Understanding, Exploiting and Improving Inter-view Relationships

Sibi Venkatesan
June 30, 2020

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Artur Dubrawski, Chair
Jeff Schneider
Srinivasa Narasimhan
Junier Oliva, UNC, Chapel Hill
Abstract

Multi-view machine learning has received substantial attention in various applications over recent years. These applications typically involve learning on data obtained from multiple sources of information, for example, in multi-sensor systems such as self-driving cars and patient bed-side monitoring. Learning models for such applications can often benefit from leveraging not only the information from individual sources, but also the interactions and relationships between these sources.

In this proposal, we look at multi-view learning approaches which try to model these inter-view interactions explicitly. Here, we define interactions and relationships between views in terms of the information which is shared across these views, including corroboration and redundancy of information between views. We distinguish between global relationships, which are shared across all views, and local relationships, which are only shared between a subset of views. For example, in a multi-camera system, we can think of global relationships to be defined over the part of a scene which is visible to all cameras, while local relationships may exist between a subset of views to be defined by the intersection of the fields of view of only those cameras.

We consider three main aspects of modeling such inter-view relationships. First, we look at understanding relationships within multi-view data. We describe two methods which aim to uncover and model local relationships between views: (i) Robust Multi-view Auto-Encoder, which generalizes the idea of drop-out to views as a whole and (ii) One-vs-Rest Embedding Learning, which explicitly models the local relationships by considering each view separately. We also propose extensions to these methods, as well as alternate approaches to understanding inter-view relationships.

Next, we look at exploiting this understanding to solve down-stream tasks and real-world problems. Here, we use our proposed models to tackle real-world problems, and demonstrate the effectiveness of explicitly modeling inter-view relationships. We also discuss how we can extend our approaches to looking at special applications, such as dynamical systems and asynchronous multi-view data.

Finally, we discuss improving inter-view relationships by facilitating favorable interactions between views in multi-view data. We first show how we can re-interpret individual views as data points, allowing us to apply traditional machine learning approaches to modeling inter-view relationships. We then describe Scalable Active Search as a candidate approach for view-selection. We also propose additional methods to improve inter-view relationships using our view-as-data-point interpretation, and discuss ways for their online improvement.
## Contents

### I Overview

1 **Introduction** 
   1.1 Multi-view relationships ........................................... 3
   1.1.1 Understanding multi-view relationships .......................... 4
   1.1.2 Exploiting multi-view relationships .............................. 5
   1.1.3 Improving on multi-view relationships ............................ 5
   1.2 Application Domains: Special Cases ................................. 6
   1.2.1 Temporal and Dynamical Systems .................................. 6
   1.2.2 Asynchronous multi-view data ...................................... 6
   1.3 Proposal Outline ..................................................... 7
   1.3.1 Part II: Completed Work .......................................... 7
   1.3.2 Part III: Proposed Work .......................................... 8
   1.3.3 Part IV: Timeline .................................................. 9
   1.4 Notation ............................................................. 10
   1.5 Related Work ....................................................... 10

### II Completed Work

2 **Understanding Multi-view Relationships** ................................ 15
   2.1 Introduction ......................................................... 15
   2.2 Related Work ....................................................... 15
   2.3 Approaches .......................................................... 17
   2.3.1 Multi-view One-vs-Rest CCA .................................... 17
   2.3.2 Robust Multi-view AutoEncoder ................................ 19
   2.4 Experiments ........................................................ 20
   2.4.1 Synthetic Data .................................................... 20
   2.4.2 Results .......................................................... 21
   2.4.3 Studying performance under different levels of redundancy .... 22
   2.5 Conclusion ......................................................... 24
3 Exploiting Learned Multi-View Representations for Down-Stream Tasks 27
  3.1 Introduction .......................................................... 27
  3.2 Approaches .................................................................. 28
    3.2.1 Revisiting Multi-view Representation Learning .......... 28
  3.3 Experiments .............................................................. 29
    3.3.1 Datasets ............................................................... 29
    3.3.2 Evaluation ............................................................ 30
    3.3.3 Results ................................................................. 30
    3.3.4 Conclusion ........................................................... 31

