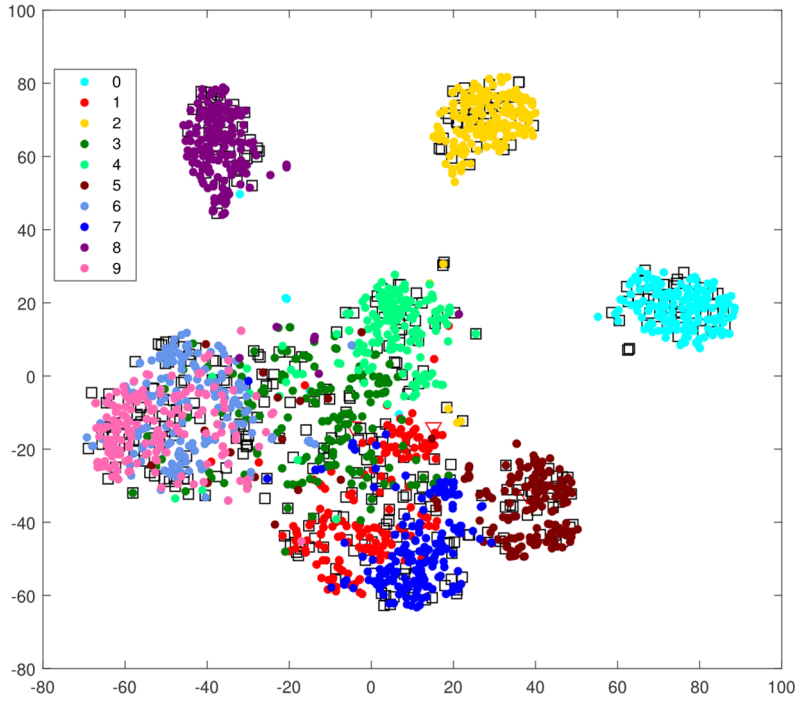
(b) Handwritten digits dataset : Chunk increment and decrement

Figure 4.2: Inner products of first five eigenvectors of MvIDDA and batch MvDA for handwritten digits dataset.

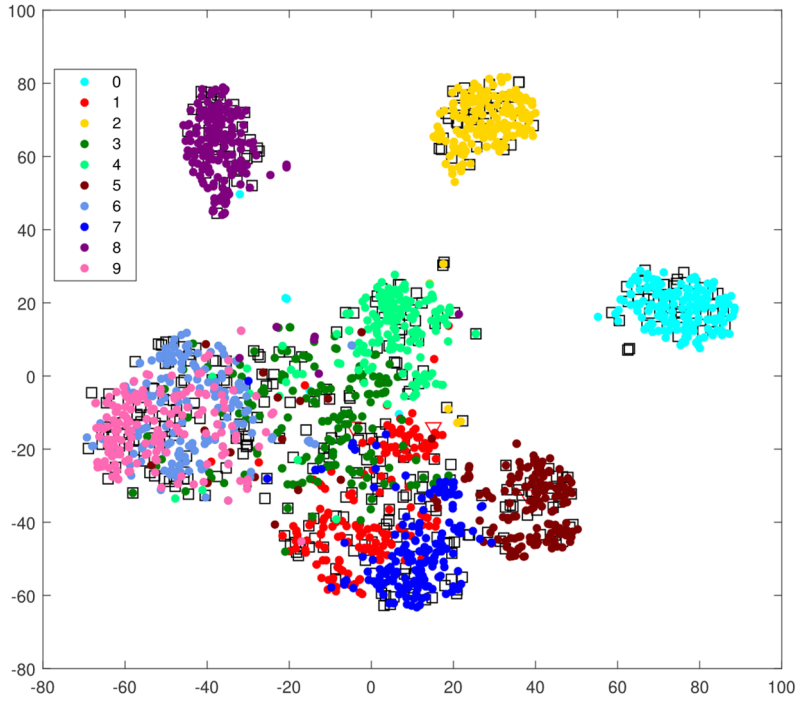
- **Discussion :** Fig.4.2 shows the inner product of the first five eigenvectors of batch MvDA and MvIDDA computed on the Handwritten Digits dataset. These figures show the inner products for increment from 0 to 1600 data samples and then decrement from 1600 to 0 data samples. The other two datasets also produce similar results and hence, are omitted. We see that the inner product of the first two eigenvectors of batch MvDA and MvIDDA converges to 1 very early. For sequential increment, as only one data point is added at each increment, all of the eigenvectors converge faster with little to no changes. In the case of the chunk increment, a group of data samples is added in each increment. Hence, inner products of eigenvectors 3 to 5 see some fluctuations in the early stages of the progress before converging gradually to 1. This shows that the common discriminative subspace constructed by the proposed method evolves over the increments to achieve the same subspace as the batch method.

  In the case of decrement, we see that the inner product stays converged to 1 when there are enough data samples. Towards the end, however, we start to see some disagreement in $EV5$ and then gradually other eigenvectors start to diverge too. Here also we see that for sequential decrement, the eigenvectors diverge much later than for chunk decrement. The agreement between the eigenvectors of both the methods shows that the discriminant subspace updated by the decremental MvIDDA is the same as that of batch MvDA for both types of increments.

  Fig. 4.3 presents a t-SNE plot [85] of data samples in the projected space constructed by MvIDDA and MvDA for handwritten digits dataset. Here, 0 to 9 are class labels that correspond to the digits from 0 to 9. We see that both the methods have formed the same projection space and hence have the same classification accuracy.



(a) MvIDDA



(b) MvDA

Figure 4.3: A plot of data samples of the handwritten digits dataset in the projected space constructed by MvIDDA and MvDA. Training data samples from different classes are shown in different colors. Correctly classified test samples are shown by black squares and incorrectly classified test samples are shown by red triangles.

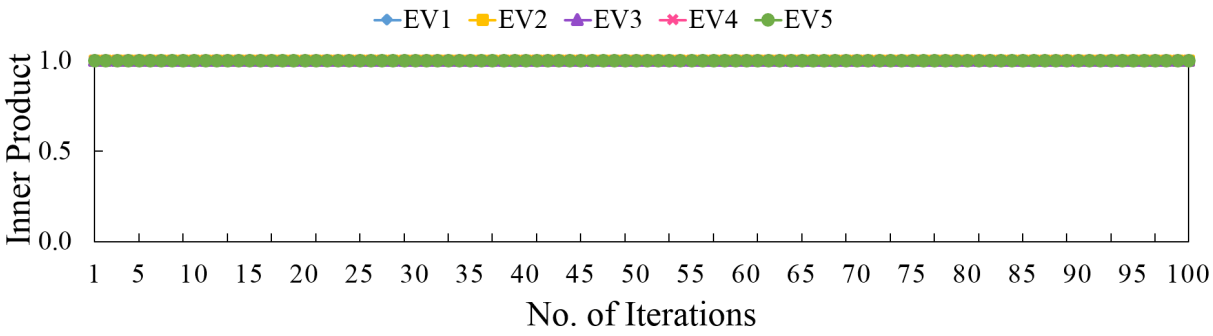


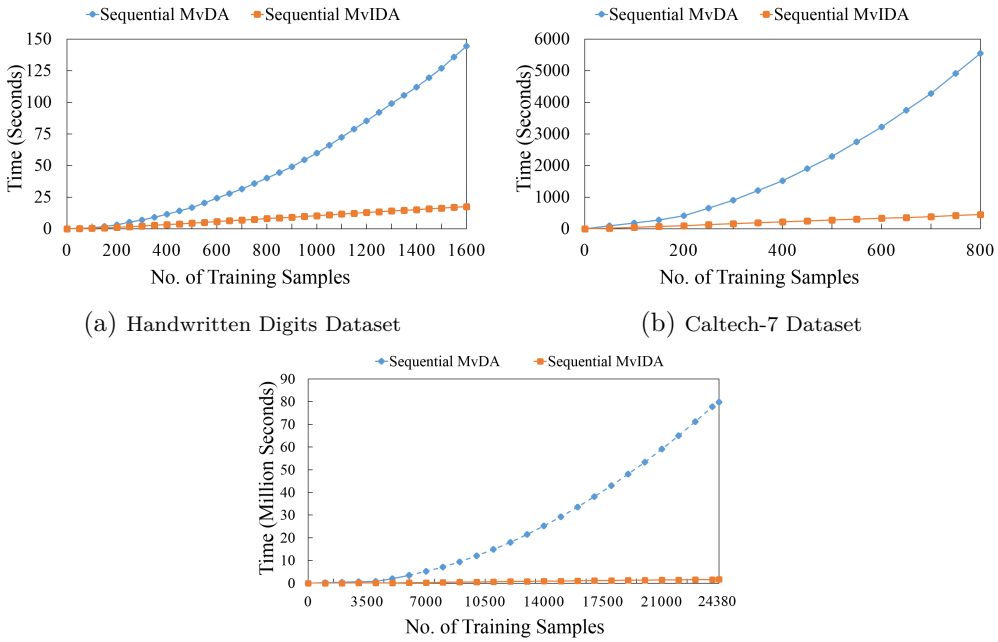
Figure 4.4: Inner product of the first five eigenvectors of every iteration for handwritten digits dataset

4.2.2.2 Order Independence

- **RQ2 :** Is this incremental method invariant to the order of addition of new data samples?
- **Experiment :** We performed 100 iterations of MvIDDA by adding the data samples in randomized order every time. In the end, we compare the first five eigenvectors of all the iterations with that of batch MvDA.
- **Discussion :** Similar to the above experiment, the inner product of the projection vectors is 1 if the discriminant space constructed by all iterations is the same. We have included the results on the Handwritten Digits dataset here in Fig.4.4. Here, graphs of the eigenvectors overlap as all the values have converged to 1, proving the order independence of the proposed method.

4.2.2.3 Training Time

- **RQ3 :** Can this incremental method reduce training time?
- **Experiment :** We record the time taken by each method at the intervals of 50 data points for *sequential* and *chunk-50*. For *chunk-100*, *chunk-150* and *chunk-200* the time is recorded at intervals of 100, 150 and 200 data samples respectively. Here, the time records for MvIDDA consist of the time taken for updating the four entities, namely- the number of data samples, the means, the within-class scatter and the between-class scatter. Similarly, the time records for MvDA consist of the time taken for recomputing these four entities. We have not considered the time taken for the computation of the projection vectors, as this step is common for both the methods.
- **Discussion :** It is intuitive for MvIDDA to require less time than batch MvDA, which is reflected in Fig. 4.5-4.6. Note that for the AwA dataset, the unit of time is million seconds and the markers are placed sparsely for better viewing. Fig.4.5 shows the time comparison between MvIDDA and batch MvDA for sequential increment on all four datasets. We see that MvIDDA requires very less training time compared to batch MvDA. As the number of samples increases, the difference in the time grows larger. Sequential MvIDDA took nearly 20 days to complete training on the AwA dataset. However, the batch MvDA is estimated to require around 912 days to complete the same. The dashed part of the Sequential MvDA curve in Fig.4.7c shows the estimated training time.

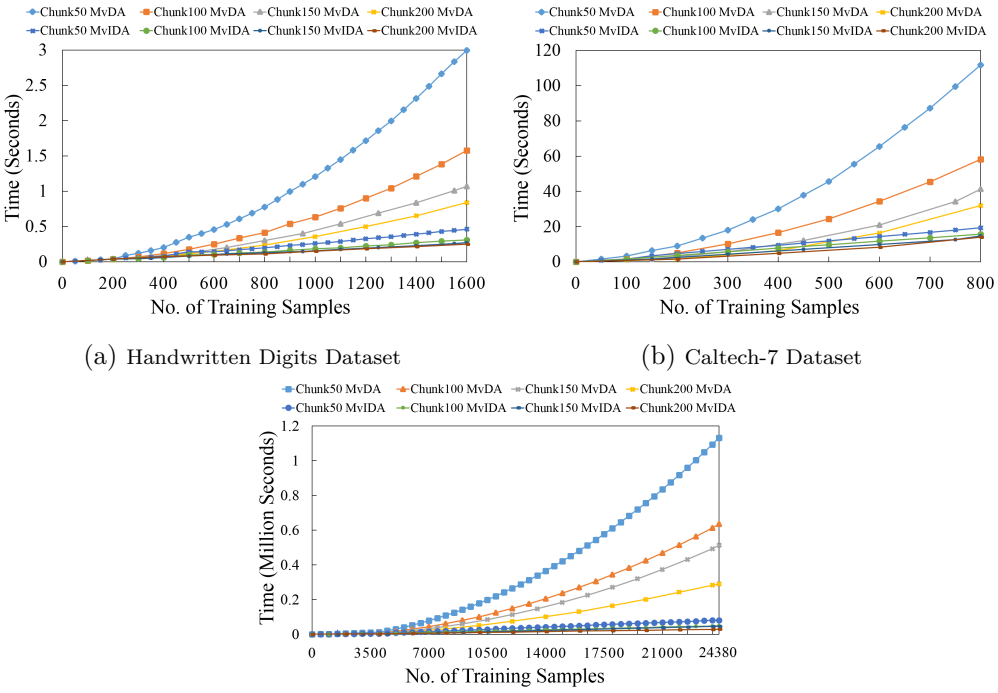


(a) Handwritten Digits Dataset

(b) Caltech-7 Dataset

(c) AwA Dataset

Figure 4.5: Comparison of training time of MvIDDA and batch MvDA : sequential increment

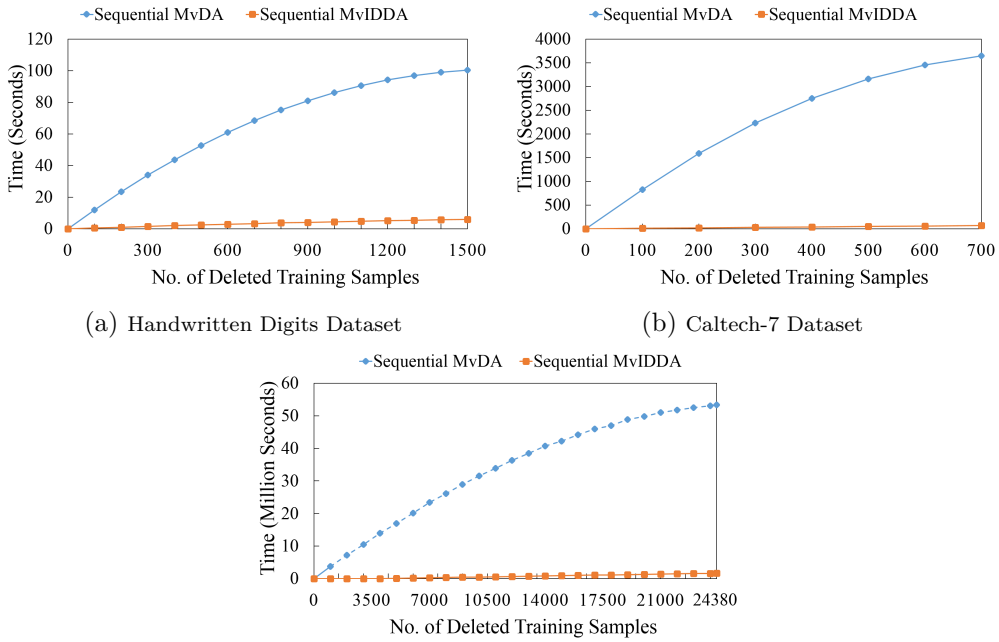


(a) Handwritten Digits Dataset

(b) Caltech-7 Dataset

(c) AwA Dataset

Figure 4.6: Comparison of training time of MvIDDA and batch MvDA : chunk increment

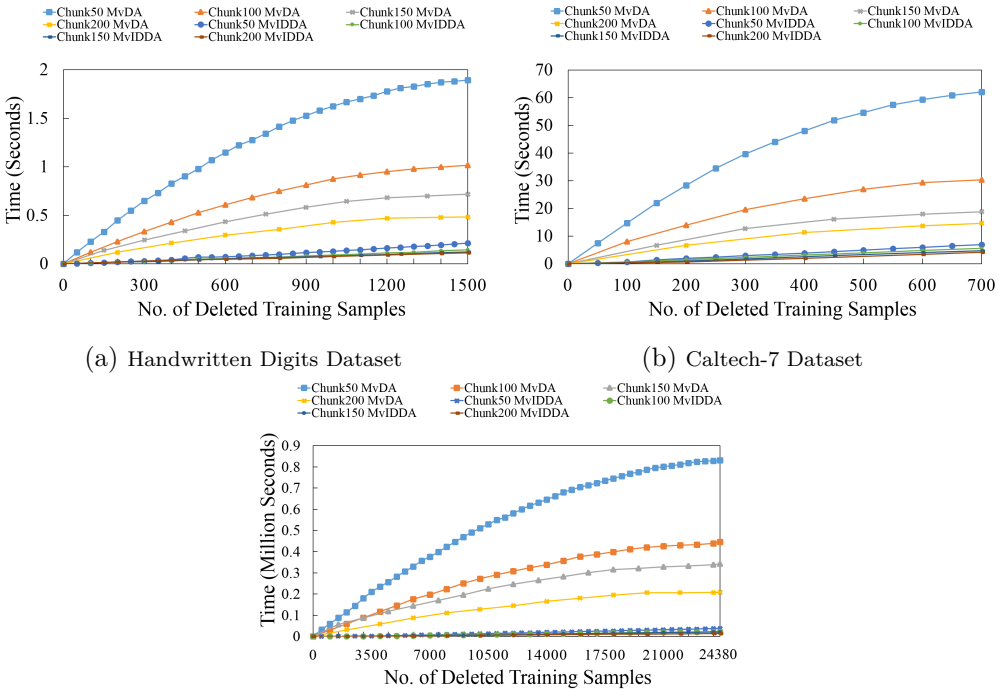


(a) Handwritten Digits Dataset

(b) Caltech-7 Dataset

(c) AwA Dataset

Figure 4.7: Comparison of training time of MvIDDA and batch MvDA : sequential decrement



(a) Handwritten Digits Dataset

(b) Caltech-7 Dataset

(c) AwA Dataset

Figure 4.8: Comparison of training time of MvIDDA and batch MvDA : chunk decrement

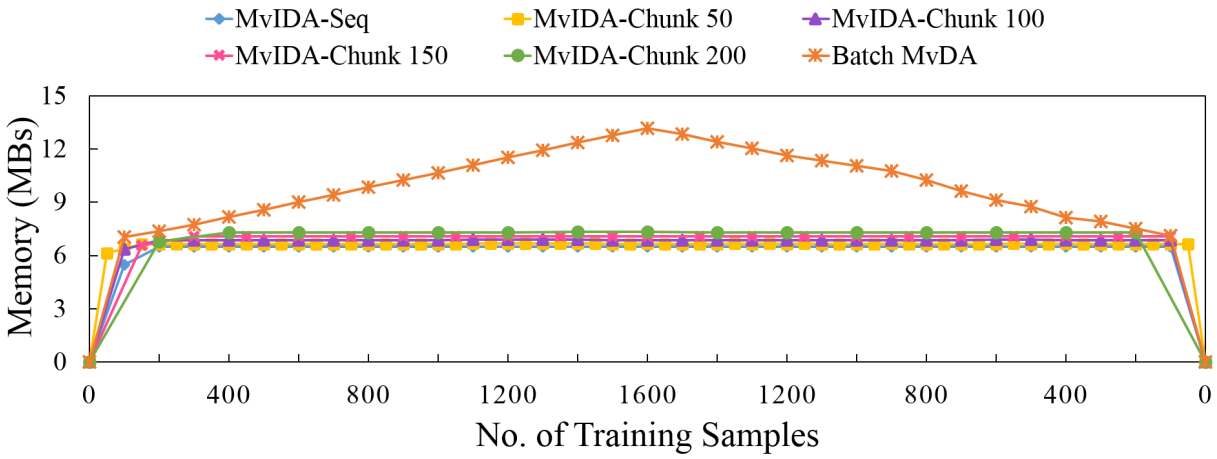


Figure 4.9: Memory usage comparison for handwritten digits dataset

Fig. 4.6 shows the time comparison between MvIDDA and batch MvDA for chunk increment. It shows training time for each of the four chunk sizes (50, 100, 150, and 200) for both methods. The time course of MvIDDA is represented by four lines at the bottom of the graph that are closer to each other. In this case also, we observe that the time required by batch MvDA sees a faster growth and is much higher than that of the MvIDDA.

Fig. 4.7 and 4.8 show the records of time taken by the decremental MvIDDA against batch MvDA. We see that the decremental MvIDDA takes much less time as it only updates the model to reflect the changes after removing some data samples. Whereas, batch MvDA discards the model and trains on all the remaining data samples after the removal. This shows the advantage of using decremental MvIDDA.

4.2.2.4 Memory Usage

- **RQ4 :** Can this incremental method reduce memory requirements?

- **Experiment :** We record the memory requirements of MvIDDA at the intervals of 50 data points for *sequential* and *chunk-50*. For *chunk-100*, *chunk-150* and *chunk-200* the time is recorded at intervals of 100, 150 and 200 data samples respectively. Memory requirements of batch MvDA are recorded by taking chunks of 100 data samples at a time.

- **Discussion :** The memory usage comparison between these methods for the Handwritten Digits dataset is shown in Fig. 4.9. We see that the memory requirement of the sequential and the chunk MvIDDA is less than that of the batch learning method. The sequential MvIDDA requires almost constant memory as it stores only the model ϕ and one new data sample. Chunk MvIDDA requires more space as, along with the model, it needs space for the new chunk of the data. The memory requirements of chunk increment vary according to the chunk size. The memory requirements stay the same for decrements as well.

  Batch MvDA, on the other hand, stores the model ϕ along with all the old and new data. Hence, the storage requirements are high and increase with the increments and decrease with the decrements in the data.

Table 4.1: Comparison of accuracy and training time : MvIDDA vs. single-view ILDA

Dataset	Accuracy (%)		Total Training Time (Seconds)			
			Chunk-100		Sequential	
	MvIDDA	ILDA	MvIDDA	ILDA	MvIDDA	ILDA
Handwritten Digits	99.00	52.75	0.40	20.42	22.44	69.60
Caltech-7	97.00	66.50	19.41	564.70	465.72	1117.53
AwA	96.55	81.62	49506.07	130043.42	1666955.69	1811942.66

4.2.2.5 Comparison with single-view ILDA

- **RQ5 :** Is this multi-view incremental method more advantageous than a single-view incremental method in terms of classification accuracy and training time?
- **Experiment :** As ILDA is a single-view incremental method, to use it on multi-view data, we concatenate all the views together to form a single view and then apply ILDA on it. We record the training time and classification accuracy for sequential and chunk-100 increments using both methods.
- **Discussion :** The results in Table 4.1 show the importance of using a multi-view method for multi-view data. MvIDDA processes the views separately and provides far better classification results than ILDA. As all the views were concatenated together, ILDA could not use the discriminatory information provided by different views. It instead weighed all the views on the same scales, leading to misinformation and low classification accuracy. MvIDDA also requires less training time than ILDA.