4 Toward Improving Multi-view Relationships: Active Search 35
  4.1 Introduction .............................................................. 35
  4.2 Scalable Active Search for View Selection .......................... 36
    4.2.1 Related Work ........................................................ 37
    4.2.2 Problem Statement ............................................... 38
    4.2.3 Approach ............................................................. 38
    4.2.4 Experiments ......................................................... 41
  4.3 Conclusion ............................................................... 44

III Proposed Work 45

5 Revisit to Understanding Multi-view Relationships 47
  5.1 Introduction .............................................................. 47
  5.2 Generative Modeling Extensions to RMAE ....................... 47
    5.2.1 Flow-based Approaches ....................................... 48
    5.2.2 Variational Approaches ....................................... 49
  5.3 Relationship Graphs ................................................... 49
    5.3.1 OVREL: Multi-fidelity Relationship Graphs ............... 50
    5.3.2 Interpreting Relationship Graphs .............................. 51
  5.4 Metrics for Qualitative Characterization of Views ............. 52
  5.5 Conclusion ............................................................... 52

6 Revisit to Exploiting Multi-view Relationships 53
  6.1 Introduction .............................................................. 53
  6.2 Temporal Data ........................................................... 54
    6.2.1 Related Work ......................................................... 54
    6.2.2 Representation Learning ........................................ 55
    6.2.3 Multi-view Tasks with Temporal Data ....................... 56
  6.3 Asynchronous Data ..................................................... 56
    6.3.1 Adversarial Training .............................................. 57
    6.3.2 Asynchronous Representation Learning ..................... 57
CONTENTS

7 Improving Multi-view Relationships
  7.1 Introduction .......................................................... 59
  7.2 View Duality ......................................................... 59
    7.2.1 Modeling over Sets and Distributions ....................... 60
    7.2.2 Learnable Relationships ..................................... 60
  7.3 Online Improvement ............................................. 61
  7.4 Conclusion ......................................................... 61

IV Timeline ............................................................ 63

8 Timeline ............................................................. 65
Part I
Overview
Chapter 1

Introduction

Multi-view machine learning, or multi-modal machine learning, involves learning on data which has multiple, potentially asynchronous, observation models. For example, videos have both an audio and visual channel which provide complementary information. Images and their captions provide different views of the same data.

The applications of multi-view machine learning are numerous. One simple setting would be to use the multiple views of a dataset to build a more robust classifier than one that can be trained on any single view alone. This is the basis for the co-training learning paradigm. Co-training is a semi-supervised learning framework which uses two or more complementary views to jointly train classifiers over each view. It does so by first building view-specific classifiers and iteratively expands the labeled dataset by adding the unlabeled points based on the confidence of these individual classifiers.

Other common applications are cross-modal translation/retrieval. For example, given an image, we would like to generate an appropriate text caption. On the other hand, given a text description, we could also retrieve the most appropriate image from a given dataset. Language translation, and cross-modal sequence-to-sequence translation in general, is another domain for the application of multi-modal machine learning.

1.1 Multi-view relationships

In this proposal, we are interested in exploring multi-view machine learning from the perspective of understanding and exploiting the relationships between the multiple views. The key idea is that multi-view data doesn’t just provide us with multiple sets of “features” for the data, it also provides structural information in the form of the interactions and redundancies between views. These relationships can be exploited in multiple ways:

1. Leveraging redundancies to bolster models built over the data.
2. Quantifying "usefulness" of a given view through the relationship.

3. Reconstructing missing or corrupted views.

To this end, we propose methods for Robust Representation Learning over multi-view data. Representation learning is, in general, an intermediate step to extract information from data in a meaningful form which is more conducive to learning over downstream tasks. By learning a representation which respects the interactions and redundancies between views, we can tackle the above problems.

We classify the methods described in this proposal along three broad directions:

1. Studying and understanding multi-view relationships.
2. Exploiting these relationships for learning problems.
3. Improving and expanding upon the relationships themselves.

We would also like to distinguish between what we call "global" and "local" relationships. Global relationships are those which are shared across all views. An example of an approach which tries to learn such relationships would be the classic formulation of a multi-view CCA problem [103]; here, they learn for a single projection for each view into common embedding space which maximizes some measure of a joint correlation.

Local relationships would be those which manifest only within a subset of views. An example of a setup with such relationships could be a multi-camera system where the common visible regions depends on the subset of cameras chosen. While this example lends itself to geometric and vision based methods of learning local relationships, eg. calibration, the more general problem poses more challenges. The nature and structure of these relationships (eg. geometric) isn’t always known within the data. We are more interested in tackling this problem in the general setting: learning local relationships over more general datasets. Of course, leveraging domain information when available would useful as a scaffold for learning such relationships.

Below, we discuss the overall ideas behind our different proposed directions:

1.1.1 Understanding multi-view relationships

This direction is closely related to the one below. The idea of "understanding" relationships between views can be considered within the context of an overarching learning problem. In this case, we can recast the problem of understanding into one of modeling the utility of these relationships to the given learning problem.

We could also treat views as proxies for these relationships in this case. It is easier to define some measure of "utility" over views, since they already have a clearly defined structure (the separation into views itself). Then we can reason
1.1. MULTI-VIEW RELATIONSHIPS

about utility of a given view based on the difference between performance, or some other metric, with and without the view. If the information which is captured by the view and its local relationships has high utility, then removing this should cause a significant change in our performance metric. Of course, such a measure of utility depends both on the overall learning problem and the multi-view model chosen.

With the absence of a specific learning problem, we consider the problem of missing-view reconstruction. Given multi-view data, we can always create such a problem, either by using pre-existing gaps in the data or by artificially removing a subset of views from the dataset.

*Goal:* To understand and evaluate the utility of relationships between views, and views themselves, within the context of a learning problem.

1.1.2 Exploiting multi-view relationships

Here, we look at the above problems from the opposite direction. Given a learning problem and our notion of utility, can we optimize our view-relationship model to maximize performance? Essentially, we exploit what we have learned from these relationships and redundancies to help solve a given learning problem.

We also look at specific cases where the multi-view data has additional properties which we need to consider, such as temporal structure within the views.

*Goal:* To apply our understanding of multi-view relationships to relevant learning problems over the multi-view data.

1.1.3 Improving on multi-view relationships

If we can model the relationships appropriately, we can even look a step further: we can reverse engineer views which optimize the utility of these relationships.

For example, consider the problem of sensor placement. If we are able to model (i) the relationship between views, or sensors in this case (ii) utility of these relationships, possibly based on the performance of a downstream task (ii), we can find a good placement of sensors to maximize this utility. In a similar vein, we can also consider the task of sensor/view "hallucination," i.e. simulate the data from a sensor if it were placed at a certain point. This is essentially view reconstruction as discussed earlier, but with flexibility in the stipulation of a view itself.

This is more in line with the philosophy of error correction: the modeling and designing of inter-view structure itself to optimize for potential downstream missing-data reconstruction.

*Goal:* To manipulate and/or design the views themselves, and their relationships, to optimize some metric of performance or utility.
1.2 Application Domains: Special Cases

We also take a look at multi-view data with (or without) certain types of structure within and between views.

1.2.1 Temporal and Dynamical Systems

We previously mentioned incorporating domain-specific information to inform our representation learning models; in that case, we were looking at using the geometric structure in a multi-camera system. In the case of temporal data, we have additional sequential inter- and intra-view structure which we can exploit.