4.3 Summary

We have proposed Multi-view Incremental Decremental Discriminant Analysis (MvIDDA), an incremental method for multi-view data in the dimensionality reduction paradigm. Two approaches, namely sequential MvIDDA and chunk MvIDDA, have been proposed to incrementally construct a common discriminant space for multi-view data.

We show that MvIDDA achieves a common discriminant subspace identical to the one constructed by batch MvDA. Hence, their classification accuracy is also the same. The classification accuracy increases with the number of views. We also show that though it produces the same results, MvIDDA requires much less training time and memory than batch MvDA. Also, MvIDDA is invariant to the input order of the data and handles data samples from new classes at any stage of training. MvIDDA also requires less training time than the single-view incremental method ILDA and produces better classification results. We see that chunk MvIDDA requires less time than sequential MvIDDA, whereas the latter requires less memory and converges to the optimum in the early stages of training. The incremental formulations of this work named as Multi-view Incremental Discriminant Analysis (MvIDA) are published in [86].

The next chapter presents a batch classification method for 2D multi-view data. This method forms a basis for an incremental method in the 2D learning paradigm.

5

2D Multi-view Discriminant Analysis

This chapter presents a batch classification method for 2D multi-view data. Using 1D methods to train on naturally 2D data leads to the loss of valuable spatial information. Also, the vectorization leads to larger scatter matrices. The use of 2D methods for 2D data preserves spatial information and eliminates the need for vectorization. Hence, we propose a 2D batch method for multi-view data to train a classification model in less time and memory.

The proposed method, 2D Multi-view Discriminant Analysis (2DMvDA), is designed to serve as a basis for an incremental-decremental 2D method presented in Chapter 6. 2DMvDA has a very straightforward formulation. However, the main idea is to introduce 2D methods to the realm of multi-view learning. It utilizes the matrix form of images to preserve spatial information and reduce the size of scatter matrices for better and faster performance. 2DMvDA also eliminates the need for feature extraction in preprocessing stage.

We present the formulations of 2DMvDA followed by the experimental setup and results. We conclude this chapter by summarizing the features of 2DMvDA.

5.1 Methodology

2DMvDA works on the same principle of discriminant analysis as MvDA. The within-class scatter and the between-class scatter are defined as follows,

$$\mathbf{S}_W = \sum_{i=1}^{c} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk} - \mathbf{M}_i\right) \left(\mathbf{Y}_{ijk} - \mathbf{M}_i\right)^T \tag{5.1}$$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i \left(\mathbf{M}_i - \mathbf{M}\right) \left(\mathbf{M}_i - \mathbf{M}\right)^T \tag{5.2}$$

Note that these equations are similar to those of MvDA. However, here we compute the scatter matrices using 2D data matrices in contrast to 1D feature vectors used by MvDA. Eq. 5.1 and Eq. 5.2 are in terms of the data samples in projected space ($\mathbf{Y}_{ijk}$). We reformulate these equations in terms of the original data samples ($\mathbf{X}_{ijk}$). The reformulated within-class scatter is given as,

$$\mathbf{S}_{jr} = \begin{cases} \sum_{i=1}^{c} \left(\sum_{k=1}^{n_{ij}} \mathbf{X}_{ijk} \mathbf{X}_{ijk}^T - \frac{n_{ij} n_{ij}}{n_i} \mathbf{M}_{ij}^{(\mathbf{X})} \mathbf{M}_{ij}^{(\mathbf{X})T} \right) & \cdots j = r \\ -\sum_{i=1}^{c} \frac{n_{ij} n_{ir}}{n_i} \mathbf{M}_{ij}^{(\mathbf{X})} \mathbf{M}_{ir}^{(\mathbf{X})T} & \cdots j \neq r \end{cases} \tag{5.3}$$

Here, $\mathbf{S}_{jr}$ is computed using the 2D matrices in the original space. The derivation for Eq. 5.3 is given in Appendix-E.
The reformulated equation for between-class scatter is given as,

$$\mathbf{D}_{jr} = \sum_{i=1}^{c} \frac{n_{ij}\ n_{ir}}{n_i}\ \mathbf{M}_{ij}^{(\mathbf{X})}\ \mathbf{M}_{ir}^{(\mathbf{X})T} - \frac{1}{n}\left(\sum_{i=1}^{c} n_{ij}\ \mathbf{M}_{ij}^{(\mathbf{X})}\right)\left(\sum_{i=1}^{c} n_{ir}\ \mathbf{M}_{ir}^{(\mathbf{X})}\right)^{T} \tag{5.4}$$

Here, $\mathbf{D}_{jr}$ is computed using the 2D matrices in the original space. The derivation for Eq. 5.4 is given in Appendix-A.

We convert Eq. 2.1 into a ratio trace function so that it can be solved as a generalized eigenvalue problem. The reformulated optimal function is,

$$\mathbf{W}^{opt} = \left(\mathbf{W}_1^{opt}, \cdots, \mathbf{W}_v^{opt}\right) = \underset{(\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_v)}{argmax}\ Tr\left(\frac{\mathbf{W}^T\mathbf{D}\mathbf{W}}{\mathbf{W}^T\mathbf{S}\mathbf{W}}\right) \tag{5.5}$$

The above equation can now be solved using the Generalized eigenvalue decomposition method as,

$$\mathbf{D}\mathbf{W}^{opt} = \lambda\mathbf{S}\mathbf{W}^{opt} \tag{5.6}$$

We then form $\mathbf{W}^{opt}$, a matrix of optimal projection vectors, by selecting the eigenvectors corresponding to the first d largest eigenvalues of $\mathbf{S}^{-1}\mathbf{D}$. This projection matrix is then used to project the test samples onto a common discriminant subspace. The projection of a test sample of size $p_j \times q$, will be of size $d \times q$ in the common discriminant subspace.

5.2 Experiments and Results

5.2.1 Experimental Setup

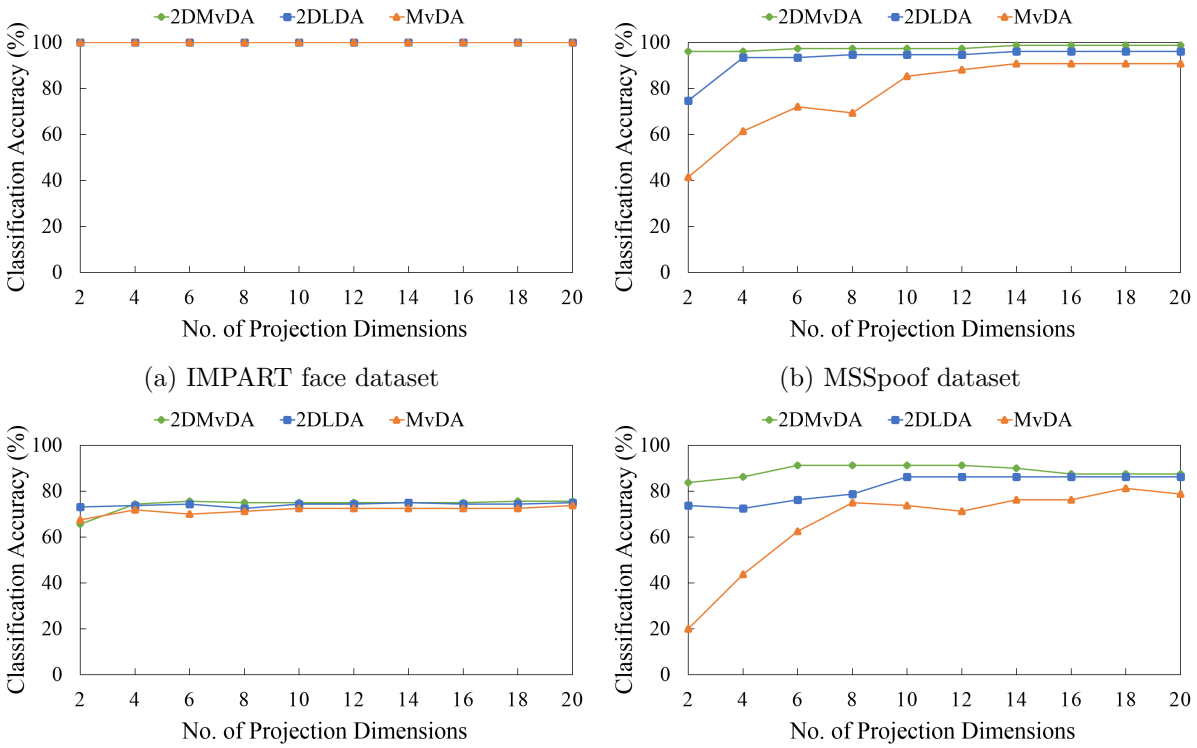
MvDA, being a 1D method, needs the training data in vectorized format. However, due to the large images in these datasets, MvDA needed more than 100 Terabytes of processing memory. Hence, we used rescaled versions of datasets to train the models ϕ_A, ϕ_B and ϕ_C using 2DMvDA, 2DLDA, and MvDA, respectively.

In another set of experiments, we used original versions of the datasets mentioned above to show the full strength of 2DMvDA. This time, we trained the models only on 2DMvDA and 2DLDA. We have divided each dataset into two parts for both sets of experiments. 80% of the total data samples are used to train the models. The remaining 20% data samples are used as a test set.

To compare the three methods, we record the time and memory required by these methods to train classification models. Also, to compare the classification accuracy of these models, we extract 10 sets of eigenvectors corresponding to first d largest eigenvalues where, $d = \{i * 2 | i = 1, \cdots, 10\}$. These eigenvectors form the projection matrix $\mathbf{W}^{opt}$ and are used to project each test data sample ($\mathbf{Y}^{Test}$) onto the common discriminant subspace. The methodology for classification is the same as that used for MvIDDA.

5.2.2 Results

This section presents the results of experiments designed in order to answer the research questions posed in chapter 1.



(a) IMPART face dataset

(b) MSSpoof dataset

(c) Stereo face dataset

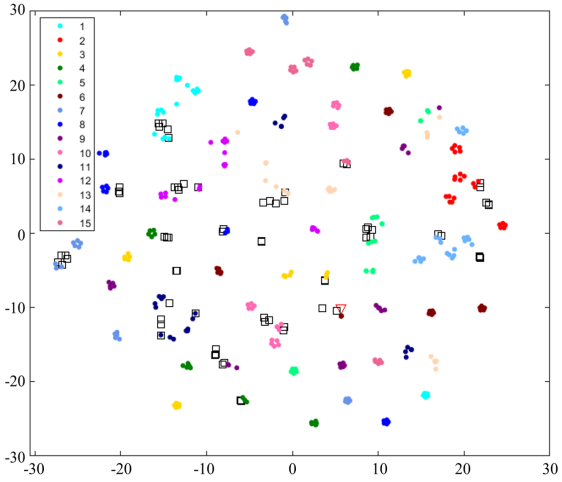
(d) ORL dataset

Figure 5.1: Classification accuracy vs. number of projection dimensions (On rescaled datasets)

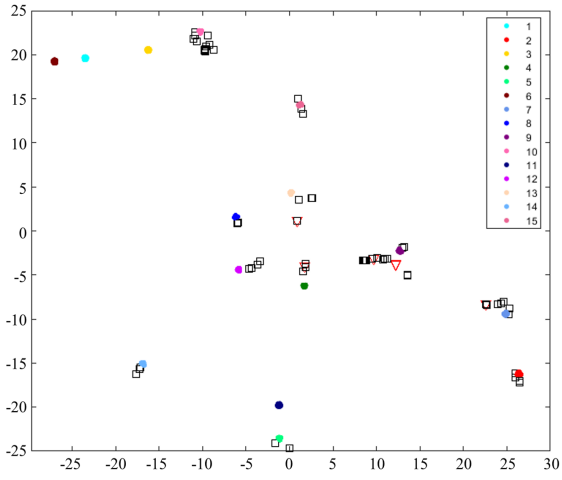
5.2.2.1 A. Classification Accuracy

- **RQ1 :** Can a 2D multi-view method build a better classification model than a 1D multi-view method or a 2D single-view method?
- **Experiment :** Record classification accuracy of the methods by varying the no. of projection dimensions $d = \{i * 2 | i = 1, \cdots, 10\}$.
- **Discussion :** Fig. 5.1 compares the accuracies of these three methods. We see that the classification accuracy of 2DMvDA is better than or the same as that of the other two methods. 2DMvDA performs much better than MvDA even in lower-dimensional space as the former extracts the 2D features that provide more information than the 1D features. We also see that 2DMvDA performs at par or better than 2DLDA, despite both being 2D methods. This is because 2DLDA considers all the views as one, depriving itself of the discriminatory information that 2DMvDA gains by processing the views separately. This proves the benefits of using two-dimensional representation and multiple views.

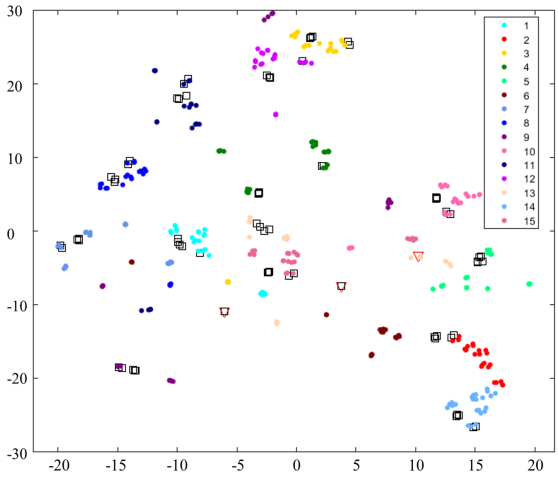
  We have plotted the classification results of these three methods on the MSSpoof dataset using t-SNE. Fig. 5.2 shows the results. Here, the training data samples from different classes are shown in different colors. The correctly classified test data samples are denoted with black squares, and those classified incorrectly are denoted with red triangles. We see that the subspace constructed by 2DMvDA and 2DLDA is better at discriminating between different classes than MvDA. This leads to the better classification accuracy of 2D methods. MvDA performs poorly as it does not take spatial information into account.



(a) 2DMvDA



(b) MvDA



(c) 2DLDA

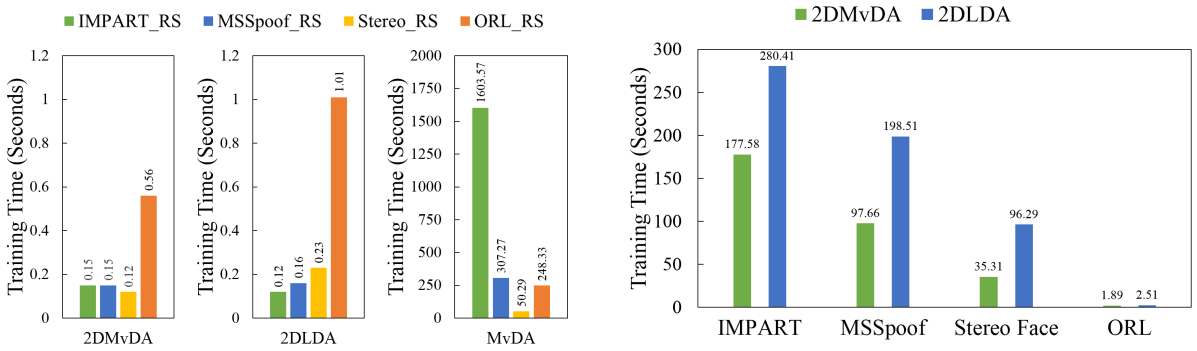
Figure 5.2: Visualization of Classification on MSSpoof Dataset: (i) Training data samples are denoted with different colors for each class. (ii) Correctly classified test samples are denoted with hollow black squares. (iii) Misclassified test samples are denoted with hollow red triangles.

5.2.2.2 Training Time

- **RQ2 :** Can the use of 2D matrices reduce training time?

- **Experiment :** Record training time of (i) all three methods on the rescaled datasets and (ii) 2DMvDA and 2DLDA on the original datasets.

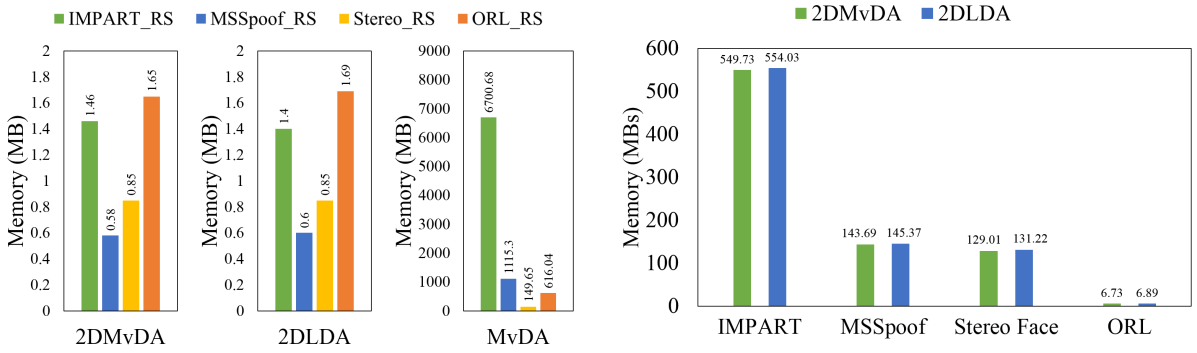
- **Discussion :** Fig. 5.3a shows the records of training time of each method on the rescaled versions of all four datasets. The suffix (RS) represents the rescaled versions of the datasets. Note that the scale on the y-axis of 2DMvDA and 2DLDA is much smaller than that of MvDA. We see that the 2DMvDA requires less than a second to train the model, whereas the 1D method -MvDA- has much greater time requirements. The records of average training time on the original versions of the datasets using 2DMvDA and 2DLDA are presented in Fig. 5.3a.



(a) On rescaled datasets

(b) On original datasets

Figure 5.3: Comparison of training time



(a) On rescaled datasets

(b) On original datasets

Figure 5.4: Comparison of memory requirement

The records of average training time of 2DMvDA and 2DLDA on original datasets are presented in Fig. 5.3b. We see that 2DMvDA trains the model in 30% to 70% of the training time of 2DLDA. The experiments using MvDA could not be performed due to the larger memory requirements of this 1D method. However, by linearly extrapolating the time requirements on the rescaled dataset, we say that MvDA would need a little over 21 days to train on the original version of the IMPART dataset. 2DMvDA, on the other hand, needs only 3 minutes on the same dataset.

5.2.2.3 Memory Usage

- **RQ3 :** Can the use of 2D matrices reduce memory requirements?
- **Experiment :** Record memory usage of (i) all three methods on the rescaled versions of datasets and (ii) 2DMvDA and 2DLDA on the original datasets.
- **Discussion :** The records of memory requirements on the rescaled version of datasets are presented in Fig. 5.4a. Note that the scale on the y-axis of 2DMvDA and 2DLDA is much smaller than that of MvDA. We see in Fig. 5.4a that though same as 2DLDA, the proposed method requires less than 0.1% of the memory required by MvDA to train the model. The

difference in memory usage of 2DMvDA against MvDA is due to the latter's need to store large scatter matrices. MvDA uses the vectorized form of images and produces much larger scatter matrices. However, 2DMvDA uses the matrix form, resulting in smaller scatter matrices. This is very advantageous when working with large datasets.

The records of average memory requirements on the original versions of the datasets using 2DMvDA and 2DLDA are presented in Fig. 5.4b. We can see that even for original datasets, the 2D methods need less memory than that of MvDA on rescaled datasets. The estimated memory requirement of MvDA for the original version of IMPART dataset is more than 2500 Terabytes, whereas 2DMvDA requires only 0.56 Gigabytes to train on the same dataset. This shows the benefit of using the 2D matrix representation instead of a vectorized form of the image matrix.

5.3 Summary

The presented classification method, 2DMvDA, is designed for multi-view 2D data. 2DMvDA constructs a common discriminant subspace directly from original 2D matrices, thereby preserving the spatial information in data and eliminating the need for feature extraction in the pre-processing stage.

The proposed method is compared with a 2D single-view method (2DLDA) and a 1D multi-view method (MvDA) for classification accuracy, training time, and memory requirements. 2DMvDA provides better classification results than the other two methods as it benefits from spatial information and multiple views. Even though the gain in classification accuracy is not high, 2DMvDA achieves it with less than 0.01% of training time and 0.0001% of memory that MvDA needs to train the same model. 2DMvDA even performs better than 2DLDA with 50% less training time on average. Also, the comparison of the classification accuracy shows that the proposed method performs better than 2DLDA and MvDA, even with fewer projection dimensions. This work is published in [87].

In the next chapter, we present an incremental-decremental method based on 2DMvDA. This method aims to introduce incremental learning in the 2D learning paradigm.

6

2D Multi-view Incremental Decremental Discriminant Analysis

In this chapter, we present an incremental-decremental classification method for 2D multi-view data. This method is based on 2DMvDA and extends it further by enabling the incremental learning and decremental unlearning of the data samples. It has similar formulations for incremental learning as that of MvIDDA. However, it presents a decremental unlearning formulation as well.

This method, 2D Multi-view Incremental Decremental Discriminant Analysis (2DMvIDDA) supports the addition or deletion of data samples without retraining the model from scratch. Like MvIDDA, this method also supports the addition of data samples from existing as well as previously unseen classes. Though the incremental formulations of 2DMvIDDA have been presented as one generalized case, they can be converted into the four specialized cases as those of MvIDDA. Fig. 6.1 shows an example of incremental learning and decremental unlearning of views.

We present the formulations of 2DMvIDDA, the experimental setup, and results. We conclude this chapter by summarizing the features of the proposed method.

6.1 Methodology

6.1.1 Addition of Data Samples

Here, we present the update formulations for increments when a set of new data samples ($\bar{\mathcal{X}}$) is presented. The size of this set need not be the same for each increment. Some data samples in $\bar{\mathcal{X}}$ may belong to new classes. Let $\bar{\mathcal{Y}} = \{\bar{\mathbf{Y}}_{ijk} = \mathbf{W}_j^T \bar{\mathbf{X}}_{ijk} | i = 1, \cdots, c'; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$ be the new set $\bar{\mathcal{X}}$ projected onto the shared subspace.

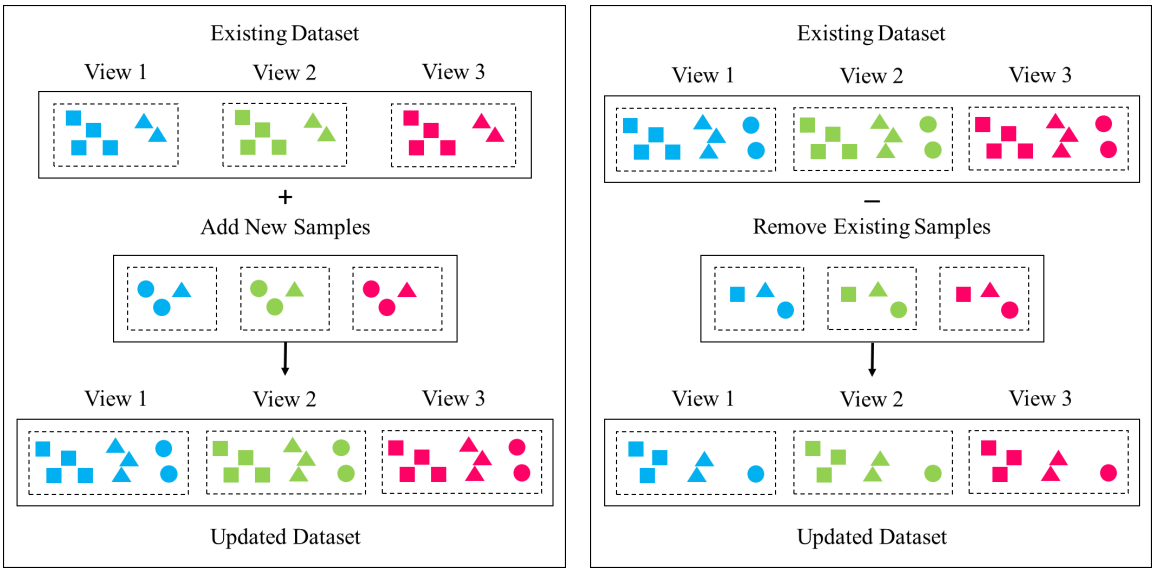
We then update the *data sample counts* and the *means* to include the new samples as follows:

6.1.1.1 Updating the data sample counts

The number of data samples per class per view, the number of data samples per class, and the total number of data samples are updated to include all the new data samples.

$$n'_{ij} = n_{ij} + \bar{n}_{ij}, \;\; n'_i = n_i + \bar{n}_i \;\text{ and }\; n' = n + \bar{n} \tag{6.1}$$

for $i = 1, \cdots, c'$ and $j = 1, \cdots, v$.



(a) Incremental learning (b) Decremental unlearning

Figure 6.1: Chunk increment and decrement : (a) A chunk of new data samples is added. Some of the new data samples belong to a new class. (b) A chunk of existing data samples is deleted from the dataset. Views are depicted with different colors (blue, green and red) and classes are depicted with different shapes (square, triangle and circle).

6.1.1.2 Updating the means

We update all the older means to reflect the change due to the addition of new data samples. The update equations are as follows-

$$\mathbf{M}'_{ij}{}^{(\mathbf{X})} = \frac{n_{ij}\mathbf{M}_{ij}{}^{(\mathbf{X})} + \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{X}}_{ijk}}{n_{ij} + \bar{n}_{ij}} = \frac{n_{ij}\mathbf{M}_{ij}{}^{(\mathbf{X})} + \bar{n}_{ij}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})}}{n'_{ij}} \tag{6.2}$$

$$\mathbf{M}'_i = \frac{n_i\mathbf{M}_i + \sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{Y}}_{ijk}}{n_i + \bar{n}_i} = \frac{n_i\mathbf{M}_i + \bar{n}_i\bar{\mathbf{M}}_i}{n'_i} \tag{6.3}$$

$$\mathbf{M}' = \frac{n\mathbf{M} + \sum_{i=1}^{c'}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{Y}}_{ijk}}{n + \bar{n}} = \frac{n\mathbf{M} + \bar{n}\bar{\mathbf{M}}}{n'} \tag{6.4}$$

We then update the within-class and the between-class scatter matrix. Here, the equations are presented in the projected space first. Then, these equations are presented in terms of the original data samples for computation.

Algorithm 2 2DMvIDDA: Incremental algorithm

Input :

- A trained model ϕ that consists of:
  - Data sample counts: n, n_i, n_{ij}
  - Mean per class per view: $\mathbf{M}_{ij}^{(\mathbf{X})}$
  - Scatter matrices: $\mathbf{S}_{jr}$, $\mathbf{D}_{jr}$
  - Set of class labels of existing data samples: $\mathcal{C}$
- Set of new data samples: $\bar{\mathcal{X}}$
- Data sample counts of new data samples: $\bar{n}$, $\bar{n}_i$, $\bar{n}_{ij}$
- Set of class labels of new data samples: $\bar{\mathcal{C}}$

while new data is encountered **do**

Update n, n_i, n_{ij} using Eq. (6.1).
Update $\mathbf{M}_{ij}^{(\mathbf{X})}$ using Eq.(6.2).
Update $\mathbf{S}_{jr}$ using Eq.(6.6).
Update $\mathbf{D}_{jr}$ using Eq.(6.8).
$\mathcal{C} = \mathcal{C} \cup \bar{\mathcal{C}}$.
Compute the projection matrix $\mathbf{W}^{opt}$ using Eq.(2.5).

end while

Output :

- Updated model ϕ
- The projection matrix $\mathbf{W}^{opt}$

6.1.1.3 Updating the within-class scatter matrix

We only need to update the scatter of those classes to which new data samples are introduced. Hence, we have,

$$\begin{aligned}\mathbf{S}'_W &= \sum_{i\in\mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i\in\mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk} - \mathbf{M}'_i\right)\left(\mathbf{Y}_{ijk} - \mathbf{M}'_i\right)^T + \sum_{i\in\mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{Y}}_{ijk} - \mathbf{M}'_i\right)\left(\bar{\mathbf{Y}}_{ijk} - \mathbf{M}'_i\right)^T \\ &= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{S}'_{jr} \mathbf{W}_r\end{aligned} \tag{6.5}$$

where,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jj} + \sum_{i=1}^{c'} \left[\frac{1}{\bar{n}_i n_i (n_i + \bar{n}_i)} \mathbf{E}\mathbf{E}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i} \bar{\mathbf{M}}_{ij}^{(\mathbf{X})} \bar{\mathbf{M}}_{ij}^{(A)^T} + \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{X}}_{ijk} \bar{\mathbf{X}}_{ijk}^T \right] & j = r \\ \mathbf{S}_{jr} + \sum_{i=1}^{c'} \left[\frac{1}{\bar{n}_i n_i (n_i + \bar{n}_i)} \mathbf{E}\mathbf{F}^T - \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i} \bar{\mathbf{M}}_{ij}^{(\mathbf{X})} \bar{\mathbf{M}}_{ir}^{(A)^T} \right] & j \neq r \end{cases} \tag{6.6}$$

here, $\mathbf{E} = \bar{n}_{ij} n_i \bar{\mathbf{M}}_{ij}^{(\mathbf{X})} - \bar{n}_i n_{ij} \mathbf{M}_{ij}^{(\mathbf{X})}$ and $\mathbf{F} = \bar{n}_{ir} n_i \bar{\mathbf{M}}_{ir}^{(\mathbf{X})} - \bar{n}_i n_{ir} \mathbf{M}_{ir}^{(\mathbf{X})}$.

The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ is given in Appendix-F. Each submatrix of S' consists of known quantities and is in closed form. This is true for the decremental formulation as well.

6.1.1.4 Updating the between-class scatter matrix

This matrix is computed using only the updated class-mean ($\mathbf{M}'_i$) and the total mean ($\mathbf{M}'$). Both the means are updated after each increment, hence the scatter matrix is updated as,

$$\mathbf{S}'_B = \sum_{i=1}^{c'} n'_i (\mathbf{M}'_i - \mathbf{M}')(\mathbf{M}'_i - \mathbf{M}')^T = \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{D}'_{jr} \mathbf{W}_r \tag{6.7}$$

where each $\mathbf{D}_{jr}$ is given as,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c'} \frac{n'_{ij}\, n'_{ir}}{n'_i}\, \mathbf{M}'_{ij}{}^{(\mathbf{X})} \mathbf{M}'_{ir}{}^{(\mathbf{X})T} - \frac{1}{n'} \left(\sum_{i=1}^{c'} n'_{ij}\, \mathbf{M}'_{ij}{}^{(\mathbf{X})} \right) \left(\sum_{i=1}^{c'} n'_{ir}\, \mathbf{M}'_{ir}{}^{(\mathbf{X})} \right)^T \tag{6.8}$$

The derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ is given in Appendix-A.

After updating the means and the scatter matrices, 2DMvIDDA computes the optimal projection matrix $\mathbf{W}^{opt}$ using Eq. 2.5. The test data samples can be classified using this projection matrix until new training data samples are presented, or any of the existing ones are deleted. If new data samples are added, these steps are repeated. The incremental learning formulation is summarized in Algorithm2.

6.1.2 Deletion of Data Samples

Here, we present the update formulations for decremental unlearning when a subset ($\bar{\mathcal{X}}$) of existing data samples is to be deleted. $\bar{\mathcal{X}}$ may have one or more data samples in it. The projection of $\bar{\mathcal{X}}$ onto the shared subspace is given as $\bar{\mathcal{Y}} = \{\bar{\mathbf{Y}}_{ijk} = \mathbf{W}_j^T \bar{\mathbf{X}}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, \bar{n}_{ij}\}$.

We update the *data sample counts* and the *means* to reflect the changes in the dataset after the decrement.

6.1.2.1 Updating the data sample counts

The number of data samples per class per view, number of data samples per class, and the total number of data samples are updated to reflect the deletion of the data samples in $\bar{\mathcal{X}}$.

$$n'_{ij} = n_{ij} - \bar{n}_{ij}, \;\; n'_i = n_i - \bar{n}_i \;\text{ and }\; n' = n - \bar{n} \tag{6.9}$$

for $i = 1, \cdots, c$ and $j = 1, \cdots, v$.

6.1.2.2 Updating the means

We update all the older means to reflect the change due to the deletion of new data samples. The update equations are as follows-

$$\mathbf{M}'_{ij}{}^{(\mathbf{X})} = \frac{n_{ij}\mathbf{M}_{ij}{}^{(\mathbf{X})} - \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{X}}_{ijk}}{n_{ij} - \bar{n}_{ij}} = \frac{n_{ij}\mathbf{M}_{ij}{}^{(\mathbf{X})} - \bar{n}_{ij}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})}}{n'_{ij}} \tag{6.10}$$

$$\mathbf{M}'_i = \frac{n_i\mathbf{M}_i - \sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{Y}}_{ijk}}{n_i - \bar{n}_i} = \frac{n_i\mathbf{M}_i - \bar{n}_i\bar{\mathbf{M}}_i}{n'_i} \tag{6.11}$$

$$\mathbf{M}' = \frac{n\mathbf{M} - \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{Y}}_{ijk}}{n - \bar{n}} = \frac{n\mathbf{M} - \bar{n}\bar{\mathbf{M}}}{n'} \tag{6.12}$$

Algorithm 3 2DMvIDDA: Decremental algorithm

Input :
- A trained model ϕ that consists of:
  - Data sample counts: n, n_i, n_{ij}
  - Mean per class per view: $\mathbf{M}_{ij}^{(\mathbf{X})}$
  - Scatter matrices: $\mathbf{S}_{jr}$, $\mathbf{D}_{jr}$
  - Set of class labels of existing data samples: $\mathcal{C}$
- Set of data samples to be deleted: $\bar{\mathcal{X}}$
- Data sample counts of the samples to be deleted: $\bar{n}$, $\bar{n}_i$, $\bar{n}_{ij}$
- Set of classes that are empty after deletion: $\hat{\mathcal{C}}$

while $\bar{\mathcal{X}}$ is not empty **do**
Update n, n_i, n_{ij} using Eq. (6.9).
Update $\mathbf{M}_{ij}^{(\mathbf{X})}$ using Eq.(6.10).
Update $\mathbf{S}_{jr}$ using Eq.(6.14).
Update $\mathbf{D}_{jr}$ using Eq.(6.16).
$\mathcal{C} = \mathcal{C} - \hat{\mathcal{C}}$.
Compute the projection matrix $\mathbf{W}^{opt}$ using Eq.(2.5).
end while
Output :
- Updated model ϕ
- The projection matrix $\mathbf{W}^{opt}$

6.1.2.3 Updating the within-class scatter matrix

We only need to update the scatter of those classes from which some existing data samples are deleted. So, we have,

$$\begin{aligned}\mathbf{S}'_W &= \sum_{i\in\mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i\in\mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk} - \mathbf{M}'_i\right)\left(\mathbf{Y}_{ijk} - \mathbf{M}'_i\right)^T - \sum_{i\in\mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{Y}}_{ijk} - \mathbf{M}'_i\right)\left(\bar{\mathbf{Y}}_{ijk} - \mathbf{M}'_i\right)^T \\ &= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{S}'_{jr} \mathbf{W}_r\end{aligned} \tag{6.13}$$

where,

$$\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jj} - \sum\limits_{i\in\mathcal{CH}} \left[\frac{1}{\bar{n}_i n_i (n_i - \bar{n}_i)} \mathbf{E}\mathbf{E}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i} \bar{\mathbf{M}}_{ij}^{(\mathbf{X})} \bar{\mathbf{M}}_{ij}^{(A)^T} - \sum\limits_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{X}}_{ijk} \bar{\mathbf{X}}_{ijk}^T \right] & j = r \\ \mathbf{S}_{jr} - \sum\limits_{i\in\mathcal{CH}} \left[\frac{1}{\bar{n}_i n_i (n_i - \bar{n}_i)} \mathbf{E}\mathbf{F}^T + \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i} \bar{\mathbf{M}}_{ij}^{(\mathbf{X})} \bar{\mathbf{M}}_{ir}^{(A)^T} \right] & j \neq r \end{cases} \tag{6.14}$$

here, $\mathbf{E} = \bar{n}_i n_{ij} \mathbf{M}_{ij}^{(\mathbf{X})} - \bar{n}_{ij} n_i \bar{\mathbf{M}}_{ij}^{(\mathbf{X})}$ and $\mathbf{F} = \bar{n}_i n_{ir} \mathbf{M}_{ir}^{(\mathbf{X})} - \bar{n}_{ir} n_i \bar{\mathbf{M}}_{ir}^{(\mathbf{X})}$. The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ is similar to that of incremental formulations given in Appendix-F and hence is not presented separately.

6.1.2.4 Updating the between-class scatter matrix

The between-class scatter matrix is updated as,

$$\mathbf{S}'_B = \sum_{i=1}^{c'} n'_i (\mathbf{M}'_i - \mathbf{M}')(\mathbf{M}'_i - \mathbf{M}')^T = \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{D}'_{jr} \mathbf{W}_r \tag{6.15}$$

where each $\mathbf{D}_{jr}$ is given as,

$$\mathbf{D}'_{jr} = \sum_{i=1}^{c'} \frac{n'_{ij}\, n'_{ir}}{n'_i} \mathbf{M}'_{ij}{}^{(\mathbf{X})} \mathbf{M}'_{ir}{}^{(\mathbf{X})^T} - \frac{1}{n'} \left(\sum_{i=1}^{c'} n'_{ij}\, \mathbf{M}'_{ij}{}^{(\mathbf{X})} \right) \left(\sum_{i=1}^{c'} n'_{ir}\, \mathbf{M}'_{ir}{}^{(\mathbf{X})} \right)^T \tag{6.16}$$

The derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ is as given in Appendix-A.

After updating the means and the scatter matrices, we update $\mathcal{C}$ by removing the labels of classes from which all the data samples were deleted. 2DMvIDDA then computes the optimal projection matrix $\mathbf{W}^{opt}$ using Eq. 2.5. The test data samples can be classified using this projection matrix until new training data samples are added, or some existing ones are deleted. If any data samples are deleted, the model is updated using the aforementioned formulations. The formulations of decremental unlearning are presented in Algorithm 3.

6.1.3 Discussion

In this section, we discuss how the discriminant methods, namely 2DMvDA, MvIDDA, MvDA, 2DLDA, and ILDA, are special cases of the proposed method 2DMvIDDA. We also state how their formulations can be obtained from 2DMvIDDA formulations.

6.1.3.1 2D Multi-view Discriminant Analysis (2DMvDA)

2DMvDA is a discriminant analysis-based method for 2D Multi-view data. Let $\mathcal{A} = \{\mathbf{A}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, n_{ij}\}$ be the training set for 2DMvDA. Each $\mathbf{A}_{ijk} \in \mathbb{R}^{p_j \times q}$. The incremental formulation of 2DMvIDDA can be used as 2DMvDA by treating all the data samples as a new set having data samples from c new classes. Here, the existing samples will be 0 as no data samples were already presented for training. Once the scatter matrices are computed using these formulations, the projection matrix $\mathbf{W}^{opt}$ can be obtained by using Eq. 2.5.

6.1.3.2 Multi-view Incremental Decremental Discriminant Analysis (MvIDDA)

MvIDDA is a discriminant analysis-based incremental decremental method for 1D multi-view data. Let the dataset be denoted as $\mathcal{A} = \{\mathbf{a}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, n_{ij}\}$. Each $\mathbf{a}_{ijk} \in \mathbb{R}^{1 \times q}$. The incremental learning formulations of 2DMvIDDA can be used as MvIDDA if we keep the number of rows (p_j) of every data sample equal to 1, as MvIDDA takes 1D data samples as input. The four incremental formulations of MvIDDA are just the special cases of the incremental formulation of 2DMvIDDA. Hence, once the scatter matrices are computed using the incremental formulations according to the type of increment, the projection matrix $\mathbf{W}^{opt}$ can be obtained by using Eq. 2.5. We can also obtain the decremental formulations of MvIDDA using the decremental formulations of 2DMvIDDA in a similar fashion to the incremental formulations.

6.1.3.3 Multi-view Discriminant Analysis (MvDA)

MvDA is a discriminant analysis-based batch method for 1D multi-view data. Let the dataset be denoted as $\mathcal{A} = \{\mathbf{a}_{ijk} | i = 1, \cdots, c; j = 1, \cdots, v; k = 1, \cdots, n_{ij}\}$. Each $\mathbf{a}_{ijk} \in \mathbb{R}^{1 \times q}$. The incremental formulation of 2DMvIDDA can be used as MvDA by

- Keeping the number of rows (p_j) of every data sample equal to 1, as MvDA takes 1D data samples as input.
- Treating all the data samples as one set of data having data samples from c new classes.

Here, the existing samples will be 0 as no data samples were already presented to the model. Once the scatter matrices are computed using these formulations of 2DMvIDDA, the projection matrix $\mathbf{W}^{opt}$ is obtained using Eq.2.5.

6.1.3.4 2D Linear Discriminant Analysis (2DLDA)

2DLDA is a discriminant analysis-based batch method that trains on the single-view 2D data. Let $\mathcal{A} = \{\mathbf{A}_{ijk} | i = 1, \cdots, c; j = 1; k = 1, \cdots, n_{ij}\}$ be the whole training set for 2DLDA. Each $\mathbf{A}_{ijk} \in \mathbb{R}^{p \times q}$ and $j = 1$ as there is only one view. The incremental formulation of 2DMvIDDA can be used as 2DLDA by treating all the data samples as the set of new data having data samples from c new classes and only one view. The number of existing samples will be 0 as no data samples were already presented to the model. The projection matrix $\mathbf{W}^{opt}$ can be obtained by using Eq. 2.5 after the scatter matrices are computed using the incremental formulations of 2DMvIDDA.

6.1.3.5 Incremental Linear Discriminant Analysis (ILDA)

ILDA is an incremental method based on discriminant analysis and trains on 1D single-view data. Let the dataset be denoted as $\mathcal{A} = \{\mathbf{a}_{ijk} | i = 1, \cdots, c; j = 1; k = 1, \cdots, n_{ij}\}$. Each $\mathbf{a}_{ijk} \in \mathbb{R}^{1 \times q}$. The incremental formulations of 2DMvIDDA can be used as ILDA by

- Keeping the number of rows (p_j) of every data sample equal to 1, as ILDA takes 1D data samples as input.
- Keeping $j = 1$ as ILDA is a single-view method.

The optimal projection matrix $\mathbf{W}^{opt}$ can be obtained by using Eq. 2.5 after the scatter matrices are computed using the suitable formulations according to the type of increment. Similar to the incremental formulations, we can also formulate a decremental version of ILDA using the decremental formulations of 2DMvIDDA.

6.2 Experiments and Results

6.2.1 Experimental Setup

We divided the datasets into two parts: the training dataset has 80% of the data samples, and the remaining (20%) are used for testing. The training set is used to simulate increments and decrements in the data. We have taken sets of 20 data samples for each increment or decrement.

The experiments that compare all the 2D and 1D methods are performed using rescaled versions of the datasets owing to the greater memory needs of 1D methods. We compare the classification accuracy, training time, and memory requirements of 2DMvIDDA to the methods discussed in Section 6.1.3.

The experiments that compare only the 2D methods are performed on the original datasets to show that the 2D methods require much less time and memory, even for large images. Here, the classification accuracy remains the same as when the rescaled versions were used. Hence, we focus on the training time and memory requirements. The methodology for classification is the same as that used for MvIDDA.

6.2.2 Results

This section presents the results of experiments designed in order to answer the research questions posed in chapter 1. Research question 1 and 2 are answered in sections 6.2.2.1 and 6.2.2.2. The third question is answered in two parts in sections 6.2.2.3 and 6.2.2.4.

6.2.2.1 Similarity of the Discriminant Subspace

- **RQ1 :** Can the 2D incremental-decremental method, without using old data samples, produce an identical model to that of its batch counterpart, 2DMvDA?
- **Experiment :** To compare the discriminant subspace, we incrementally trained two models using 2DMvIDDA and 2DMvDA, respectively. Then, we deleted existing data samples from the dataset in a similar fashion. At each update, the inner products of the first five eigenvectors of both models are computed. As the eigenvectors are of unit length, we say two eigenvectors are equal when their inner product is 1.