Sequential data also opens up interesting downstream learning problems such as forecasting and, in the case of controllable systems, policy learning. For example, medical and predictive maintenance data are important applications which often have underlying temporal structure. In the medical domain, multi-modal sequential data is abundant, from low-level continuous signals like patient vital signs (heart rate, blood pressure, etc.) to high-level level discrete information like a patient’s hospital visits, treatments, etc. Further, there are also questions we could ask given this data in terms of patient diagnosis, prognosis (essentially, forecasting) or potential treatment options.

Predictive maintenance mirrors the medical field, where the entities concerned instead are pieces of equipment like mechanical systems and machines. Similar to medical data, we can collect analogous temporal information about these systems. For example, consider a car, which in its own right is a complicated mechanical system. We might have data collected from the car, like engine temperature, oil pressure, fuel efficiency, GPS location, etc. over a period of time. Of course, with the recent inception of self-driving vehicles, we could have access to much more information like sensor measurements of the surroundings, control actions (eg. movement of the steering wheel, gear shifts, etc.) and such. Here, we would like to answer questions in terms of predicting remaining useful life (RUL) or forecasting the system’s activity for anomaly detection.

1.2.2 Asynchronous multi-view data

We are also interested in the more extreme case of missing-view data where we have very little corresponding data between different views. In the worst case, every data point is only associated with a single view. This problem has been discussed within the context of multi-view clustering [53]. Eg. different hospitals have different sensors and any given patient only goes to one of the hospitals. Here, a patient is a data point and their vital signs/sensor readings is a view, based on the hospital they go to.
Considering this more general and more complicated scenario, we can try to answer some questions. Is it still possible to build the same kinds of models, without explicit relationship structure between views? If not, what is the best we can hope to achieve? This question could be tackled from two directions. Given a particular learning problem: (i) identifying, and possibly proving, the theoretical limitations of a model in this scenario as compared to the synchronized setting. (ii) identifying additional assumptions (other than synchronization) which would be needed to learn a model. To this end, we intend to explore a few approaches within the asynchronous setting, and attempt to theoretically analyze their limitations.

1.3 Proposal Outline

We propose learning over multi-view systems from the perspective of exploring the relationships between views. The major ideas in this proposal can be split into the three directions described before:

1. Understanding the relationships between the multiple views
2. Exploiting these relationships
3. Improving these relationships given our understanding

Central to these directions is the concept of Multi-view Representation Learning. We want to extract meaningful information from both the views themselves as well as the inter-view relationships, and representation learning is a means to this end. Further, understand and exploiting inter-view relationships are closely related, since our analysis of these relationships is often tied to the learning problem.

This proposal will describe our completed work under items (1) and (2) as well as our proposed work for all three.

The remainder of this document will be structured as follows:

1.3.1 Part II: Completed Work

This part describes our progress in Multi-view Representation Learning, primarily toward understanding and exploiting the underlying structure captured by multi-view relationships.

Chapter 2: Understanding Inter-view Relationships Through Representation Learning

In this chapter, we consider the problem of synchronized multi-view representation learning. We would like to learn a representation for multi-view data that can capture and respect local inter-view relationships. As mentioned before, we will use
the missing-view reconstruction problem as a a scaffold on which we build our understanding.

Here, we describe a couple of algorithmic ideas to approach this problem:

1. **One-vs-Rest Embedding Learning**: Explicitly consider local relationships by building an embedding for each view individually, in terms of its dependencies on other views.

2. **Robust Multi-View Auto-Encoder**: Extend the idea of drop-out to the view-level to encourage learning a robust representation, resilient to views going missing.

For these approaches, we evaluate on synthetic datasets to demonstrate their efficacy. We also look at the concept of “relationship graph” as defined by these methods, as an attempt to understand inter-view interactions.

**Chapter 3: Exploiting Learned Multi-View Representations for Down-Stream Tasks**

Representation Learning acts as an intermediate step to consolidate the relevant information from multi-view which can then be used to tackle down-stream applications.

In this chapter, we look at the applications of medical patient monitoring and predictive maintenance and evaluate the learned representations based on their performance on said tasks. We also discuss joint-optimization approaches where we simultaneously learn a representation and optimize performance on the down-stream task.