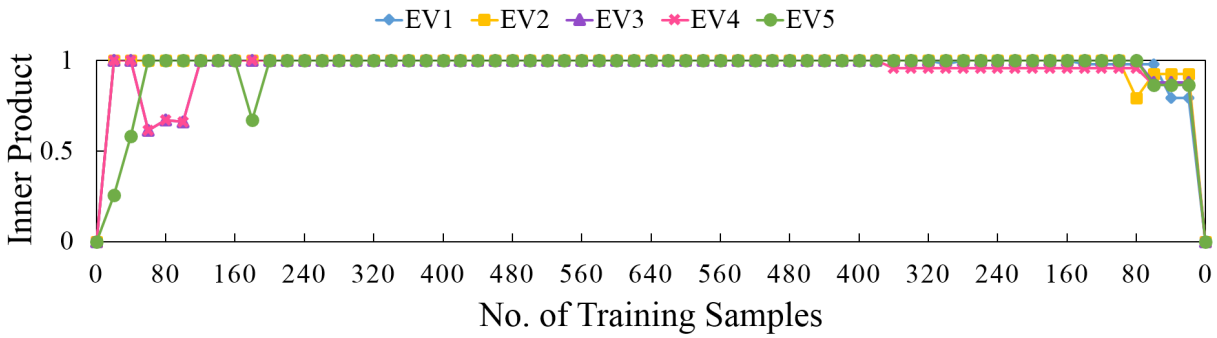


Figure 6.2: Inner product between first five eigenvectors of 2DMvIDDA and 2DMvDA for Stereo face dataset during incremental and decremental phases.

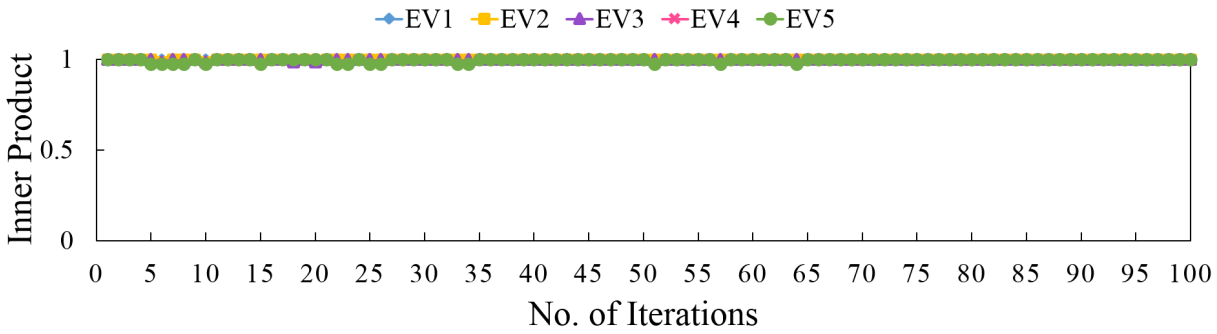


Figure 6.3: Inner product of the eigenvectors over 100 iterations of 2DMvIDDA with those of 2DMvDA.

- **Discussion :** Fig. 6.2 presents the inner products computed over the updates on the Stereo Face dataset. The inner products are presented for the increments from 0 to 640 data samples at first. Then, the inner products for the decrements from 640 to 0 data samples are shown. The figure shows the inner products converged to 1 for all updates, except for the first few increments and the last few sets of decrements. This implies that the shared discriminant subspace constructed by 2DMvIDDA is the same as that constructed by 2DMvDA. A minor difference is seen in the case of first few increments, as the less number of data samples causes some inaccuracies in eigenvalue computation. The same is observed at the end of decrements too.

6.2.2.2 Order Independence

- **RQ2 :** Is this method invariant to the order of addition or deletion of data samples?
- **Experiment :** We trained the model using 2DMvIDDA on the training set a hundred times by adding or deleting the data samples in a randomized order every time. When updates are done after each iteration, we compute the inner product of eigenvectors of the trained model with the eigenvectors of the batch method, 2DMvDA.
- **Discussion :** We experimented on all four datasets and found that the inner product of eigenvectors from both models is equal to or very close to 1 every time. To avoid redundancy, we only present the results for the Stereo Face dataset in Fig. 6.3. Here, we see the inner products of these eigenvectors are converged to 1. This proves the order independence of the proposed method.

Table 6.1: Comparison of 2DMvIDDA with other methods based on discriminant analysis

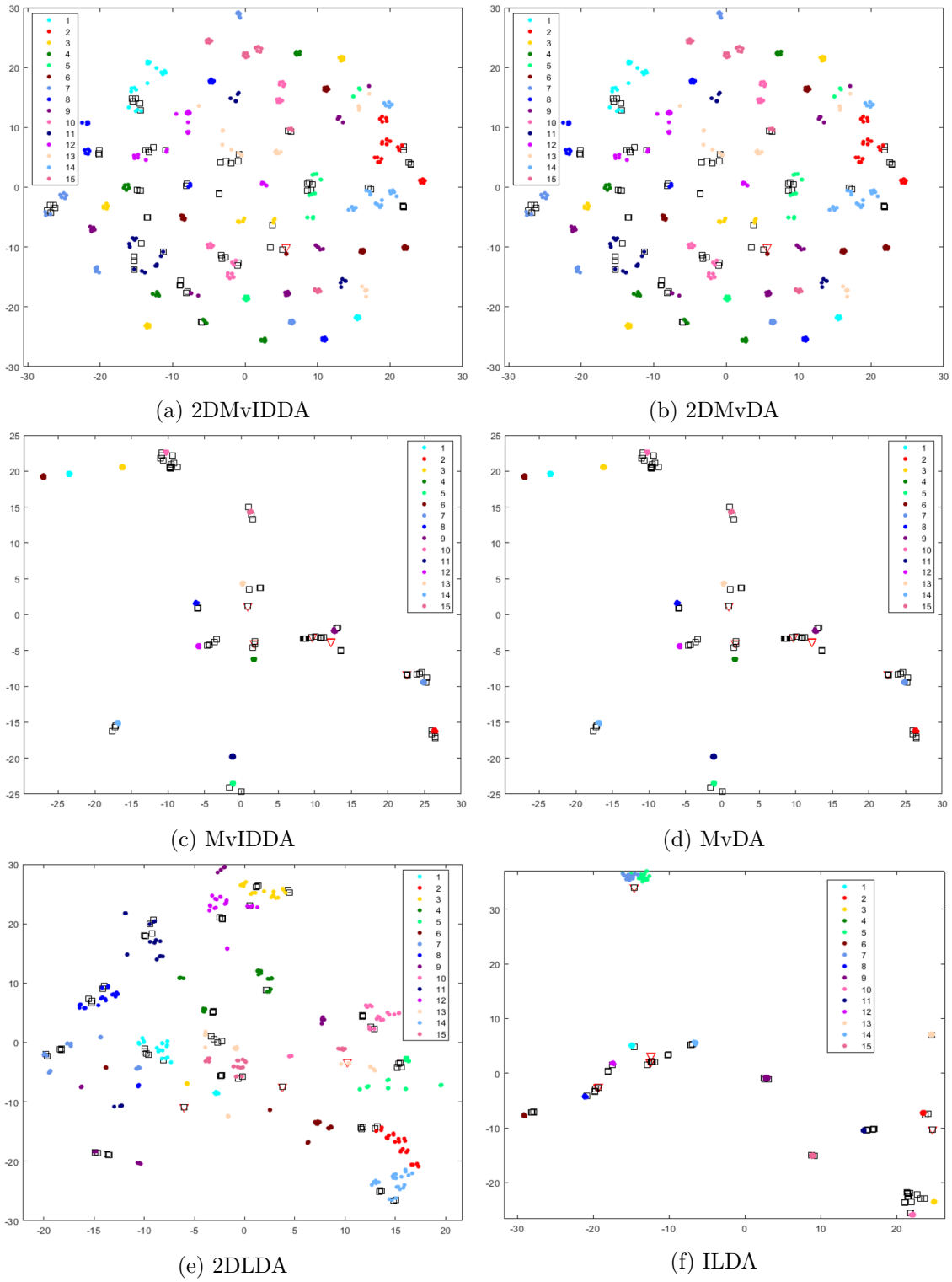
Dataset	Method	Classification Accuracy (%)	Memory (MBs)	Training Time (seconds)
IMPART	**2DMvIDDA**	**100.00**	**3.24**	**0.11**
	2DMvDA	100.00	13.60	0.15
	MvIDDA	100.00	10751.61	587.31
	MvDA	100.00	10761.98	1835.93
	2DLDA	100.00	13.60	1.32
	ILDA	-	-	-
MSSpoof	**2DMvIDDA**	**98.67**	**1.49**	**0.01**
	2DMvDA	98.67	29.75	0.14
	MvIDDA	90.67	1678.95	647.63
	MvDA	90.67	1707.21	2551.35
	2DLDA	96.00	29.75	1.54
	ILDA	86.66	1677.72	3915.05
Stereo Face	**2DMvIDDA**	**75.62**	**20.25**	**0.19**
	2DMvDA	75.62	157.87	0.93
	MvIDDA	73.75	9682.16	1423.10
	MvDA	73.75	9811.52	3970.58
	2DLDA	75.00	157.87	12.24
	ILDA	70.78	9663.67	90863.35
ORL	**2DMvIDDA**	**91.25**	**5.85**	**0.11**
	2DMvDA	91.25	45.41	0.70
	MvIDDA	81.25	1916.05	806.30
	MvDA	81.25	1955.61	1777.62
	2DLDA	86.25	45.41	2.95
	ILDA	57.50	1911.10	5561.28

6.2.2.3 Comparison on the Rescaled Datasets

- **RQ3 :** How is this method more advantageous than other discriminant analysis-based methods?
- **Experiment :** We trained models using 2DMvIDDA, 2DMvDA, MvIDDA, MvDA, 2DLDA, and ILDA on the rescaled versions of the datasets due to the memory constraints of 1D methods. However, ILDA could not run even on the rescaled version of IMPART dataset. Parameters for the rest of the methods on all the datasets have been recorded.
- **Discussion :** Table 6.1 lists the parameters recorded for the above-mentioned methods. Here, we see that 2DMvIDDA shows the highest classification accuracy with the lowest time and memory requirements among all these methods.

  The reason for the higher classification accuracy is the use of multiple views and the 2D form of the data. For the same reasons, 2DMvDA also has the same accuracy as 2DMvIDDA, but it takes much more time and memory for training. MvIDDA and MvDA use the multi-view data effectively, but they lose the spatial information as they use a vectorized form of images. 2DLDA uses the 2D form of data, but it merges all the views and hence does not benefit from the contrasting information that each view offers. ILDA has the least classification accuracy as it uses neither the multiple views separately nor the 2D form of the data.

  2DMvIDDA requires less memory than the rest because (i) the use of 2D form leads to smaller scatter matrices, and (ii) as it is an incremental method, it only needs to store the existing model and the data samples to be added or deleted. It does not need to store the whole dataset, unlike the batch



(a) 2DMvIDDA (b) 2DMvDA

(c) MvIDDA (d) MvDA

(e) 2DLDA (f) ILDA

Figure 6.4: Visualization of Classification on MSSpoof Dataset: (i) Training data samples are denoted with different colored dots for different classes. (ii) Correctly classified test data samples are denoted with black squares. (iii) Misclassified test data samples are denoted with red triangles.

Unlike the batch methods, it does not need to store the whole dataset. 2DMvDA and 2DLDA too benefit from the use of 2D form. However, these are batch methods and they need to store all data samples and the model. So, the demand for storage increases with the number of data samples. MvIDDA and ILDA are incremental methods and do not store old data samples. However, as they use the vectorized format, the size of scatter matrices increases, leading to more memory requirements. MvDA has the largest memory requirements as it is neither an incremental method nor a 2D method. Note that the memory requirements in Table 6.1 are in MBs.

The time records for the incremental methods consist of the time taken for updating the four entities, namely- the number of data sample, the means, the within-class scatter and the between-class scatter. Similarly, the time records for the batch methods consist of the time taken for recomputing these four entities. We have not considered the time taken for the computation of the projection vectors, as this step is common for both types the methods. The reason for the lesser time requirements of 2DMvIDDA is the same as that for the memory requirements. (i) The 2D form leads to smaller scatter matrices which further leads to less computation time, and (ii) due to the incremental nature of the method, it only requires to *update* the model instead of *retraining* like other batch methods. 2DMvDA and 2DLDA require more time because of their non-incremental nature. However, they need less time than the 1D methods as they operate on smaller scatter matrices. MvIDDA and ILDA benefit from the incremental nature, but they need more time because of the larger scatter matrices. Like memory requirements, MvDA is most expensive (except for ILDA) in terms of training time too, as it is a non-incremental 1D method. The reason for ILDA's need for more training time is machine-dependent rather than due to the algorithm itself. As ILDA is a 1D single-view method, the vectorized views are all concatenated and used as the input. In contrast, MvDA and MvIDDA use multi-dimensional matrices, which improves their performance even though the total matrix size is the same for these three methods.

The difference between the time and memory requirements of 1D batch and 1D incremental methods is not as much as in the case of 2D methods because the data sample count is significantly less than the number of dimensions. The difference will be more pronounced when the number of data samples is increased.

Figure 6.4 presents the classification results of these methods on the MSSpoof dataset plotted using t-SNE. The different colored dots represent the training data samples from different classes. The correctly classified test data samples are denoted with black squares, and the incorrectly classified data samples are denoted with red triangles. We can see that the subspace constructed by 2DMvIDDA, 2DMvDA, and 2DLDA is better at discriminating between different classes than that constructed by the 1D methods. Also, The 1D multi-view methods (MvIDDA and MvDA) have constructed a better subspace than the 1D single-view method ILDA. This difference in the subspace construction leads to the difference in the classification performance of these methods.

6.2.2.4 Comparison on the Original Datasets

- **RQ3 :** How is this method advantageous than other discriminant analysis-based methods?
- **Experiment :** We train classifiers on the original datasets using only the 2D methods and record their time and memory requirements. Here, the time records for 2DMvIDDA consist of the time taken for updating the four entities, namely- the number of data sample, the means, the within-class scatter and the between-class scatter. Similarly, the time records for 2DMvDA and 2DLDA consist of the time taken for recomputing these four entities. We have not considered the time taken for the computation of the projection vectors, as this step is common for all the methods. To give an idea of the reduction in memory requirements due to the use of 2D data, we also estimated the memory requirements of MvIDDA, which requires the least memory among the three 1D methods. Note that the memory requirements are in GBs.
- **Discussion :** Table 6.2 lists the training time and memory usage of these 2D methods. We see the same trends with the original datasets as with the rescaled ones. Even among the 2D methods, 2DMvIDDA requires much less memory than the other two methods. However, the main aim here is to

Table 6.2: Comparison of 2DMvIDDA with other 2D methods on the original datasets. Table also lists estimated memory requirements of MvIDDA.

Dataset	Method	Memory (GBs)	Training Time (seconds)
IMPART	**2DMvIDDA**	**1.29**	**151.19**
	2DMvDA	5.44	356.59
	2DLDA	5.44	3094.38
	MvIDDA	1601800	-
MSSpoof	**2DMvIDDA**	**0.38**	**14.07**
	2DMvDA	7.61	126.54
	2DLDA	7.61	1686.57
	MvIDDA	102400	-
Stereo	**2DMvIDDA**	**0.50**	**10.46**
	2DMvDA	3.94	44.90
	2DLDA	3.94	598.85
	MvIDDA	5625	-
ORL	**2DMvIDDA**	**0.02**	**0.41**
	2DMvDA	0.18	2.77
	2DLDA	0.18	11.42
	MvIDDA	15.2	-

show how little are the memory requirements of 2DMvIDDA compared to 1D methods, which is evident from the huge memory requirements of MvIDDA. As 1D methods use the vectorized form of input images, the size of scatter matrices increases drastically. In the IMPART dataset, the images are of size 1920×1080, so scatter matrices computed by the 1D methods will be of size 10368000×10368000, whereas those computed by the 2D methods are of size 9600×9600 only. This reduction in the size of scatter matrices saves enormous amounts of memory and training time. ILDA also requires the same amount of memory as MvIDDA as it is also a 1D incremental method. Being a batch method, MvDA needs even more memory to store all the historical data.

6.3 Summary

We have presented an incremental-decremental method for 2D multi-view data in this chapter. This method enables the use of the original 2D form of the data to capture the spatial features and multiple views to gather contrasting information. The proposed method does not need to store historical data. It incrementally learns a model by taking only the old model and the data samples to be added. It also decrementally unlearns existing data samples, using the existing model and the data samples to be deleted. 2DMvIDDA is also presented as the generalized form of other discriminant analysis-based methods, namely- 2DMvDA, MvIDDA, MvDA, 2DLDA, and ILDA. The formulations of these methods can be obtained from the formulations or a part of the formulations of 2DMvIDDA.

We show that being a scalable method for 2D multi-view data, 2DMvIDDA constructs a better discriminant subspace with significantly less time and memory. This subspace constructed by 2DMvIDDA is the same as that of batch 2DMvDA. 2DMvIDDA also handles data samples from previously unseen classes and is order-invariant.

The next chapter presents an incremental-decremental method for multi-view data that can add or remove views from a trained model. This method is also based on discriminant analysis.

7

View Incremental Decremental Multi-view Discriminant Analysis

We present in this chapter View Incremental Decremental Multi-view Discriminant Analysis (VIDMvDA), which is a classification method for 2D multi-view data. The motivation behind this method is similar to that of the other incremental methods presented in this thesis. However, those methods aimed to facilitate the addition/deletion of data samples, whereas VIDMvDA aims at supporting the addition/deletion of views. VIDMvDA is based on discriminant analysis and aims to reduce computational costs. It supports the addition or deletion of multiple views without retraining the model from scratch. All the updates in the model are obtained in a closed-form.

We present the formulations of VIDMvDA, experimental setup, and results in subsequent sections. We compare the proposed method with its batch counterpart (MvDA) on parameters such as- similarity of the discriminant subspace, classification accuracy, training time, and memory requirements. We also show that the sequence of addition or removal of views of data does not alter the final common discriminant subspace. We conclude this chapter by summarizing the features of VIDMvDA.

7.1 Methodology

As we saw in section 2.1.1.1, the scatter matrices are made up of view-wise submatrices. These submatrices are computed in a pairwise manner from all the views. To add a new view, we must compute the scatter of this view with respect to each of the existing views and itself. For example, when the within-class scatter of the data samples belonging to $(v+1)^{th}$ view is to be updated, $\mathbf{S}_{j(v+1)}$ and $\mathbf{S}_{(v+1)j}$ for all $\{j = 1, \cdots, (v+1)\}$ are computed. These pairwise submatrices are then appended onto the existing scatter matrix $\mathbf{S}$. We also need to update the existing submatrices as the number of data samples per class used to compute the scatters changes after each increment. In the case of decrement, we delete the submatrices of the deleted view/s and update the remaining submatrices using the updated number of data samples per class.

Fig. 7.1 presents an example of updates in the scatter matrix with the increments or decrements in a pictorial form. In this figure, we see that at time t_1, there exists a within-class scatter matrix of two views. When a new view 3 is added at the transition from t_1 to t_2,

- The submatrices pertaining to the third view ($\mathbf{S}_{13}, \mathbf{S}_{23}, \mathbf{S}_{31}$, $\mathbf{S}_{32}$ and $\mathbf{S}_{33}$) are appended to the scatter matrix from t_1 as shown in Fig. 7.1.
- The existing submatrices from t_1 ($\mathbf{S}_{11}, \mathbf{S}_{12}, \mathbf{S}_{21}$ and $\mathbf{S}_{22}$) are updated in t_2 to reflect the addition of the new view.

When view 2 is deleted at the transition from t_2 to t_3,

- The submatrices pertaining to the second view ($\mathbf{S}_{12}, \mathbf{S}_{21}$, $\mathbf{S}_{22}, \mathbf{S}_{23}$ and $\mathbf{S}_{32}$) from t_2 are deleted at t_3.

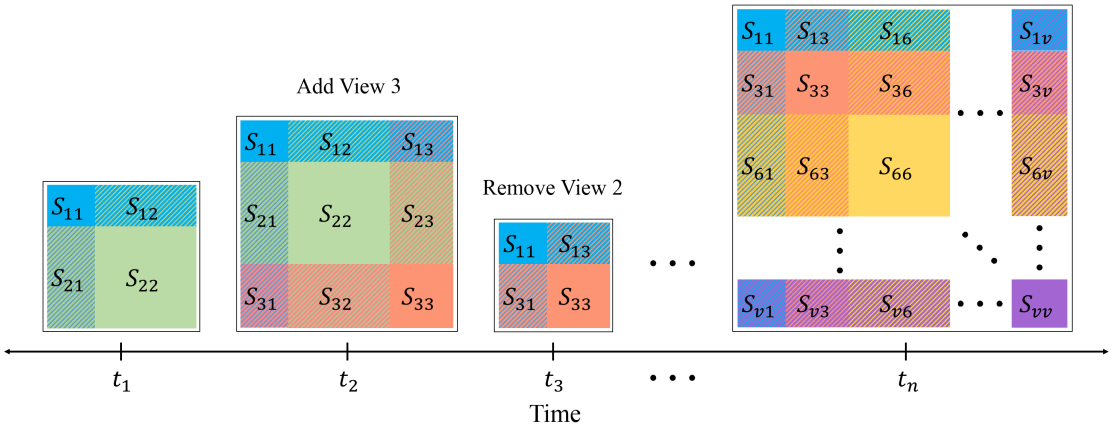


Figure 7.1: Updates in a scatter matrix after the addition and deletion of views over time. Each submatrix $\mathbf{S}_{jr}$ is represented with the combination of colors of j^{th} and r^{th} view.

- The remaining submatrices ($\mathbf{S}_{11}, \mathbf{S}_{13}, \mathbf{S}_{31}$ and $\mathbf{S}_{33}$) from t_2 are updated at t_3 to reflect the deletion of the view.

Following similar procedures for adding and deleting more views, we see the evolved scatter matrix at t_n.

7.1.1 Addition of Views

Let the set of new views be denoted as $\bar{\mathcal{V}} = \{v+1, v+2, \cdots, v'\}$. When new views are added, four quantities are updated, namely- (i) data sample counts, (ii) the means, (iii) the within-class scatter and (iv) the between-class scatter. These updates are given as follows:

7.1.1.1 Updating the number of data samples

As each of the new views contains features of all the existing data samples, the cardinality of each new view is the same as any previous view. Hence, the *number of samples per class per view* (n'_{ij}) is also the same as that of any previous views. Other counts n_i and n are updated as follows,

$$n'_i = n_i + \sum_{j=v+1}^{v'} n'_{ij} \quad \text{and} \quad n' = n + \sum_{i=1}^{c} \sum_{j=v+1}^{v'} n'_{ij} \tag{7.1}$$

7.1.1.2 Updating the means

The three means ($\mathbf{m}_{ij}$, $\mathbf{m}_i$, $\mathbf{m}$) are updated as follows,

$$\mathbf{m}'^{(\mathbf{x})}_{ij} = \bar{\mathbf{m}}^{(\mathbf{x})}_{ij} = \frac{1}{\bar{n}_{ij}} \sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk} \tag{7.2}$$

$$\mathbf{m}'_i = \frac{n_i\, \mathbf{m}_i + \bar{n}_i\, \bar{\mathbf{m}}_i}{n'_i} \tag{7.3}$$

$$\mathbf{m}' = \frac{n\, \mathbf{m} + \bar{n}\, \bar{\mathbf{m}}}{n'} \tag{7.4}$$

Algorithm 4 VIDMvDA: Incremental algorithm

Input :

- An existing model ϕ that consists of n, n_i, n_{ij}, $\mathbf{m}_{ij}^{(\mathbf{x})}$, $\mathbf{S}$, $\mathbf{D}$, $\mathcal{C}$, $\mathcal{V}$
- New views of data samples: $\bar{\mathcal{X}}$
- Set of new views: $\bar{\mathcal{V}}$

while new data is encountered **do**

Update n, n_i, n_{ij} as given in Eq.7.1
Update mean per class per view ($\mathbf{m}_{ij}^{(\mathbf{x})}$) using Eq.7.2
Update the within-class scatter matrix ($\mathbf{S}$) using Eq.7.6
Update the between-class scatter matrix ($\mathbf{D}$) using Eq.7.8
$\mathcal{V} = \mathcal{V} \cup \bar{\mathcal{V}}$
Calculate the projection matrix ($\mathbf{W}^{opt}$) using Eq.2.5

end while

Output :

- Updated model ϕ' and the projection matrix $\mathbf{W}^{opt}$

7.1.1.3 Updating the within-class scatter

The new scatter matrix is the summation of the scatter of the old and the new view with respect to the updated class mean. The equation to update within-class scatter is as follows,

$$\mathbf{S}'_W = \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} (\mathbf{y}_{ijk} - \mathbf{m}'_i)(\mathbf{y}_{ijk} - \mathbf{m}'_i)^T + \sum_{i=1}^{c}\sum_{j=v+1}^{v'}\sum_{k=1}^{\bar{n}_{ij}} (\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i)(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i)^T \tag{7.5}$$

This equation is in the projected space. We then express it only in terms of the old scatter matrix and new data in the original space. Here, we compute the submatrices ($\mathbf{S}_{jr}$) of the within-class scatter of the new view with existing views in a pair-wise manner. Also, the existing submatrices are updated to reflect the change in the *number of samples per class*. So, we have three cases based on the values of j and r. At the transition from t_1 to t_2 in Fig. 7.1, we see these three cases.

- *Case-1* is when ($j \in \bar{\mathcal{V}}$ && $r \in \bar{\mathcal{V}}$), but ($j \neq r$). This case is for computing the scatter of new views with the existing views. In Fig. 7.1, submatrices $\mathbf{S}_{13}, \mathbf{S}_{23}, \mathbf{S}_{31}$ and $\mathbf{S}_{32}$ at t_2 represent this case.
- *Case-2* is when ($j \in \bar{\mathcal{V}}$ && $r \in \bar{\mathcal{V}}$) and ($j = r$), that is when we compute the scatter of the new view with respect to itself. Submatrix $\mathbf{S}_{33}$ at t_2 represents this case in Fig. 7.1.
- *Case-3* is for updating the existing submatrices, which occurs when ($j \notin \bar{\mathcal{V}}$ && $r \notin \bar{\mathcal{V}}$). Submatrices $\mathbf{S}_{11}, \mathbf{S}_{12}$, $\mathbf{S}_{21}$ and $\mathbf{S}_{22}$ at t_2 in Fig. 7.1 represent *Case-3*.

The equation for updating the within-class scatter matrix in all three cases is given below.

$$\mathbf{S}'_{jr} = \begin{cases} -\sum_{i=1}^{c} \frac{\bar{n}_{ij}\bar{n}_{ir}}{n'_i}\, \mathbf{m}_{ij}{}^{(\mathbf{x})}\, \mathbf{m}_{ir}{}^{(\mathbf{x})^T} & \text{case 1} \\ \sum_{i=1}^{c}\left[\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{n'_i}\, \mathbf{m}_{ij}{}^{(\mathbf{x})}\, \mathbf{m}_{ij}{}^{(\mathbf{x})^T}\right] & \text{case 2} \\ \mathbf{S}_{jr} + \sum_{i=1}^{c} \frac{n'_i - n_i}{n_i n'_i}\, n_{ij} n_{ir}\, \mathbf{m}_{ij}{}^{(\mathbf{x})} \mathbf{m}_{ir}{}^{(\mathbf{x})^T} & \text{case 3} \end{cases} \tag{7.6}$$

The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ is given in Appendix-G.

7.1.1.4 Updating the between-class scatter

The between-class scatter depends only on the means, so updating the between-class scatter after each increment is easier. The equation in projected space does not explicitly depend on the views.

$$\mathbf{S}'_B = \sum_{i=1}^{c} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \tag{7.7}$$

Here, the view information is embedded in the total and class-wise means. It creates two cases of updates when rewritten using the data samples in the original space.

In *Case-1*, the between-class scatter of the new views with the existing views is computed in a pairwise manner and appended to the existing scatter matrix ($\mathbf{D}$). Here, ($j \in \bar{\mathcal{V}} \;\; || \;\; r \in \bar{\mathcal{V}}$). In *Case-2*, all the existing submatrices are adjusted to reflect the change in the means and the number of data samples. Here, ($j \notin \bar{\mathcal{V}} \;\; \&\& \;\; r \notin \bar{\mathcal{V}}$). The update equations for both the cases of the between-class scatter are as follows,

$$\mathbf{D}'_{jr} = \begin{cases} \sum\limits_{i=1}^{c} \frac{n'_{ij}\, n'_{ir}}{n'_i}\, \mathbf{m}'_{ij}{}^{(\mathbf{x})}\, \mathbf{m}'_{ir}{}^{(\mathbf{x})^T} - \frac{1}{n'} \left(\sum\limits_{i=1}^{c} n'_{ij}\, \mathbf{m}'_{ij}{}^{(\mathbf{x})} \right) \left(\sum\limits_{i=1}^{c} n'_{ir}\, \mathbf{m}'_{ir}{}^{(\mathbf{x})} \right)^T & \text{case 1} \\ \mathbf{D}_{jr} - \sum\limits_{i=1}^{c} \frac{n'_i - n_i}{n_i\, n'_i}\, n_{ij}\, n_{ir} \mathbf{m}_{ij}{}^{(\mathbf{x})}\, \mathbf{m}_{ir}{}^{(\mathbf{x})^T} + \frac{n' - n}{n\, n'} \left(\sum\limits_{i=1}^{c} n_{ij}\, \mathbf{m}_{ij}{}^{(\mathbf{x})} \right) \left(\sum\limits_{i=1}^{c} n_{ir}\, \mathbf{m}_{ir}{}^{(\mathbf{x})} \right)^T & \text{case 2} \end{cases} \tag{7.8}$$

The derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ is given in Appendix-A. After performing steps (i)-(iv), we compute the optimal projection matrix by solving the generalized eigenvalue problem given in Eq. 2.5. The obtained projection matrix ($\mathbf{W}^{opt}$) is then used for classification by projecting each test sample onto a shared subspace where they can be assigned labels based on their proximity to the available classes. This projection matrix is used until further addition or deletion of the views. Whenever we encounter yet another new view/s of the data, we repeat the update process and compute the new projection matrix. The whole process is summarized in Algorithm 4. This formulation can be used for 2D data as well.

7.1.2 Deletion of Views

Let $\bar{\mathcal{V}} \subset \mathcal{V}$ be a set of $\bar{v}$ views to be removed from the dataset. When any of the existing views are deleted, four quantities, namely (i) the data sample counts, (ii) the means, (iii) the within-class scatter, and (iv) the between-class scatter. These updates are given as follows:

7.1.2.1 Updating the number of data samples

As these views of all existing data samples are deleted, the *number of data samples per class per view* (n'_{ij}) of each of these views will also be deleted or made equal to 0. The *number of data samples per class* (n_i), and the *total number of data samples* (n) are updated as,

$$n'_i = n_i - \sum_{j=1}^{\bar{v}} n_{ij} \quad \text{and} n' = n - \sum_{i=1}^{c} \sum_{j=1}^{\bar{v}} n_{ij} \tag{7.9}$$

7.1.2.2 Updating the means

The three means ($\mathbf{m}_{ij}$, $\mathbf{m}_i$, $\mathbf{m}$) are updated as follows,

$$\mathbf{m}'_{ij}{}^{(\mathbf{x})} = 0 \tag{7.10}$$

$$\mathbf{m}'_i = \frac{n_i\, \mathbf{m}_i - \bar{n}_i\, \bar{\mathbf{m}}_i}{n'_i} \tag{7.11}$$

$$\mathbf{m}' = \frac{n\, \mathbf{m} - \bar{n}\, \bar{\mathbf{m}}}{n'} \tag{7.12}$$

Algorithm 5 VIDMvDA: Decremental algorithm

Input :

- An existing model ϕ that consists of n, n_i, n_{ij}, $\mathbf{m}_{ij}^{(\mathbf{x})}$, $\mathbf{S}$, $\mathbf{D}$, $\mathcal{C}$, $\mathcal{V}$
- Views of data samples to be removed: $\bar{\mathcal{X}}$
- Set of views to be removed: $\bar{\mathcal{V}}$

while there is a view is to be removed **do**

Update n, n_i, n_{ij} as given in 7.9
Update mean per class per view ($\mathbf{m}_{ij}^{(\mathbf{x})}$) using Eq.7.10
Update the within-class scatter matrix ($\mathbf{S}$) using Eq.7.14
Update the between-class scatter matrix ($\mathbf{D}$) using Eq.7.16
$\mathcal{V} = \mathcal{V} - \bar{\mathcal{V}}$
Calculate the projection matrix ($\mathbf{W}^{opt}$) using Eq.2.5

end while

Output :

- Updated model ϕ' and the projection matrix $\mathbf{W}^{opt}$

7.1.2.3 Updating the within-class scatter

The new scatter matrix is the summation of the scatter of the remaining data samples from the updated class mean. The equation in the projected space is as follows,

$$\mathbf{S}'_W = \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} (\mathbf{y}_{ijk} - \mathbf{m}'_i)(\mathbf{y}_{ijk} - \mathbf{m}'_i)^T - \sum_{i=1}^{c}\sum_{j=1}^{\bar{v}}\sum_{k=1}^{\bar{n}_{ij}} (\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i)(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i)^T \tag{7.13}$$

This equation is in the projected space. In the original space, we delete all the submatrices ($\mathbf{S}_{jr}$) of the within-class scatter that pertain to the views to be deleted. The remaining submatrices are updated to reflect the change in the *number of samples per class*. Here, we have two cases based on the values of j and r. For example, at the transition from t_2 to t_3 in Fig. 7.1, we see these two cases.

- *Case-1* is when either ($j \in \bar{\mathcal{V}}$ || $r \in \bar{\mathcal{V}}$). This case is for deleting the pairwise scatter of views in $\bar{\mathcal{V}}$ with any other view. Submatrices $\mathbf{S}_{12}, \mathbf{S}_{21}, \mathbf{S}_{22}, \mathbf{S}_{23}$ and $\mathbf{S}_{32}$ from time-stamp t_2 are deleted from the scatter matrix at t_3 in Fig. 7.1.
- *Case-2* is for updating the existing submatrices. This occurs when ($j \notin \bar{\mathcal{V}}$ && $r \notin \bar{\mathcal{V}}$). Submatrices $\mathbf{S}_{11}, \mathbf{S}_{13}, \mathbf{S}_{31}$ and $\mathbf{S}_{33}$ at t_3 represent *Case-2* in Fig. 7.1.

The within-class scatter matrix is updated as follows-

$$\mathbf{S}'_{jr} = \begin{cases} Delete & \text{case 1} \\ \mathbf{S}_{jr} + \sum_{i=1}^{c} \frac{n'_i - n_i}{n_i n'_i}\, n_{ij} n_{ir}\, \mathbf{m}_{ij}{}^{(\mathbf{x})} \mathbf{m}_{ir}{}^{(\mathbf{x})^T} & \text{case 2} \end{cases} \tag{7.14}$$

The derivation of $\mathbf{S}'_W$ and $\mathbf{S}'_{jr}$ is the same as in the case of incremental formulation given in Appendix-G.

7.1.2.4 Updating the between-class scatter

As we have seen in the case of increment, the equation of between-class scatter in projected space does not explicitly depend on the views.

$$\mathbf{S}'_B = \sum_{i=1}^{c} n'_i (\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \tag{7.15}$$

The between-class scatter also has the same two cases as the within-class scatter in the original space. For *case-1*, the submatrices pertaining to the views in set $\bar{\mathcal{V}}$ are deleted. For *case-2*, the submatrices are updated as follows,

$$\mathbf{D}'_{jr} = \begin{cases} Delete & \text{case 1} \\ \sum_{i=1}^{c} \frac{n'_{ij}\, n'_{ir}}{n'_i}\, \mathbf{m}_{ij}{}^{(\mathbf{x})}\, \mathbf{m}_{ir}{}^{(\mathbf{x})^T} - \frac{1}{n'} \left(\sum_{i=1}^{c} n'_{ij}\, \mathbf{m}_{ij}{}^{(\mathbf{x})} \right) \left(\sum_{i=1}^{c} n'_{ir}\, \mathbf{m}_{ir}{}^{(\mathbf{x})} \right)^T & \text{case 2} \end{cases} \tag{7.16}$$

The derivation of $\mathbf{S}'_B$ and $\mathbf{D}'_{jr}$ is given in Appendix-A.

After following steps (i)-(iv), we compute the optimal projection matrix using Eq. 2.5. The projection matrix ($\mathbf{W}^{opt}$), thus obtained, is then used until a new view of the data arrives or any of the existing view/s are deleted. While deleting another view/s of the data, we repeat the update process presented above and compute the new projection matrix. The whole process is summarized in Algorithm 5. This formulation can be used for 2D data as well.

7.2 Experiments and Results

7.2.1 Experimental Setup

We have divided the data samples from each dataset into two subsets- one for training (80%) and the other as a test set (20%). The training set is used to simulate the increments and decrements of views. Though VIDMvDA can handle one or more views of data in each increment or decrement, to demonstrate the effect of adding the views, we have used the views one at a time for the experiments.

For experiments with 1D datasets, two models, ϕ_A and ϕ_B, are trained on each dataset. ϕ_A is trained using VIDMvDA, and ϕ_B is trained using the 1D batch method, MvDA. While simulating the increments, we add one view at a time from 1 to 6. VIDMvDA updates ϕ_A by considering only the old model and the new view after each increment, whereas MvDA discards ϕ_B and trains a new model after each increment using all the existing views and the new view. After both models have been trained on all views of the dataset, we simulate decrement by removing one view at a time from 1 to 5. While simulating the decrement, ϕ_A is updated to reflect the deletion of each view using only the existing model, whereas ϕ_B is discarded and trained anew using all the remaining views after the deletion.

For experiments with 2D datasets, two models, ϕ_A and ϕ_B, are trained on each dataset. ϕ_A is trained using VIDMvDA, and ϕ_B is trained using the 2D batch method, 2DMvDA. The increments and decrements are simulated in a similar way as with the 1D datasets.

The updated models after each increment/decrement are used to classify the data samples from the test set. Here, only the views used in the training phase of that time step are taken into account. Let $\mathbf{W}_A^{opt}$ and $\mathbf{W}_B^{opt}$ be the optimal projection matrices obtained from models ϕ_A and ϕ_B, respectively. The projection matrix from each method is then used to project every test sample ($\mathbf{y}^{Test}$) onto the shared space. We use the same methodology as that used by MvIDDA for classification.

7.2.2 Results

This section presents the results of experiments designed in order to answer the research questions posed in chapter 1.

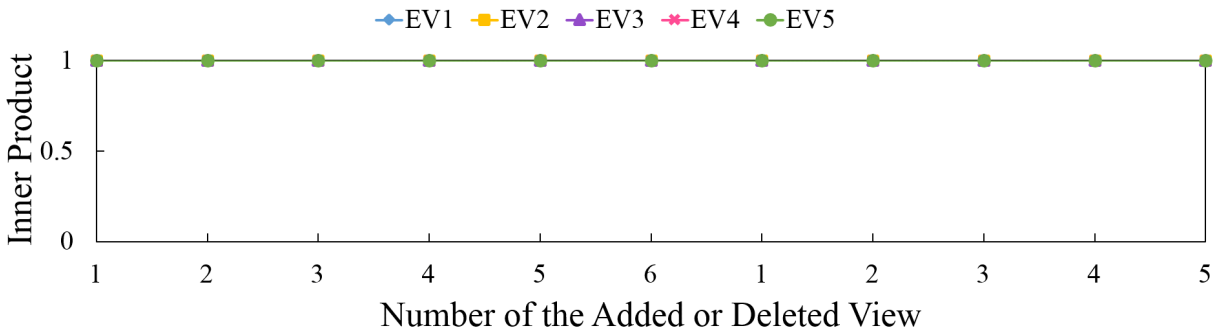


Figure 7.2: Inner products of the first five projection vectors of VIDMvDA and MvDA on handwritten digits dataset. Views are added to the initially empty dataset one by one from 1 to 6. After that, they are removed from the dataset in the order from 1 to 5.

7.2.2.1 Similarity of the Discriminative Subspace

- **RQ1 :** Can the incremental-decremental method, without using old views, produce an identical model to that of its batch counterpart?
- **Experiment :** We train two models using VIDMvDA and its batch counterpart to compare the discriminant subspace. We first incrementally learn all views one by one and then unlearn those in a similar fashion. The inner product of projection vectors of these methods is taken after each update.
- **Discussion :** Fig. 7.2 presents the inner product of projection vectors obtained from the models trained on handwritten digits dataset. The results on all the other datasets are similar to those presented in this figure. Hence, no separate results are presented. We see the inner product of projection vectors after adding each view incrementally from 1 to 6. The figure also shows the inner product after removing them in the same order from 1 to 5. After the deletion of the 6^{th} view, the scatter matrices become zero. Here, the labels EV1 to EV5 represent the first five projection vectors and the number of the added/deleted view is written on the x-axis. We see that the inner product of projection vectors is either 1 or very close to 1 even after the addition of new views. The same observation is made for the deletion operation. This observation shows that the discriminant subspace constructed by VIDMvDA is nearly identical to that of the its batch counterparts, MvDA or 2DMvDA.

7.2.2.2 Order Independence

- **RQ2 :** Is this method invariant to the order of addition or deletion of data samples?
- **Experiment :** We trained a hundred models using VIDMvDA on the training set by adding or deleting the data samples in a randomized order every time. When the updates are done after each iteration, we compute the inner product of the eigenvectors of the trained model with the eigenvectors of the batch method, MvDA for 1D datasets or 2DMvDA for 2D datasets.
- **Discussion :** We use the inner product of projection vectors to check whether the subspace constructed by VIDMvDA is similar to that constructed by a batch method. Here, we have

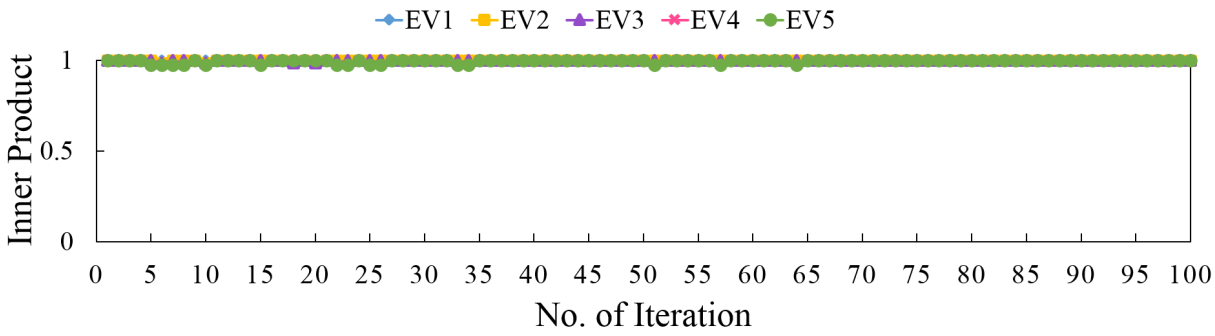


Figure 7.3: Inner products of the first five projection vectors of 100 iterations of VIDMvDA with those of MvDA on handwritten digits dataset.

Table 7.1: Comparison of the classification accuracy of VIDMvDA and MvDA. First column shows the number of the view that was added or deleted.

View	**Handwritten Digits**		**Caltech-7**		**AwA**	
No.	**VIDMvDA**	**MvDA**	**VIDMvDA**	**MvDA**	**VIDMvDA**	**MvDA**
Addition of Views						
1	98.25	96.75	89.00	89.00	90.69	90.69
2	98.75	92.00	90.00	90.00	94.43	94.43
3	98.00	97.25	92.50	92.50	94.27	94.27
4	100.00	99.00	96.00	96.00	94.43	92.79
5	100.00	98.75	97.00	97.00	95.43	95.43
6	99.75	99.00	97.00	97.00	96.55	96.55
Deletion of Views						
1	100.00	99.00	97.00	97.00	96.55	96.55
2	100.00	98.75	95.50	95.50	96.02	95.73
3	98.75	98.75	96.50	96.50	96.02	96.02
4	100.00	97.00	96.00	96.00	94.88	94.88
5	100.00	98.75	96.00	96.00	93.36	93.36

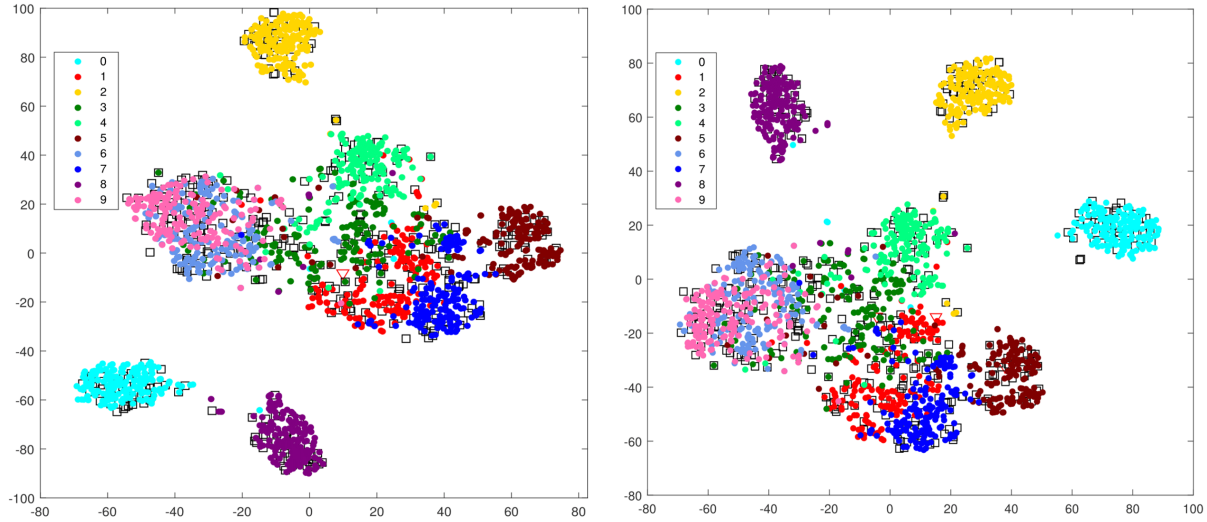
only shown the results obtained on the Handwritten Digits dataset to avoid redundancy. We see in Fig. 7.3 that the values of the inner product of the projection vectors have converged to 1 regardless of the order of addition or deletion of views, demonstrating the order independence of VIDMvDA.