**1.3.2 Part III: Proposed Work**

Here, we move to the work we propose to do. These include extensions of the completed work, their applications to temporal and dynamical data as well additional methods for different contexts.

**Chapter 4: Toward Improving Multi-view Relationships: Active Search**

Here, we look at improving multi-view relationships from the perspective of system design. We would want to choose views which maximize some metric of performance or efficiency, such as redundancy or reconstruction.

If we are able to model our multi-view relationships appropriately, we can then consider actively choosing the best view for optimizing the relevant metric.

To this end, we describe an approach for Scalable Active Search [122], which allows us to interactively search for and select points under some similarity metric. We discuss reinterpreting the approach from the perspective system design for view selection.
Chapter 5: Revisit to Understanding Multi-view Relationships

We revisit the methods and ideas in Chapter 2 and discuss some of their extensions and alternatives. We try to formalize the idea of a relationship graph and look at ways in which we can interpret and analyze it to understand multi-view relationships.

We also discuss some measures of utility for views, given the context of a problem. Here, we want to provide metrics which can be used to define "usefulness" of a view as an explicit quantity, and use these for modeling decisions for a given problem.

Chapter 6: Revisit to Exploiting Multi-view Relationships

In this chapter, we look at specific cases in application domains where our multi-view data has certain properties. We look at what the additional (or lack of) structure would entail, and how we can appropriately deal with them through our models:

- **Temporal and Dynamical Systems**: The additional structure of time allows us to augment our existing approaches to learn better representation models. We look at some of these augmentations, as well as discuss new learning problems which come to light within the context of temporal data.

- **Asynchronous Data**: Here, we no longer assume that we have aligned views with matching samples for any given data point, or at least not enough to learn reliable models. We discuss how to tackle such a case and explore existing approaches within this framework, and their application to multi-view data.

Chapter 7: Improving Multi-view Relationships

In this chapter, we take a brief look at additional methods we can use to enhance and optimize multi-view relationships from the system-design standpoint. We describe Active Search as a candidate for view selection in Chapter 4; here we look at other viable graph based and metric learning approaches which would be appropriate for improving multi-view relationships.

1.3.3 Part IV: Timeline

The tentative time-line for completing the proposed work is about a year, to Fall 2021. This timeline, and the checkpoints along the way, will be described in the last chapter.
1.4 Notation

Here is the notation we will be adopting through this document. Each view of the multi-view data will be represented as $X_i$ where $i$ is the index of the view, and $K$ is the number of views. The samples within each view will be indexed by a superscript: the $j^{th}$ sample of view $i$ is given by $x^j_i$. In general, we assume correspondences for a given sample index across different views. For example, $x^j_1$ and $x^j_2$ both correspond to two views of the same underlying data-point. We take total number of underlying data points to be $N$.

In the case of missing views, we take $K_\alpha$ to be the set of view indices which are available; this depends on the context but is usually associated with a single sample. If the $j^{th}$ sample of view $i$ is missing, we do not have a value for $x^j_i$, and $i \notin K_\alpha$. In the case of explicit latent representations, we take $L_i$ to be the latent representation of view $i$ and $L_{all}$ as the shared latent representation across all views.

For special cases, such as time-series data or asynchronous views, we will describe appropriate notation changes if needed.

1.5 Related Work

Multi-modal machine learning has seen an impetus of recent work, owing to the increasingly easy access to a vast amount of multi-view data. For instance, video data, language-to-language translation, and even on-board sensors on self-driving cars give multiple modalities of viewing the same dataset. It is then imperative to interpret and analyze these multiple channels holistically for models to make the best use of the increased amount of information.

Following the categorizations by [6], we describe the general directions of multi-modal machine learning research in recent years.

**Representation Learning**  Representation learning involves extracting meaningful information from multi-view data, typical functioning as an intermediate step before other downstream tasks. The aim is to recast the data into a space which is more conducive to learning. The usual challenges with multi-view representation learning are the possible heterogeneous sources of data, different noise processes across views or missing data.