7.2.2.3 Comparison of Classification Accuracy

- **RQ3 :** Can this method perform as well as the batch method in terms of classification accuracy?
- **Experiment :** We have noted the classification performance of the incremental and the batch methods after adding each view to the dataset and later removing them one by one.
- **Discussion :** Here, we have used 5 projection vectors, i.e. $d = 5$, for both the models ϕ_A and ϕ_B. The results show that VIDMvDA has the same or better classification accuracy than its batch counterparts. Table 7.1 presents the records of classification accuracies on 1D datasets

Table 7.2: Comparison of the classification accuracy of VIDMvDA and 2DMvDA. First column shows the number of the view that was added or deleted.

View	IMPART		MSSpoof		Stereo Face		ORL	
No.	**VIDMvDA**	**2DMvDA**	**VIDMvDA**	**2DMvDA**	**VIDMvDA**	**2DMvDA**	**VIDMvDA**	**2DMvDA**
Addition of Views								
1	100.00	100.00	97.33	97.33	95.00	95.00	90.00	90.00
2	90.00	90.00	100.00	98.67	98.12	98.12	87.50	87.50
3	100.00	100.00	-	-	-	-	91.25	91.25
4	100.00	100.00	-	-	-	-	-	-
5	100.00	100.00	-	-	-	-	-	-
Deletion of Views								
1	100.00	100.00	100.00	100.00	97.50	97.50	93.75	93.75
2	100.00	100.00	-	-	-	-	78.75	78.75
3	100.00	100.00	-	-	-	-	-	-
4	100.00	100.00	-	-	-	-	-	-



(a) VIDMvDA (b) MvDA

Figure 7.4: A plot of data samples of the handwritten digits dataset in the projected space constructed by (a) VIDMvDA and (b) MvDA. Training data samples from different classes are shown in different colors. Black squares show correctly classified test samples, and red triangles show incorrectly classified test samples.

and Table 7.2 shows the records for 2D datasets. We saw in 7.2.2.1, that the discriminant subspaces of both models are very similar. The same is observed with the classification accuracy as well. The accuracy of these two models is the same for all the datasets, except for the handwritten digits dataset and MSSpoof dataset. VIDMvDA performs a little better than the batch methods on these datasets.

A t-SNE plot of handwritten digits dataset in the projected space constructed by VIDMvDA and MvDA is presented in Fig. 7.4. Here, 0 to 9 are the class labels. We see that both models project the data in a similar manner. Both methods have set classes 0, 2 and 8 (denoted with cyan, yellow and violet) farther from the other classes. Classes 6 and 9 (pink and sky-blue) are closer to each other, as are classes 1 and 7 (red and blue).

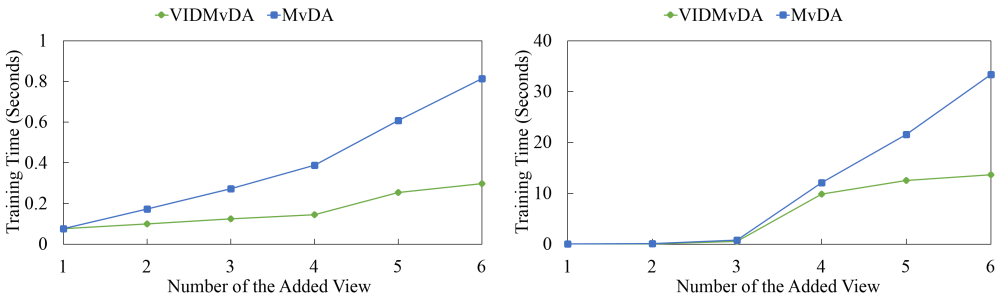
7.2.2.4 Comparison of Training Time

- **RQ4 :** Can this method reduce training time?
- **Experiment :** To compare the training time of VIDMvDA and MvDA/2DMvDA, we trained models using these methods by adding the views in a one-by-one manner from view 1 to 6. Similarly, for deletion, the time was noted after each view was deleted from view 1 to 5. We record the time taken by each method after every addition/deletion of views.
- **Discussion :** As VIDMvDA only updates the existing model instead of retraining from scratch, it is expected to require less time for training than the batch method as the latter discards the existing model and retrains on the whole dataset. This observation is reflected in the recorded training time of these methods. Fig. 7.5 and 7.6 present the records of training time of three 1D datasets and one 2D dataset for increments and decrements, respectively. The records for the remaining 2D datasets are presented in Table 7.3 as these datasets have less number of views, and hence, the results cannot be effectively shown as a line graph.

  We see in Fig. 7.5 and Table 7.3 that as new views get added over time, the batch methods need much more training time than VIDMvDA. VIDMvDA only updates the model when a new view is added, whereas the batch methods train on all the historical views and the new view. Hence, the difference in the training time increases with the number of views. This difference in training time becomes more pronounced in the AwA dataset as it is the largest among these datasets. We can see that VIDMvDA requires around 1.5 hours to train on the AwA dataset with all six views, whereas, MvDA takes more than 6 hours to complete training on the same dataset.

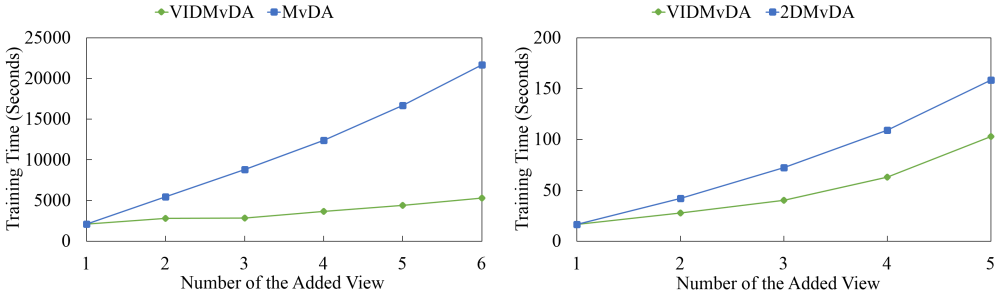
  We see in Fig. 7.6 and Table 7.3 that though the number of views is decreasing over time, the batch methods still take more training time than VIDMvDA. Similar to the addition, VIDMvDA only updates the model each time a view is deleted, whereas the batch methods train on all historical views except the deleted view. So, we see that VIDMvDA is faster in deletion as well. Following the same example of the AwA dataset, we can see that VIDMvDA decrementally unlearns 5 views in just 4.4 *minutes*, whereas, MvDA takes more than 4 *hours* to unlearn the same.

  The deletion operation takes much less time than the addition. This is mainly due to the difference in computation of the within-class scatter. In the case of addition, every submatrix is either computed newly or is updated from old submatrices, and as the number of views increases, the count of submatrices grows with it. Whereas for deletion, the submatrices associated with the deleted views are simply deleted, and computations are only needed for the remaining submatrices, which reduce in numbers after each deletion.



(a) Handwritten digits dataset

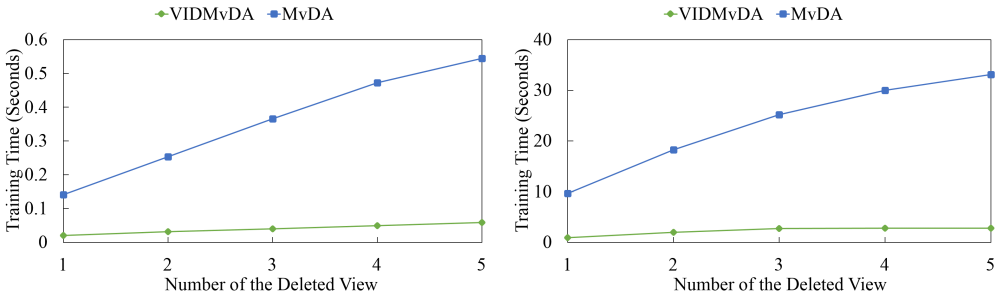
(b) Caltech-7 dataset



(c) AwA dataset

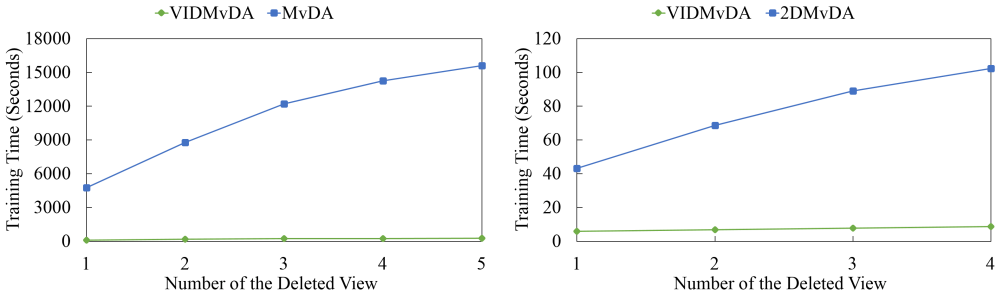
(d) IMPART dataset

Figure 7.5: Addition of views : (a)-(c) training time of VIDMvDA vs. MvDA. (d) training time of VIDMvDA vs. 2DMvDA



(a) Handwritten digits dataset

(b) Caltech-7 dataset



(c) AwA dataset

(d) IMPART dataset

Figure 7.6: Deletion of views : (a)-(c) training time of VIDMvDA vs. MvDA. (d) training time of VIDMvDA vs. 2DMvDA

Table 7.3: Comparison of training time of VIDMvDA and 2DMvDA. First column shows the number of the view that was added or deleted.

View No.	MSSpoof		Stereo Face		ORL	
	VIDMvDA	**2DMvDA**	**VIDMvDA**	**2DMvDA**	**VIDMvDA**	**2DMvDA**
Addition of Views						
1	49.93	49.93	15.34	15.34	0.24	0.24
2	64.97	108.25	22.46	33.29	0.76	1.09
3	-	-	-	-	1.21	1.86
Deletion of Views						
1	5.33	43.86	4.81	14.11	0.04	0.19
2	-	-	-	-	0.32	1.17

Table 7.4: Comparison of memory requirements (in MBs) of VIDMvDA and MvDA.

	Handwritten Digits	**Caltech-7**	**AwA**
VIDMvDA	0.34	2.57	11.07
MvDA	23.38	292.55	7535.07

Table 7.5: Comparison of memory requirements (in GBs) of VIDMvDA and 2DMvDA.

	IMPART	**MSSpoof**	**Stereo Face**	**ORL**
VIDMvDA	1.29	0.38	0.50	0.02
2DMvDA	5.44	7.61	3.94	0.18

7.2.2.5 Comparison of Memory Requirements

- **RQ5 :** Can this method reduce memory requirements?

- **Experiment :** We record the total amount of storage taken by each model after training on all six views.

- **Discussion :** As VIDMvDA only stores the existing model and the new data samples instead of storing all the historical data along with the model, it requires less memory for training than the batch method. This is reflected in the recorded memory requirements. Table 7.4 and Table 7.5 present the records of memory requirements of these methods on the 1D and 2D datasets, respectively.

  We see that VIDMvDA requires much less memory than the batch methods. The difference between the memory requirements of these methods is again highlighted in the case of the AwA dataset. As it is a very large dataset, the required storage space is also significant. However, as VIDMvDA does not need to store all the historical data, it takes only 11 MBs, while MvDA takes around 7.5 GBs of memory.

7.3 Summary

In this chapter, we have presented View Incremental Decremental Multi-view Discriminant Analysis (VIDMvDA), an incremental method for multi-view data that supports the addition and deletion of views over time. This method is based on discriminant analysis and provides closed-form solutions for incremental learning and decremental unlearning. In the case of increment, it produces the updated discriminant subspace using only the existing model and the newly added views. It also supports the deletion of any existing views and produces the updated subspace after the deletion using only the existing model. This provision allows it to train in much less time than the batch method. Also, as VIDMvDA does not need to store all the historical data, it requires much less memory than its batch counterpart, MvDA.

Through experiments, we have also shown that, while taking less time and memory, VIDMvDA constructs a similar discriminative subspace as MvDA. Hence, it also performs at par or sometimes better than MvDA in terms of classification accuracy. We have also proven that the order of addition or deletion of views does not affect the subspace construction of VIDMvDA.

8

Summary and Future Directions

This thesis has contributed incremental-decremental learning methods and one batch method in different niches of supervised learning. These methods are based on Multi-view Discriminant Analysis. We have identified a common framework to obtain the incremental formulations. This framework consists of four steps- (i) Updating the number of data samples, (ii) Updating the means, (iii) Updating the within-class scatter matrix, and (iv) Updating the between-class scatter matrix. This framework has been followed to implement the data increment and view increment. The batch method also follows this framework. However, it only implements it to train the model from scratch. We present a chapter-wise summary that highlights key points of these methods followed by a brief discussion and future scope in this direction.

8.1 Multi-view Incremental Decremental Discriminant Analysis

Multi-view Incremental Decremental Discriminant Analysis (MvIDDA) is an incremental decremental method for 1D multi-view data based on discriminant analysis. MvIDDA follows the framework mentioned above to update the model after each increment/decrement. It supports the data sample update in two forms- sequential and chunk. It also supports the addition of previously unseen classes of data. MvIDDA incrementally updates a common discriminant space for multi-view data using only the existing model and the new data samples. It is also capable of unlearning the data samples using olny the existing model and the data samples to be deleted. As MvIDDA is the first incremental decremental multi-view method in the supervised learning domain, it was compared against batch multi-view MvDA and single-view incremental ILDA on various parameters. The comparisons with MvDA show that the proposed method is order-independent and can build the same model as MvDA using less time and memory. This improvement is possible because of the incremental nature of MvIDDA. Compared to ILDA, the proposed method was found to have better classification accuracy while taking significantly less time for training. This improvement is due to the multi-view nature of MvIDDA. The sequential formulations of MvIDDA train a model using less memory and are more precise than the chunk formulations in the early stages of increments. Similar trends are found near the end for decrements in data. The chunk formulations train a model using less time than sequential formulations but converge to the optimum a little later than the sequential formulations.

8.2 2D Multi-view Discriminant Analysis

2D Multi-view Discriminant Analysis is a multi-view batch learning method for image datasets. 2DMvDA uses the above-mentioned framework to produce a classification model for 2D data. The minimal time and memory requirements of 2DMvDA make it a suitable candidate for large, image-based datasets as these datasets are inherently two-dimensional and require a lot of resources because of their huge sizes when 1D methods are used. Also, the use of 2D images without vectorization preserves the spatial information in these matrices and, in turn, produces better classification accuracy. We compared this method with MvDA and 2DLDA on classification accuracy, training time, and memory. 2DMvDA proves to be much more efficient and a little better at classification than the other two methods due to the benefit it gets from multiple views and the spatial information from the original 2D form of images.

8.3 2D Multi-view Incremental Decremental Discriminant Analysis

2D Multi-view Incremental Decremental Discriminant Analysis (2DMvIDDA) is an incremental-decremental method based on 2DMvDA. It supports incremental learning and decremental unlearning of data samples without using historical data. It only needs the existing model and the data samples to be added or deleted to update the model. 2DMvIDDA follows the four-step framework to update the trained model after each increment or decrement in data. It is also an umbrella method for the methods based on discriminant analysis, namely- 2DMvDA, MvIDDA, MvDA, 2DLDA, and ILDA. We have shown that the formulations of these methods can be obtained from the formulations of 2DMvIDDA. Also, the decremental learning formulations of 2DMvIDDA can be used to form decremental versions of MvIDDA and ILDA. 2DMvIDDA obtains the same discriminant subspace as its batch counterpart, 2DMvDA, and it is invariant to the order of addition or deletion of the data samples. As it benefits from the three paradigms- 2D learning, multi-view learning, and incremental learning- it is the fastest and most accurate method that requires the least amount of memory among the other methods based on discriminant analysis.

8.4 View Incremental Decremental Multi-view Discriminant Analysis

View Incremental Decremental Multi-view Discriminant Analysis (VIDMvDA) is an incremental method for multi-view data that supports the addition and deletion of views. VIDMvDA follows the four-step framework to update its model after the addition or deletion of views from the dataset. It uses only the existing model and the views to be added or deleted to update a trained model. As this is the only view-incremental supervised method for multi-view data, it was compared against its batch counterpart MvDA for its performance. The experiments on three datasets show that VIDMvDA constructs the same discriminant subspace as MvDA. As a result, the classification accuracy of both methods is also the same. Being an incremental method, VIDMvDA also requires much less training time and memory than MvDA. The same formulations of VIDMvDA can be used for 2D datasets.

8.5 Discussion and Future Directions

Dealing with Missing Test Data

The methods presented in this thesis solve for the linear transforms of all views simultaneously. However, the view-wise projection vectors thus found are independent and can be used separately to project the test data samples. This property can be used to classify the test data samples that have missing views. If a view of a test data sample is missing, we can still use the projection vectors for the rest of the views to project it onto the common discriminant subspace for classification. Though it does not solve the missing view problem in the training set, it can be used for the test data with incomplete views.

True Incremental Methods

The incremental methods presented in this thesis are incremental in the sense that they do not require the older data samples to update the model. As a result, the computation time and memory are reduced by a huge factor. However, these methods still have to compute the eigenvectors after each increment. To be truly incremental, these methods may employ techniques that can update or approximate the eigenvalues without computing those again. This will reduce the computation time and memory even further as these methods will not have to store scatter matrices, and work with the projection matrix alone.

Multi-view Incremental methods for Other Paradigms

The methods presented in this thesis are multi-view incremental methods in the supervised domain. As seen in chapter 2, other than those presented in this thesis, there are only four multi-view incremental methods in the literature. The methods belonging to the *data sample increment* category are in the active learning domain [58, 59]. Out of the other two that support *view increment*, one belongs to the supervised learning domain [60] and the other to the unsupervised learning domain [61]. So, there is a broad scope for multi-view incremental methods in the traditional machine learning domains such as- supervised learning, semi-supervised learning, active learning, and unsupervised learning. The opportunities are also available in the other domains such as- time-series analysis, reinforcement learning, and deep learning. These domains have different challenges and hence, demand different methodologies. For example, the time-series data is chronological and needs to be treated differently than the other types of data. The existing multi-view incremental methods may not be suitable for tasks such as shapelet finding [88] or motif discovery [89]. Hence, we see a scope to develop specialized methods in these domains.

Appendix

A. $\mathbf{S}'_B$ - Generic Derivation

The between-class scatter is dependent on the class mean and the total mean, both of which get updated in all four cases. Hence, the between-class scatter is recalculated at each increment. This is a generic derivation for all cases of all the methods presented in this thesis (MvIDDA, 2DMvDA, 2DMvIDDA, and VIDMvDA). Appropriate notations shall be used in the case of minor variations, like new class or new view (e.g., c' in place c or v' in place of v).