A good representation is often the decisive factor for performance, even given a specific learning model. This is evidenced by recent improvements in the fields of speech recognition [43] and visual object classification [64]. [9] discuss some essential properties which are favorable in a "good" representation, including smoothness, spatial and temporal coherence, natural clustering, and sparsity among others. [114] add to these properties by including others such as reflecting similarity.
in concept space and ease of obtaining even with missing or corrupted modalities.

The overall approaches can be broadly split into joint representations where the multiple modalities are usually present in both train and test time, and coordinated representations where separate but constrained/coordinated representations are learned for multiple views. The latter is useful in the case of missing view data, but usually is limited to two modalities only.

A more detailed exposition on the related literature is given in Chapter 2.

Translation Multi-modal translation is the conversion of data from one modality to another. Applications include speech synthesis [48], visual speech generation [79], video description [63], cross-modal retrieval [100], etc.

The approaches can be broadly put into two categories: Example based where methods usually learn dictionaries which facilitate the cross-modal mapping and generative where the model produces a translation.

Example based approaches are often simpler; retrieval based methods try to find the closest sample in a learned dictionary to translate an input. Some of these approaches include visual speech synthesis [12], CNN-based image representation with adaptive neighborhood selection [137], K-nn retrieval with consensus caption selection [22]. While these approaches typically have the advantage of needing to represent only one modality through which retrieval is done, they often need to be augmented with re-ranking of retrieved translations [78], [91], and the lack of representation power in a unimodal space. Other example based approaches include combination-based models where, instead of something like a dictionary look-up, they combine dictionary samples in meaningful ways to improve translations. [40] use k-NN image retrieval followed by phrases from their captions to generate target caption; [68] follow a similar methodology, with the use of Linear Programming to combine retrieved phrases. Most of these approaches have hand-crafted or heuristic-based rules.

Generative approaches are typically more challenging since both source and target modalities need to be modeled and understood. They are also harder to evaluate. They generally fall under (i) Grammar-based models - human behavior description [63], image description using and-or graph-based models [138], etc. (ii) Encoder-Decoder models - machine translation [55], image captioning [77], [124], video descriptions [102], [123], etc., (iii) Continuous-generation models, usually for live translation - visual speech genesis [119], text-to-speech [143], etc.

Alignment Multi-modal alignment deals with finding relationships or correspondences between subcomponents of samples from two or more modalities. For example, aligning movie script with the book chapters it was based on [149], finding areas in the image referred to by words in the caption [57].

There are three broad categories here. Explicit alignment, which can be unsupervised such as Dynamic Time Warping if similarity metrics exist [10], [65], [118]
CHAPTER 1. INTRODUCTION

or generative model-based (e.g., aligning visual objects to spoken words [141], alignment between movie scenes to screenplay [20], sentence-to-video translation [85]). It can also be supervised, for example, using techniques like Canonical Correlation Analysis [94] for finding shared coordinate spaces, or deep learning methods like CNN-based similarity measures between scenes and book-text [149].

Implicit alignment, which is used as an intermediate (often latent) step for another task for improving performance. For example, generative models such as aligning words between languages for machine translation [125], phoneme-to-transcription [108] or neural-network based models which use attention mechanisms (e.g., image areas [135], words in sentence [5], segments of audio [15], etc.). There are also some alternatives to attention-based neural-network based approaches, such as dot-product similarity between images for cross modal retrieval [57], [58].

Fusion Multi-modal fusion involves the integration of multi-modal data with the aim of predicting an outcome measure, say classification or regression. Work has been done under multi-modal fusion for over 25 years, and the field has progressed and matured since. The main benefits of fusion are (i) robust predictions, (ii) ability to capture complementary info, (iii), operate when one modality is missing. In fact, recent work has blurred the line between multi-modal fusion and representation learning, owing to the advent of deep models interlacing representation learning with classification or regression.

Fusion sees a broad range of applications including Audio-Visual Speech Recognition (AVSR) [96], multi-modal emotion recognition [111], medical image analysis [51], multimedia event detection [70].

These approaches are broadly categorized under model-agnostic and model-based. Under these categories, they can also be divided into "early", "late" and "hybrid" fusion techniques depending on where in the pipeline the modality-fusion takes place. The vast majority of approaches are model-agnostic, and include early feature-based integration, late individual-modality decision based using aggregation schemes [105], [82], [96], and hybrid multi-modal speaker identification [133] and event detection [70].

Model-based approaches include multiple-kernel learning based (MKL) approaches [13], [34], [95], [140], graphical model-based approaches (generative [86], [35] or discriminative [28], [97], [7], and neural network based [30], [87], [130], [89], [52].
Part II
Completed Work
Chapter 2

Understanding Multi-view Relationships

2.1 Introduction

In this chapter, we consider the problem of representation learning for multi-view data. Our goal is to extract a useful and robust latent representation by leveraging the relationships between the different views. Unlike typical existing approaches for multi-view representation learning, we try to model "local" relationships between subsets of views. Our models can then leverage these local relationships to provide a robust latent representation.

We propose two methods to approach this problem. The first is an extension of CCA where we consider multiple one-vs-rest CCA problems, one for each view. To encourage searching for local relationships, we add group-sparse regularization where the groups correspond to the views without additional information.

The second method is a more straightforward extension of a multi-view AutoEncoder. We encourage learning a robust representation by dropping random subsets of the views while still reconstruction all views during training. This is essentially dropout as applied to views as opposed to layer units.

We describe simple synthetic experiments showing the usefulness of our approaches in exploiting local redundancy in the data.

2.2 Related Work

Learning over multiple modalities is often difficult, due to heterogeneous sources of data, different levels of noise or missing data in some views. This makes it imperative to extract meaningful information from the different views in a robust fashion. Representation learning is thus one of the core directions of multi-modal machine learning research. It is common as an intermediate step before learning
CHAPTER 2. UNDERSTANDING MULTI-VIEW RELATIONSHIPS

over a down-stream task.

Many such learning methods are tailored to certain domains, wherein they exploit the structure available specific to the data. For example, Audio-Video Speech Recognition (AVSR) has been the subject of research for many years now. Traditionally, deep neural networks are used to handle visual, textual and acoustic data [88], [92], [126] where the model projects the modalities into a joint space [3], [83], [92], [134]. This representation is then used for the relevant learning task. It is common to pretrain such networks using AutoEncoders on unsupervised data [44].

Multi-view AutoEncoders are also extended to learn latent representations over multi-modal data. [88] learn modality specific AEs and then fuse together the latent states into a final shared representation. [106] use auto-encoders for semantic concept grounding, with the addition of a loss-term for object-label prediction. [126] fine-tunes the representation learned by the generic AEs on a given task.

Sequential data often needs additional care while learning latent representations; the data is not always fixed-length (sentences, video, etc.). RNN/LSTM based encoder-decoder networks are often used for this where the hidden state at the end of a variable-length sequence is used as its fixed-length representation [5], [123]. RNNs are often used for ASVR [19], audio-visual affect recognition [16], [89], etc.

Neural networks have their advantages; given domain-specific architectures and the potential for pre-training, they often show superior performance on certain tasks. But they need a lot of data and are not always able to gracefully handle missing data from modalities.

Another popular multi-modal representation learning approach is based on graphical models. Unsupervised methods such as Deep Boltzman Machines (DBN) have been extended to multi-modal datasets. [113] introduce multi-modal Deep Belief Networks for representation learning. Multi-modal DBNs are also used for audio-visual emotion recognition [59], AVSR [47], gesture recognition [131], and other applications [92], [115].

Graphical model based approaches deal with missing data well, and can often be trained in an unsupervised manner. But they tend to be difficult to train with high computational costs, often needing approximate variational training methods [114].