We start with the equation in the projected space and arrive at the equation in the original space.

$$
\begin{aligned}
\mathbf{S}'_B &= \sum_{i=1}^{c} n'_i(\mathbf{m}'_i - \mathbf{m}')(\mathbf{m}'_i - \mathbf{m}')^T \\
&= \sum_{i=1}^{c} n'_i \mathbf{m}'_i \mathbf{m}'^T_i - \sum_{i=1}^{c} n'_i \mathbf{m}'_i \mathbf{m}'^T - \sum_{i=1}^{c} n'_i \mathbf{m}' \mathbf{m}'^T_i + \sum_{i=1}^{c} n'_i \mathbf{m}' \mathbf{m}'^T \\
&= \sum_{i=1}^{c} n'_i \mathbf{m}'_i \mathbf{m}'^T_i - n' \mathbf{m}' \mathbf{m}'^T - \mathbf{m}'(n' \mathbf{m}'^T) + n' \mathbf{m}' \mathbf{m}'^T \\
&= \sum_{i=1}^{c} n'_i \mathbf{m}'_i \mathbf{m}'^T_i - n' \mathbf{m}' \mathbf{m}'^T \\
&= \sum_{i=1}^{c} n'_i \left(\sum_{j=1}^{v'} \frac{n'_{ij}}{n'_i} \mathbf{W}_j^T \mathbf{m}'^{(x)}_{ij} \right) \left(\sum_{r=1}^{v'} \frac{n'_{ir}}{n'_i} \mathbf{W}_r^T \mathbf{m}'^{(x)}_{ir} \right)^T - n' \left(\sum_{j=1}^{v'} \sum_{i=1}^{c} \frac{n'_{ij}}{n'} \mathbf{W}_j^T \mathbf{m}'^{(x)}_{ij} \right) \left(\sum_{r=1}^{v'} \sum_{i=1}^{c} \frac{n'_{ij}}{n'} \mathbf{W}_r^T \mathbf{m}'^{(x)}_{ir} \right)^T \\
&= \sum_{j=1}^{v'} \sum_{r=1}^{v'} \mathbf{W}_j^T \mathbf{D}'_{jr} \mathbf{W}_r
\end{aligned}
$$

Where,

$$
\mathbf{D}'_{jr} = \sum_{i=1}^{c} \frac{n'_{ij}\, n'_{ir}}{n'_i} \mathbf{m}'^{(x)}_{ij} \mathbf{m}'^{(x)^T}_{ir} - \frac{1}{n'} \left(\sum_{i=1}^{c} n'_{ij}\, \mathbf{m}'^{(x)}_{ij} \right) \left(\sum_{i=1}^{c} n'_{ir}\, \mathbf{m}'^{(x)^T}_{ir} \right)
$$

B. MvIDDA: $\mathbf{S}'_W$ - Sequential Increment and Existing Class

In the case of *sequential increment and existing class*, the within-class scatter matrix ($\mathbf{S}_W$) is updated to include the new data sample that belongs to an already existing class E. Hence, we only have to update the scatter of class E. This update does not affect the scatter of other classes. Hence, we keep the class-wise scatters of other classes as they are and add the updated scatter of class E to it. However, this scatter is not computed from scratch. We derive the formula to obtain the updated class-wise scatter of class E without using the old data samples.

$$
\begin{aligned}
\mathbf{S}'_W &= \sum_{i=1,i\neq E}^{c} \mathbf{S}_{W_i} + \mathbf{S}'_{W_E} \\
&= \sum_{i=1,i\neq E}^{c} \mathbf{S}_{W_i} + \sum_{j=1}^{v}\sum_{k=1}^{n_{Ej}} \left(\mathbf{y}_{Ejk} - \mathbf{m}'_E\right)\left(\mathbf{y}_{Ejk} - \mathbf{m}'_E\right)^T + \sum_{j=1}^{v} \left(\mathbf{y}'_j - \mathbf{m}'_E\right)\left(\mathbf{y}'_j - \mathbf{m}'_E\right)^T \\
&= \sum_{i=1,i\neq E}^{c} \mathbf{S}_{W_i} + \sum_{j=1}^{v}\sum_{k=1}^{n_{Ej}} \left(\mathbf{y}_{Ejk} - \mathbf{m}_E - \frac{v(\bar{\mathbf{m}}_E - \mathbf{m}_E)}{n_E + v}\right)\left(\mathbf{y}_{Ejk} - \mathbf{m}_E - \frac{v(\bar{\mathbf{m}}_E - \mathbf{m}_E)}{n_E + v}\right)^T \\
&\qquad + \sum_{j=1}^{v} \left(\mathbf{y}'_j - \mathbf{m}_E - \frac{v(\bar{\mathbf{m}}_E - \mathbf{m}_E)}{n_E + v}\right)\left(\mathbf{y}'_j - \mathbf{m}_E - \frac{v(\bar{\mathbf{m}}_E - \mathbf{m}_E)}{n_E + v}\right)^T \\
&= \sum_{i=1,i\neq E}^{c} \mathbf{S}_{W_i} + \mathbf{S}_{W_E} + \frac{v\, n_E}{(n_E + v)}\left(\bar{\mathbf{m}}_E - \mathbf{m}_E\right)\left(\bar{\mathbf{m}}_E - \mathbf{m}_E\right)^T + \sum_{j=1}^{v} \left(\mathbf{y}'_j - \bar{\mathbf{m}}_E\right)\left(\mathbf{y}'_j - \bar{\mathbf{m}}_E\right)^T \\
&= \mathbf{S}_W + \frac{v\, n_E}{(n_E + v)}\left(\bar{\mathbf{m}}_E - \mathbf{m}_E\right)\left(\bar{\mathbf{m}}_E - \mathbf{m}_E\right)^T + \sum_{j=1}^{v} \left(\mathbf{y}'_j - \bar{\mathbf{m}}_E\right)\left(\mathbf{y}'_j - \bar{\mathbf{m}}_E\right)^T \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \left[\mathbf{S}_{jr} + \frac{n_E}{v(n_E + v)}\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}^{(x)}\right)\left(\bar{\mathbf{x}}_r - \mathbf{m}_{Er}^{(x)}\right)^T - \frac{1}{v}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_r^T\right]\mathbf{W}_r + \sum_{j=1}^{v} \mathbf{W}_j^T\bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T\mathbf{W}_j \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T\mathbf{S}'_{jr}\mathbf{W}_r \\
&= \mathbf{W}^T\mathbf{S}'\mathbf{W}
\end{aligned}
$$

Where,

$$
\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jr} + \frac{n_E}{v(n_E+v)}\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)^T + \bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T - \frac{1}{v}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T & j = r \\ \mathbf{S}_{jr} + \frac{n_E}{v(n_E+v)}\left(\bar{\mathbf{x}}_j - \mathbf{m}_{Ej}{}^{(x)}\right)\left(\bar{\mathbf{x}}_r - \mathbf{m}_{Er}{}^{(x)}\right)^T - \frac{1}{v}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_r^T & j \neq r \end{cases}
$$

C. MvIDDA: $\mathbf{S}'_W$ - Sequential Increment and New Class

In the case of *sequential increment and new class*, as the new data sample belongs to a new class (denoted as class N), we do not have to make changes in the scatters of any of the old classes. We only add the scatter of the new sample in the old scatter matrix. As this scatter of class N does not need any older data samples, the derivation is straightforward.

$$
\begin{aligned}
\mathbf{S}'_W &= \mathbf{S}_W + \sum_{j=1}^{v} \left(\mathbf{y}'_j - \mathbf{m}_N\right)\left(\mathbf{y}'_j - \mathbf{m}_N\right)^T \\
&= \mathbf{S}_W + \sum_{j=1}^{v} \mathbf{y}'_j\mathbf{y}'_j{}^T - \sum_{j=1}^{v} \mathbf{y}'_j\mathbf{m}_N^T - \sum_{j=1}^{v} \mathbf{m}_N\mathbf{y}'_j{}^T + \sum_{j=1}^{v} \mathbf{m}_N\mathbf{m}_N^T \\
&= \mathbf{S}_W + \sum_{j=1}^{v} \mathbf{y}'_j\mathbf{y}'_j{}^T - v\mathbf{m}_N\mathbf{m}_N^T
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{S}'_W &= \mathbf{S}_W + \sum_{j=1}^{v} \mathbf{W}_j^T \bar{\mathbf{x}}_j \bar{\mathbf{x}}_j^T \mathbf{W}_j - v \left(\frac{1}{v} \sum_{j=1}^{v} \mathbf{W}_j^T \mathbf{m}_{Nj}^{(x)} \right) \left(\frac{1}{v} \sum_{r=1}^{v} \mathbf{W}_r^T \mathbf{m}_{Nr}^{(x)} \right)^T \\
&= \mathbf{S}_W + \sum_{j=1}^{v} \mathbf{W}_j^T \bar{\mathbf{x}}_j \bar{\mathbf{x}}_j^T \mathbf{W}_j - v \left(\frac{1}{v} \sum_{j=1}^{v} \mathbf{W}_j^T \bar{\mathbf{x}}_j \right) \left(\frac{1}{v} \sum_{r=1}^{v} \mathbf{W}_r^T \bar{\mathbf{x}}_r \right)^T \\
&= \sum_{j=1}^{v} \sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{S}_{jr} \mathbf{W}_r + \sum_{j=1}^{v} \mathbf{W}_j^T \bar{\mathbf{x}}_j \bar{\mathbf{x}}_j^T \mathbf{W}_j - \sum_{j=1}^{v} \sum_{r=1}^{v} \frac{1}{v} \mathbf{W}_j^T \bar{\mathbf{x}}_j \bar{\mathbf{x}}_r^T \mathbf{W}_r \\
&= \sum_{j=1}^{v} \sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{S}'_{jr} \mathbf{W}_r = \mathbf{W}^T \mathbf{S}' \mathbf{W}
\end{aligned}
$$

Where,

$$
\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jr} + \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_j^T - \frac{1}{v} \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_j^T & j = r \\ \mathbf{S}_{jr} - \frac{1}{v} \bar{\mathbf{x}}_j \, \bar{\mathbf{x}}_r^T & j \neq r \end{cases}
$$

D. MvIDDA: $\mathbf{S}'_W$ - Chunk Increment

In the case of *chunk increment and existing class*, we have to update the scatters of all existing classes to which new samples have been added denoted by CH. The classes to which no new data samples were added are denoted by UC. The update for *chunk increment and new class* is obtained in a similar way. Hence, a separate derivation for the same is not provided. In the case of a new class, the scatters of new classes are also included in CH. We keep the scatters of classes in UC as they are and update the scatter of the classes in set CH. We derive the equation for updating the scatter matrix without using the old data samples.

$$
\begin{aligned}
\mathbf{S}'_W &= \sum_{i \in \mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i \in \mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \left(\mathbf{y}_{ijk} - \mathbf{m}'_i \right) \left(\mathbf{y}_{ijk} - \mathbf{m}'_i \right)^T + \sum_{i \in \mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i \right) \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i \right)^T \\
&= \sum_{i \in \mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i \in \mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \left(\mathbf{y}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i (\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i} \right) \left(\mathbf{y}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i (\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i} \right)^T \\
&\quad + \sum_{i \in \mathcal{CH}} \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i (\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i} \right) \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i (\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i} \right)^T \\
&= \mathbf{S}_W + \sum_{i \in \mathcal{CH}} \frac{1}{n_i + \bar{n}_i} \Bigg[n_i (\bar{n}_i)^2 \sum_{j=1}^{v} \left(\bar{\mathbf{m}}_i - \mathbf{m}_i \right) \left(\bar{\mathbf{m}}_i - \mathbf{m}_i \right)^T + n_i^2 \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i \right) \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i \right)^T \\
&\qquad + (2 n_i \bar{n}_i + \bar{n}_i^2) \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i \right) \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i \right)^T \Bigg] \\
&= \mathbf{S}_W + \sum_{i \in \mathcal{CH}} \left[\frac{v (\bar{n}_i)^2 n_i + \bar{n}_i (n_i)^2}{(n_i + \bar{n}_i)^2} \left(\bar{\mathbf{m}}_i - \mathbf{m}_i \right) \left(\bar{\mathbf{m}}_i - \mathbf{m}_i \right)^T + \sum_{j=1}^{v} \sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i \right) \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i \right)^T \right]
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{S}'_W &= \sum_{j=1}^{v}\sum_{r=1}^{v}\mathbf{W}_j^T\left[\mathbf{S}_{jr}+\sum_{i=1}^{c}\left[\frac{v\bar{n}_i+n_i}{\bar{n}_i n_i(n_i+\bar{n}_i)^2}\left(\mathbf{a}\right)\left(\mathbf{b}\right)^T-\frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ir}^{(x)^T}\right]\right]\mathbf{W}_r+\sum_{j=1}^{v}\mathbf{W}_j^T\left[\sum_{i=1}^{c}\sum_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_j^T\right]\mathbf{W}_j\\
&= \sum_{j=1}^{v}\sum_{r=1}^{v}\mathbf{W}_j^T\mathbf{S}'_{jr}\mathbf{W}_r\\
&= \mathbf{W}^T\mathbf{S}'\mathbf{W}
\end{aligned}
$$

Where, $\mathbf{a}=(\bar{n}_{ij}n_i\bar{\mathbf{m}}_{ij}^{(x)}-\bar{n}_i n_{ij}\mathbf{m}_{ij}^{(x)})$ and $\mathbf{b}=(\bar{n}_{ir}n_i\bar{\mathbf{m}}_{ir}^{(x)}-\bar{n}_i n_{ir}\mathbf{m}_{ir}^{(x)})$
and,

$$
\mathbf{S}'_{jr}=\begin{cases}\mathbf{S}_{jj}+\sum\limits_{i\in\mathcal{CH}}\left[\frac{v\bar{n}_i+n_i}{\bar{n}_i n_i(n_i+\bar{n}_i)^2}\mathbf{a}\mathbf{a}^T-\frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ij}^{(x)^T}+\sum\limits_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T\right] & j=r\\ \mathbf{S}_{jr}+\sum\limits_{i\in\mathcal{CH}}\left[\frac{v\bar{n}_i+n_i}{\bar{n}_i n_i(n_i+\bar{n}_i)^2}\mathbf{a}\mathbf{b}^T-\frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{m}}_{ij}^{(x)}\bar{\mathbf{m}}_{ir}^{(x)^T}\right] & j\neq r\end{cases}
$$

E. 2DMvDA: $\mathbf{S}'_W$ - Within-class Scatter

The within-class scatter of 2DMvDA is given in terms of the projected data samples. We reformulate it in terms of the original data samples so as to convert the optimization function in Eq. 2.1 to trace-ratio formulation in Eq. 2.4,

$$
\begin{aligned}
\mathbf{S}_W^y &= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\left(\mathbf{Y}_{ijk}-\mathbf{M}_i\right)\left(\mathbf{Y}_{ijk}-\mathbf{M}_i\right)^T\\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\left(\mathbf{Y}_{ijk}\mathbf{Y}_{ijk}^T-\mathbf{Y}_{ijk}\mathbf{M}_i^T-\mathbf{M}_i\mathbf{Y}_{ijk}^T+\mathbf{M}_i\mathbf{M}_i^T\right)\\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\mathbf{Y}_{ijk}\mathbf{Y}_{ijk}^T-\sum_{i=1}^{c}n_i\mathbf{M}_i\mathbf{M}_i^T-\sum_{i=1}^{c}n_i\mathbf{M}_i\mathbf{M}_i^T+\sum_{i=1}^{c}n_i\mathbf{M}_i\mathbf{M}_i^T\\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\mathbf{Y}_{ijk}\mathbf{Y}_{ijk}^T-\sum_{i=1}^{c}n_i\mathbf{M}_i\mathbf{M}_i^T\\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\mathbf{W}_j^T\mathbf{X}_{ijk}\mathbf{X}_{ijk}^T\mathbf{W}_j-\sum_{i=1}^{c}n_i\left(\frac{1}{n_i}\sum_{j=1}^{v}n_{ij}\mathbf{W}_j^T\mathbf{M}_{ij}^{(\mathbf{X})}\right)\left(\frac{1}{n_i}\sum_{r=1}^{v}n_{ir}\mathbf{M}_{ir}^{(\mathbf{X})T}\mathbf{W}_r\right)\\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\mathbf{W}_j^T\mathbf{X}_{ijk}\mathbf{X}_{ijk}^T\mathbf{W}_j-\sum_{j=1}^{v}\sum_{r=1}^{v}\left(\sum_{i=1}^{c}n_{ij}\mathbf{W}_j^T\mathbf{M}_{ij}^{(\mathbf{X})}\right)\left(\sum_{i=1}^{c}\frac{n_{ir}}{n_i}\mathbf{M}_{ir}^{(\mathbf{X})T}\mathbf{W}_r\right)\\
&= \sum_{j=1}^{v}\sum_{r=1}^{v}\mathbf{W}_j^T\left[\sum_{i=1}^{c}\left(\sum_{k=1}^{n_{ij}}\mathbf{X}_{ijk}\mathbf{X}_{ijk}^T-\frac{n_{ij}n_{ij}}{n_i}\mathbf{M}_{ij}^{(\mathbf{X})}\mathbf{M}_{ij}^{(\mathbf{X})T}\right)\right]\mathbf{W}_r\\
&= \sum_{j=1}^{v}\sum_{r=1}^{v}\mathbf{W}_j^T\mathbf{S}_{jr}\mathbf{W}_r
\end{aligned}
$$

where,

$$
\mathbf{S}_{jr}=\begin{cases}\sum_{i=1}^{c}\left(\sum_{k=1}^{n_{ij}}\mathbf{X}_{ijk}\mathbf{X}_{ijk}^T-\frac{n_{ij}n_{ij}}{n_i}\mathbf{M}_{ij}^{(\mathbf{X})}\mathbf{M}_{ij}^{(\mathbf{X})T}\right) & \cdots j=r\\ -\sum_{i=1}^{c}\frac{n_{ij}n_{ir}}{n_i}\mathbf{M}_{ij}^{(\mathbf{X})}\mathbf{M}_{ir}^{(\mathbf{X})T} & \cdots j\neq r\end{cases}
$$

F. 2DMvIDDA: $\mathbf{S}'_W$ - Within-class Scatter

For 2DMvIDDA, the derivation of incremental updates is similar to the MvIDDA chunk increment derivation given in Appendix D. Hence, derivation of only the decremental formulation is given here. In the case of *decrement*, the scatters of classes from which data samples have been removed are updated. $\mathcal{CH}$ denotes the classes that got some data samples removed from them, and $\mathcal{UC}$ denotes those left unchanged.

$$
\begin{aligned}
\mathbf{S}'_W &= \sum_{i\in\mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk}-\mathbf{M}'_i\right)\left(\mathbf{Y}_{ijk}-\mathbf{M}'_i\right)^T - \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{Y}}_{ijk}-\mathbf{M}'_i\right)\left(\bar{\mathbf{Y}}_{ijk}-\mathbf{M}'_i\right)^T \\
&= \sum_{i\in\mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk}-\mathbf{M}_i-\frac{\bar{n}_i(\mathbf{M}_i-\bar{\mathbf{M}}_i)}{n_i-\bar{n}_i}\right)\left(\mathbf{Y}_{ijk}-\mathbf{M}_i-\frac{\bar{n}_i(\mathbf{M}_i-\bar{\mathbf{M}}_i)}{n_i-\bar{n}_i}\right)^T \\
&\quad - \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{Y}}_{ijk}-\mathbf{M}_i-\frac{\bar{n}_i(\mathbf{M}_i-\bar{\mathbf{M}}_i)}{n_i-\bar{n}_i}\right)\left(\bar{\mathbf{Y}}_{ijk}-\mathbf{M}_i-\frac{\bar{n}_i(\mathbf{M}_i-\bar{\mathbf{M}}_i)}{n_i-\bar{n}_i}\right)^T \\
&= \sum_{i\in\mathcal{UC}} \mathbf{S}_{W_i} + \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \left(\mathbf{Y}_{ijk}-\mathbf{M}_i\right)\left(\mathbf{Y}_{ijk}-\mathbf{M}_i\right)^T + \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \frac{\bar{n}_i^2}{(n_i-\bar{n}_i)^2}\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)^T \\
&\quad - \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \frac{1}{(n_i-\bar{n}_i)^2}\left[n_i\left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)+\bar{n}_i\left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)^T\right]\left[n_i\left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)+\bar{n}_i\left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)^T\right]^T \\
&= \mathbf{S}_W + \sum_{i\in\mathcal{CH}} \frac{\bar{n}_i^2 n_i - n_i^2\bar{n}_i}{(n_i-\bar{n}_i)^2}\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)^T - \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)\left(\bar{\mathbf{Y}}_{ijk}-\bar{\mathbf{M}}_i\right)^T \\
&= \mathbf{S}_W + \sum_{i\in\mathcal{CH}} \frac{\bar{n}_i n_i}{(n_i-\bar{n}_i)}\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)\left(\mathbf{M}_i-\bar{\mathbf{M}}_i\right)^T - \sum_{i\in\mathcal{CH}}\sum_{j=1}^{v}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{Y}}_{ijk}\bar{\mathbf{Y}}_{ijk}^T + \sum_{i\in\mathcal{CH}} \bar{n}_i\bar{\mathbf{M}}_i\bar{\mathbf{M}}_i^T \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T\left[\mathbf{S}_{jr} - \sum_{i\in\mathcal{CH}}\left[\frac{1}{\bar{n}_i n_i(n_i-\bar{n}_i)}(\mathbf{E})(\mathbf{F})^T + \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})}\bar{\mathbf{M}}_{ir}^{(\mathbf{X})^T}\right]\right]\mathbf{W}_r \\
&\quad - \sum_{j=1}^{v}\mathbf{W}_j^T\left[\sum_{i\in\mathcal{CH}}\sum_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{X}}_{ijk}\bar{\mathbf{X}}_{ijk}^T\right]\mathbf{W}_j \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v}\mathbf{W}_j^T\mathbf{S}'_{jr}\mathbf{W}_r = \mathbf{W}^T\mathbf{S}'\mathbf{W}
\end{aligned}
$$

Where, $\mathbf{E} = (\bar{n}_i n_{ij}\mathbf{M}_{ij}^{(\mathbf{X})} - n_i\bar{n}_{ij}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})})$ and $\mathbf{F} = (\bar{n}_i n_{ir}\mathbf{M}_{ir}^{(\mathbf{X})} - n_i\bar{n}_{ir}\bar{\mathbf{M}}_{ir}^{(\mathbf{X})})$

and,

$$
\mathbf{S}'_{jr} = \begin{cases} \mathbf{S}_{jj} - \sum_{i\in\mathcal{CH}}\left[\frac{1}{\bar{n}_i n_i(n_i-\bar{n}_i)}\mathbf{E}\mathbf{E}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})}\bar{\mathbf{M}}_{ij}^{(X)^T} + \sum_{k=1}^{\bar{n}_{ij}}\bar{\mathbf{X}}_{ijk}\bar{\mathbf{X}}_{ijk}^T\right] & j = r \\ \mathbf{S}_{jr} - \sum_{i\in\mathcal{CH}}\left[\frac{1}{\bar{n}_i n_i(n_i-\bar{n}_i)}\mathbf{E}\mathbf{F}^T + \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\bar{\mathbf{M}}_{ij}^{(\mathbf{X})}\bar{\mathbf{M}}_{ir}^{(\mathbf{X})^T}\right] & j \neq r \end{cases}
$$

G. VIDMvDA: $\mathbf{S}'_W$ - Within-class Scatter

For VIDMvDA, in the case of *increment*, the scatters are updated differently for each of the three cases. The derivation of within-class scatter is given below. The decremental unlearning formulation is also derived in a similar fashion. Here, the scatters of all classes are updated because each new view contains all data samples from all the classes. The final equation in the original space is split into three cases according to the value of j and r as given in section 7.1.1.3.

$$
\begin{aligned}
\mathbf{S}'_W &= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \left(\mathbf{y}_{ijk} - \mathbf{m}'_i\right)\left(\mathbf{y}_{ijk} - \mathbf{m}'_i\right)^T + \sum_{i=1}^{c}\sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i\right)\left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}'_i\right)^T \\
&= \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}} \left(\mathbf{y}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i(\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i}\right)\left(\mathbf{y}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i(\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i}\right)^T \\
&\quad + \sum_{i=1}^{c}\sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i(\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i}\right)\left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i - \frac{\bar{n}_i(\bar{\mathbf{m}}_i - \mathbf{m}_i)}{n_i + \bar{n}_i}\right)^T \\
&= \mathbf{S}_W + \sum_{i=1}^{c} \frac{1}{(n_i + \bar{n}_i)^2}\left[n_i(\bar{n}_i)^2 \left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T + n_i^2 \sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i\right)\left(\bar{\mathbf{y}}_{ijk} - \mathbf{m}_i\right)^T \right. \\
&\qquad \left. + (2n_i\bar{n}_i + \bar{n}_i^2) \sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i\right)\left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i\right)^T \right] \\
&= \mathbf{S}_W + \sum_{i=1}^{c}\left[\frac{\bar{n}_i n_i}{n_i + \bar{n}_i}\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T + \sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i\right)\left(\bar{\mathbf{y}}_{ijk} - \bar{\mathbf{m}}_i\right)^T \right] \\
&= \mathbf{S}_W + \sum_{i=1}^{c}\left[\frac{\bar{n}_i n_i}{n_i + \bar{n}_i}\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)\left(\bar{\mathbf{m}}_i - \mathbf{m}_i\right)^T + \sum_{j\in\mathcal{A}}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{y}}_{ijk}\bar{\mathbf{y}}_{ijk}^T - \bar{n}_i\bar{\mathbf{m}}_i\bar{\mathbf{m}}_i^T \right] \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \left[\mathbf{S}_{jr} + \sum_{i=1}^{c} \frac{\bar{n}_i n_{ij} n_{ir}}{n_i(n_i + \bar{n}_i)}\, \mathbf{m}_{ij}{}^{(\mathbf{x})}\mathbf{m}_{ir}{}^{(\mathbf{x})^T} \right] \mathbf{W}_r + \sum_{j\in\mathcal{A}} \mathbf{W}_j^T \left[\sum_{i=1}^{c}\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T \right] \mathbf{W}_j \\
&\quad - \sum_{j\in\mathcal{A}}\sum_{r\in\mathcal{A}} \mathbf{W}_j^T \left[\sum_{i=1}^{c} \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i} \bar{\mathbf{m}}_{ij}^{(x)} \bar{\mathbf{m}}_{ir}^{(x)^T} \right] \mathbf{W}_r \\
&= \sum_{j=1}^{v}\sum_{r=1}^{v} \mathbf{W}_j^T \mathbf{S}'_{jr} \mathbf{W}_r = \mathbf{W}^T \mathbf{S}' \mathbf{W}
\end{aligned}
$$

Where,

$$
\mathbf{S}'_{jr} = \begin{cases}
-\sum_{i=1}^{c} \frac{\bar{n}_{ij}\bar{n}_{ir}}{\bar{n}_i}\, \bar{\mathbf{m}}_{ij}^{(\mathbf{x})}\, \bar{\mathbf{m}}_{ir}^{(\mathbf{x})^T} & \text{case 1} \\
\sum_{i=1}^{c} \left[\sum_{k=1}^{\bar{n}_{ij}} \bar{\mathbf{x}}_{ijk}\bar{\mathbf{x}}_{ijk}^T - \frac{\bar{n}_{ij}\bar{n}_{ij}}{\bar{n}_i}\, \bar{\mathbf{m}}_{ij}^{(\mathbf{x})}\, \bar{\mathbf{m}}_{ij}^{(\mathbf{x})^T} \right] & \text{case 2} \\
\mathbf{S}_{jr} + \sum_{i=1}^{c} \frac{n'_i - n_i}{n_i n'_i}\, n_{ij} n_{ir}\, \mathbf{m}_{ij}{}^{(\mathbf{x})}\mathbf{m}_{ir}{}^{(\mathbf{x})^T} & \text{case 3}
\end{cases}
$$

Bibliography

[1] I. Chingovska, N. Erdogmus, A. Anjos, S. Marcel, Face recognition systems under spoofing attacks, in: Face Recognition Across the Imaging Spectrum, Springer, 2016, pp. 165–194. doi:https://doi.org/10.1007/978-3-319-28501-6_8.

[2] V. R. de Sa, D. H. Ballard, Category learning through multimodality sensing, Neural Computation 10 (5) (1998) 1097–1117.

[3] J. Zhao, X. Xie, X. Xu, S. Sun, Multi-view learning overview: Recent progress and new challenges, Information Fusion 38 (2017) 43–54.

[4] S. Sun, A survey of multi-view machine learning, Neural computing and applications 23 (7) (2013) 2031–2038.

[5] M. Kan, S. Shan, H. Zhang, S. Lao, X. Chen, Multi-view discriminant analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (1). doi:https://doi.org/10.1109/TPAMI.2015.2435740.

[6] S. Sun, X. Xie, M. Yang, Multiview uncorrelated discriminant analysis, IEEE Transactions on Cybernetics 46 (12) (2016) 3272–3284.

[7] A. Sharma, A. Kumar, H. Daumé III, D. W. Jacobs, Generalized multiview analysis: A discriminative latent space, in: Computer Vision and Pattern Recognition, (2012), pp. 2160–2167.

[8] J. Li, B. Zhang, G. Lu, D. Zhang, Generative multi-view and multi-feature learning for classification, Information Fusion 45 (2019) 215–226. doi:https://doi.org/10.1016/j.inffus.2018.02.005.

[9] G. Chao, S. Sun, Semi-supervised multi-view maximum entropy discrimination with expectation laplacian regularization, Information Fusion 45 (2019) 296–306. doi:https://doi.org/10.1016/j.inffus.2018.03.002.

[10] S. Sun, F. Jin, Robust co-training, International Journal of Pattern Recognition and Artificial Intelligence 25 (7) (2011) 1113–1126. doi:https://doi.org/10.1142/S0218001411008981.

[11] S. Yu, B. Krishnapuram, R. Rosales, R. B. Rao, Bayesian co-training, Journal of Machine Learning Research 12 (2011) 2649–2680.

[12] S. Sun, D. R. Hardoon, Active learning with extremely sparse labeled examples, Neurocomputing 73 (16-18) (2010) 2980–2988. doi:https://doi.org/10.1016/j.neucom.2010.07.007.

[13] A. Kumar, H. Daumé III, A co-training approach for multi-view spectral clustering, in: International Conference on Machine Learning, Omnipress, 2011, pp. 393–400.

[14] L. Houthuys, R. Langone, J. A. K. Suykens, Multi-view kernel spectral clustering, Information Fusion 44 (2018) 46–56. doi:https://doi.org/10.1016/j.inffus.2017.12.002.

[15] A. Kumar, P. Rai, H. Daumé III, Co-regularized multi-view spectral clustering, in: Advances in Neural Information Processing Systems, 2011, pp. 1413–1421.

[16] M. Chen, K. Q. Weinberger, J. Blitzer, Co-training for domain adaptation, in: Advances in Neural Information Processing Systems, 2011, pp. 2456–2464.

[17] Z. Xu, S. Sun, Multi-view transfer learning with adaboost, in: ICTAI, IEEE Computer Society, 2011, pp. 399–402. doi:https://doi.org/10.1109/ICTAI.2011.65.

[18] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the eleventh annual conference on Computational learning theory, 1998, pp. 92–100.

[19] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proceedings of the ninth international conference on Information and knowledge management, 2000, pp. 86–93.

[20] I. Muslea, S. Minton, C. A. Knoblock, Active learning with multiple views, Journal of Artificial Intelligence Research 27 (2000) 203–233.

[21] I. Muslea, S. Minton, C. A. Knoblock, Active learning with strong and weak views: A case study on wrapper induction, in: International Joint Conference on Artificial Intelligence, Vol. 3, 2003, pp. 415–420.

[22] E. Bruno, S. Marchand-Maillet, Multiview clustering: a late fusion approach using latent models, in: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, 2009, pp. 736–737.

[23] C. A. Knoblock, S. Minton, I. Muslea, Active learning with multiple views, arXiv–1110.

[24] M. Wozniak, K. Jackowski, Some remarks on chosen methods of classifier fusion based on weighted voting, in: International Conference on Hybrid Artificial Intelligence Systems, 2009, pp. 541–548.

[25] T. Diethe, D. R. Hardoon, J. Shawe-Taylor, Multiview fisher discriminant analysis, in: NIPS workshop on learning from multiple sources, 2008.

[26] S. Yu, L. Tranchevent, X. Liu, W. Glanzel, J. A. K. Suykens, B. D. Moor, Y. Moreau, Optimized data fusion for kernel k-means clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (5) (2011) 1031–1039.

[27] S. Koço, C. Capponi, A boosting approach to multiview classification with cooperation, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2011, pp. 209–228.

[28] S. N. Pang, S. Ozawa, N. Kasabov, Incremental linear discriminant analysis for classification of data streams, IEEE Transactions on Systems, Man and Cybernetics 35 (5) (2005) 905–914.

[29] N. Kumar, S. Madhavan, Incremental weighted linear discriminant analysis for face recognition, in: Advances in Communication and Computational Technology, 2021, pp. 677–687. doi:https://doi.org/10.1007/978-981-15-5341-7_51.

[30] H. Zhao, P. C. Yuen, J. T. Kwok, A novel incremental principal component analysis and its application for face recognition, IEEE Transactions on Systems, Man and Cybernetics 36 (4) (2006) 873–886.

[31] Y. Liu, L. Chen, C. Zhu, Improved robust tensor principal component analysis via low-rank core matrix, IEEE Journal of Selected Topics in Signal Processing 12 (6) (2018) 1378–1389. doi:https://doi.org/10.1109/JSTSP.2018.2873142.

[32] B. Gu, X. Quan, Y. Gu, V. S. Sheng, G. Zheng, Chunk incremental learning for cost-sensitive hinge loss support vector machine, Pattern Recognition 83 (2018) 196–208. doi:https://doi.org/10.1016/j.patcog.2018.05.023.

[33] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, in: Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, 2000, p. 388âĂŞ394. doi:https://doi.org/10.5555/3008751.3008808.

[34] Y. Chen, J. Xiong, W. Xu, J. Zuo, A novel online incremental and decremental learning algorithm based on variable support vector machine, Cluster Computing 22 (3) (2019) 7435–7445. doi:https://doi.org/10.1007/s10586-018-1772-4.

[35] W.-H. Lee, B. J. Ko, S. Wang, C. Liu, K. K. Leung, Exact incremental and decremental learning for ls-svm, in: IEEE International Conference on Image Processing, 2019, pp. 2334–2338. doi:https://doi.org/10.1109/ICIP.2019.8803291.

[36] R. Kashef, A boosted SVM classifier trained by incremental learning and decremental unlearning approach, Expert Systems with Applications 167 (2021) 114154. doi:https://doi.org/10.1016/j.eswa.2020.114154.

[37] E. López-López, C. V. Regueiro, X. M. Pardo, A. Franco, A. Lumini, Incremental learning techniques within a self-updating approach for face verification in video-surveillance, in: Proceedings of 9th Iberian Conference on Pattern Recognition and Image Analysis, Vol. 11868, Springer, 2019, pp. 25–37. doi:https://doi.org/10.1007/978-3-030-31321-0_3.

[38] M. Oliveira, V. Santos, A. D. Sappa, P. Dias, A. P. Moreira, Incremental texture mapping for autonomous driving, Robotics and Autonomous Systems 84 (2016) 113–128.

[39] Y. Zhang, G. Yin, D. Chen, A dynamic cold-start recommendation method based on incremental graph pattern matching, International Journal of Computational Science and Engineering 18 (1) (2019) 89–100.

[40] M. Turk, A. Pentland, Eigenfaces for recognition, Journal of Cognitive Neuroscience 3 (1) (1991) 71–86. doi:https://doi.org/10.1162/jocn.1991.3.1.71.

[41] P. Zhang, S. Deng, F. Nie, Y. Liu, X. Zhang, Q. Gao, Nuclear-norm based 2DLDA with application to face recognition, Neurocomputing 339 (2019) 94–104. doi:https://doi.org/10.1016/j.neucom.2019.01.066.

[42] C. Wu, Y. Song, Y. Zhang, Multi-view gait recognition using NMF and 2DLDA, Multimedia Tools and Applications 78 (24) (2019) 35789–35811. doi:https://doi.org/10.1007/s11042-019-08153-4.

[43] Z. Cai, W. Wang, 2DLDA-based compound distance for similar online handwritten Tibetan transliteration of the Sanskrit character recognition, in: International Conference on Frontiers in Handwriting Recognition, IEEE Computer Society, 2018, pp. 223–228. doi:http://doi.org/10.1109/ICFHR-2018.2018.00047.

[44] X. Li, J. Gui, P. Li, Randomized kernel multi-view discriminant analysis, in: European Conference on Artificial Intelligence, IEEE Computer Society, 2020, pp. 1276–1284. doi:https://doi.org/10.3233/FAIA200229.

[45] P. Hu, D. Peng, J. Guo, L. Zhen, Local feature based multi-view discriminant analysis, Knowledge-Based Systems 149 (2018) 34–46. doi:https://doi.org/10.1016/j.knosys.2018.02.008.

[46] X. Shu, P. Yuan, H. Jiang, D. Lai, Multi-view uncorrelated discriminant analysis via dependence maximization, Applied Intelligence 49 (2) (2019) 650–660. doi:https://doi.org/10.1007/s10489-018-1271-6.

[47] P. Huang, Q. Ye, Y. Li, G. Yang, Y. Liu, Multi-view learning with robust generalized eigenvalue proximal svm, IEEE Access 7 (2019) 102437–102454.

[48] J. Yin, S. Sun, Multiview uncorrelated locality preserving projection, IEEE Transactions on Neural Networks and Learning Systems 31 (9) (2020) 3442–3455.

[49] H. Zhao, P. C. Yuen, Incremental linear discriminant analysis for face recognition, IEEE Transactions on Systems, Man and Cybernetics 38 (1) (2007) 210–221.

[50] G.-F. Lu, J. Zou, Y. Wang, Incremental learning of complete linear discriminant analysis for face recognition, Knowledge-Based Systems 31 (2012) 19–27.

[51] D. Chu, L.-Z. Liao, M. K.-P. Ng, X. Wang, Incremental linear discriminant analysis: A fast algorithm and comparisons, IEEE Transactions on Neural Netw. Learning Syst 26 (11) (2015) 2716–2735.

[52] J. Liu, Z. Lian, Y. Wang, J. Xiao, Incremental kernel null space discriminant analysis for novelty detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4123–4131. doi:https://doi.org/10.1109/CVPR.2017.439.

[53] M. Karasuyama, I. Takeuchi, Multiple incremental decremental learning of support vector machines, IEEE Transactions on Neural Networks 21 (7) (2010) 1048–1059. doi:https://doi.org/10.1109/TNN.2010.2048039.

[54] J. Sirkunan, N. Shaikh-Husin, M. N. Marsono, Interleaved incremental/decremental support vector machine for embedded system, in: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1–5. doi:https://doi.org/10.1109/ISCAS.2019.8702745.

[55] R. Xiao, J. Wang, F. Zhang, An approach to incremental SVM learning algorithm, in: Proceedings of IEEE International Conference on Tools with Artificial Intelligence, 2000, pp. 268–273. doi:https://doi.org/10.1109/TAI.2000.889881.

[56] M. Artac, M. Jogan, A. Leonardis, Incremental PCA for on-line visual learning and recognition, in: International Conference on Pattern Recognition, (2002), pp. III: 781–784.

[57] I. Dagher, Incremental PCA-LDA algorithm, in: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2010, pp. 97–101.

[58] X. Nie, Y. Luo, H. Qiao, B. Zhang, Z.-P. Jiang, An incremental multi-view active learning algorithm for PolSAR data classification, in: International Conference on Pattern Recognition, 2018, pp. 2251–2255. doi:https://doi.org/10.1109/ICPR.2018.8545325.

[59] X. Nie, M. Fan, X. Huang, W. Yang, B. Zhang, X. Ma, Online semisupervised active classification for multiview polsar data, IEEE Transactions on Cybernetics (2020) 1–15doi:https://doi.org/10.1109/TCYB.2020.3026741.

[60] P. Zhou, Y.-D. Shen, L. Du, F. Ye, Incremental multi-view support vector machine, in: Proceedings of the SIAM International Conference on Data Mining, Alberta, Canada, SIAM, 2019, pp. 1–9. doi:https://doi.org/10.1137/1.9781611975673.1.

[61] P. Zhou, Y.-D. Shen, L. Du, F. Ye, X. Li, Incremental multi-view spectral clustering, Knowledge-Based Systems 174 (2019) 73–86. doi:https://doi.org/10.1016/j.knosys.2019.02.036.

[62] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, Eigenfaces vs. Fisherfaces: Recognition using class-specific linear projection, Vol. 19, 1997, pp. 711–720. doi:https://doi.org/10.1109/34.598228.

[63] M. Li, B. Yuan, 2D-LDA: A statistical linear discriminant analysis for image matrix, Pattern Recognition Letters 26 (5) (2005) 527–532. doi:https://doi.org/10.1016/j.patrec.2004.09.007.

[64] R. Zhi, Q. Ruan, Two-dimensional direct and weighted linear discriminant analysis for face recognition, Neurocomputing 71 (16) (2008) 3607–3611. doi:https://doi.org/10.1016/j.neucom.2008.04.047.

[65] Q. Wang, Z. Qin, F. Nie, Y. Yuan, Convolutional 2D LDA for nonlinear dimensionality reduction, in: Proceedings of the International Joint Conference on Artificial Intelligence, ijcai.org, 2017, pp. 2929–2935. doi:https://doi.org/10.24963/ijcai.2017/408.

[66] C.-N. Li, Y.-H. Shao, Z. Wang, N.-Y. Deng, Robust bilateral lp-norm two-dimensional linear discriminant analysis, Information Sciences 500 (2019) 274–297. doi:https://doi.org/10.1016/j.ins.2019.05.066.

[67] W. Yang, X. Yan, L. Zhang, C. Sun, Feature extraction based on fuzzy 2dlda, Neurocomputing 73 (10) (2010) 1556–1561. doi:https://doi.org/10.1016/j.neucom.2009.12.025.

[68] M. Wan, G. Yang, S. Gai, Z. Yang, Two-dimensional discriminant locality preserving projections (2DDLPP) and its application to feature extraction via fuzzy set, Multimedia Tools and Applications 76 (1) (2017) 355–371. doi:https://doi.org/10.1007/s11042-015-3057-8.

[69] X. Zhang, Y. Zhu, X. Chen, Fuzzy 2D linear discriminant analysis based on sub-image and random sampling for face recognition, International Journal of Pattern Recognition and Artificial Intelligence 34 (1) (2020) 1–19. doi:https://doi.org/10.1142/S0218001420560017.

[70] H. Li, L. Zhang, B. Huang, X. Zhou, Cost-sensitive dual-bidirectional linear discriminant analysis, Information Sciences 510 (2020) 283–303. doi:https://doi.org/10.1016/j.ins.2019.09.032.

[71] J. Yang, D. Zhang, A. F. Frangi, J.-Y. Yang, Two-dimensional PCA: A new approach to appearance-based face representation and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (1) (2004) 131–137. doi:https://doi.org/10.1109/TPAMI.2004.10004.

[72] H. Kong, L. Wang, E. K. Teoh, X. Li, J.-G. Wang, R. Venkateswarlu, Generalized 2D principal component analysis for face image representation and recognition, Neural Networks 18 (5-6) (2005) 585–594. doi:https://doi.org/10.1016/j.neunet.2005.06.041.

[73] Q. Wang, Q. Gao, Two-dimensional PCA with F-Norm minimization, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, USA, AAAI Press, 2017, pp. 2718–2724.

[74] Q. Gao, L. Ma, Y. Liu, X. Gao, F. Nie, Angle 2DPCA: A new formulation for 2DPCA, IEEE Transactions on Cybernetics 48 (5) (2018) 1672–1678. doi:https://doi.org/10.1109/TCYB.2017.2712740.

[75] C. Xu, D. Tao, C. Xu, Multi-view learning with incomplete views, IEEE Transactions on Image Processing 24 (12) (2015) 5812–5825. doi:https://doi.org/10.1109/TIP.2015.2490539.

[76] L. Zhang, Y. Zhao, Z. Zhu, D. Shen, S. Ji, Multi-view missing data completion, IEEE Transactions on Knowledge and Data Engineering 30 (7) (2018) 1296–1309. doi:https://doi.org/10.1109/TKDE.2018.2791607.

[77] C. L. Blake, C. J. Merz, UCI repository of machine learning databases, available : https://archive.ics.uci.edu/ml/datasets/Multiple%2BFeatures (1998).

[78] P. Perona, R. Fergus, F. F. Li, Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, in: Comput. Vis. Image Underst., (2007), pp. 59–70, available : http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[79] C. H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: CVPR, (2009), pp. 951–958.

[80] H. Kim, A. Evans, J. Blat, A. Hilton, Multimodal visual data registration for web-based visualization in media production, IEEE Transactions on Circuits and Systems for Video Technology 28 (4) (2018) 863–877. doi:https://doi.org/10.1109/TCSVT.2016.2642825.

[81] R. Fransens, C. Strecha, L. V. Gool, Parametric stereo for multi-pose face recognition and 3D-face modeling, in: Analysis and Modelling of Faces and Gestures, Springer Berlin Heidelberg, 2005, pp. 109–124. doi:https://doi.org/10.1007/11564386_10.

[82] The Database of Faces, AT&T laboratories cambridge, available : http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html (2002).

[83] J. W. Cooley, P. A. W. Lewis, P. D. Welsh, The fast Fourier transform and its application, IEEE Transactions on Education E-12 (1969) 27–34.

[84] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8 (6) (1986) 679–698. doi:https://doi.org/10.1109/TPAMI.1986.4767851.

[85] L. V. der Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (11).

[86] S. S. Shivagunde, A. Nadapana, V. V. Saradhi, Multi-view incremental discriminant analysis, Information Fusion 68 (2021) 149–160. doi:https://doi.org/10.1016/j.inffus.2020.10.021.

[87] S. S. Shivagunde, V. V. Saradhi, 2D multi-view discriminant analysis, Information Sciences 586 (2022) 391–407. doi:https://doi.org/10.1016/j.ins.2021.12.010.

[88] J. Lines, L. M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 289–297.

[89] S. Torkamani, V. Lohweg, Survey on time series motif discovery, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7 (2) (2017) e1199.

Publications

Journals

- **S. S. Shivagunde**, A. Nadapana, V. V. Saradhi, *Multi-view Incremental Discriminant Analysis*, Information Fusion, vol. 68, pp. 149–160, 2021.
- **S. S. Shivagunde**, V. V. Saradhi, *2D Multi-view Discriminant Analysis*, Information Sciences, vol. 586, pp. 391–407, 2022.
- **S. S. Shivagunde**, V. V. Saradhi, *2D Multi-view Incremental Decremental Discriminant Analysis*, IEEE Transactions on Cybernetics, 2022. (Under Review)
- **S. S. Shivagunde**, V. V. Saradhi, *View Incremental Decremental Multi-view Discriminant Analysis*, Applied Intelligence, 2022. (Under Review)

Vitae

Saroj Snehal Shivagunde

Current Affiliation PhD Student, Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati, India.

Email s.shivagunde@iitg.ac.in, sarojshivagunde@gmail.com

Web LinkedIn, GitHub

Research Interests I am interested in designing algorithms in the field of machine learning and pattern recognition to solve real life problems. Current topics of my research include **incremental learning** and **multi-view learning**.

Education

M.Tech, Computer Science and Engineering, 2014-2016
Indian Institute of Technology, Guwahati
Thesis Title : *Learning The Base Set of Kernels In Multiple Kernel Learning*
Supervisor : Dr. Vijaya Saradhi Vedula

B.E., Computer Science and Engineering, 2009-2013
Saraswati College of Engineering, Navi Mumbai
Thesis Title : *Image Encryption Using Logistic Map and Cheat Image*
Supervisor : Mrs. Arti Gore-Sabale