Another class of models are coordinated-representation models, where the modalities are not projected to a joint space but instead are coordinated through similarity constraints. Some of these approaches are based on similarity models [128], [129], [29], [61], [110], [93], [136]. Others are based on correlated embedding spaces such as CCA-like techniques. CCA models are common for cross-modal retrieval problems [41], [62], [100] and AV signal analysis [104], [109]. Nonlinear and deep extensions of CCA such as Kernel CCA [69], Deep CCA [2], Correspondence AEs [27], etc. have also been proposed.

Coordinated-representation models are usually limited to two-modality problems, though there have been some extensions into three or more views. Multi-
view CCA [103] learns a projection for each view to maximize some measure of global correlation, but these don’t share the simple eigen/singular-value decomposition based solutions which 2-view CCA problems often have.

2.3 Approaches

2.3.1 Multi-view One-vs-Rest CCA

Usually, existing multi-view CCA extensions [54], [103] involve finding a single set of projections for each view which maximize some overall correlation objective across all views. For example, [103] formulate their objective as follows:

$$\max_{\mathbf{w}_i} \sum_{i<j} (X_i w_i)^T (X_j w_j)$$

The projections here are not explicitly optimized to capture local redundancy between views. They try to extract something akin to a “global embedding” for each view, projections into a single shared space across all views. While this is useful to extract globally shared structure, nuanced local relationships are ignored. To naively extend this to model all possible local relationships, we would have to do an exponential number of multi-view CCA computations on all subsets.

Our proposed method attacks this problem by solving individual problems, one for each view. The goal is to understand the “directed” relationship between a given view and the remaining views. The term "directed" here refers to how we view the "contribution" of another view to the given view.

We describe the approach we propose in [121]. We reduce the problem of global latent representation learning to $K$ one-vs-rest CCA problems, one for each view. Given a single view, we learn the combined projections for each remaining view to maximize the correlation with the given view. Here, we are essentially performing a 2-view CCA computation where the remaining views together form the second "view". This way, we hope to uncover a dependency structure for correlation/reconstruction where we can model the "contribution" of any other view to the given view. Of course, a simple concatenation of views can disturb view-specific structure, and ignores the information carried by the fact that there are multiple views [73].

To remedy this, we make use of group-sparse regularization to encourage learning projections which respect this structure. The groups here correspond to the individual views within the aggregation of the remaining views. With this, we can define our objective. For now, we assume that all our projections are linear; we discuss how to move away from this a little later. For each view, we minimize the following:

$$\min_{P_{ij}} f(X_i P_{ii}, \sum_{j \neq i} X_j P_{ji}) + \lambda \sum_{j \neq i} R_G(P_{ij}) + \gamma R_{all}(P_{i:})$$

(2.2)
where \( f(A, B) = -A^T B \), \( R_G \) is the group-sparsity regularizer, \( R_{all} \) is a global regularizer and \( P_i \) refers to the concatenation of all projections \( P_{ij} \) where \( j \neq i \). We typically take the group-sparse regularizer to be the \( L_\infty \) norm and the global regularizer to be the \( L_1 \) norm. For CCA, we add the associated orthogonality constraints as well:

\[
(X_i P_{ii})^T (X_i P_{ii}) = \left( \sum_{j \neq i} X_j P_{ji} \right)^T \left( \sum_{j \neq i} X_j P_{ji} \right) = I
\]

Since this is basically a 2-view CCA problem for each view, we optimize this using linearized ADMM following [116].

While this approach has been described from the context of CCA, the primary points of note are the (i) one-vs-rest reduction to uncover local relationships and the (ii) group sparsity regularization to respect the view-structure of the data. With this in mind, the function \( f \) in Equation 2.2 (and relevant constraints) can be replaced by other appropriate loss functions which optimize similar criteria. A simple replacement is the squared \( L_2 \) reconstruction error: \( f(A, B) = ||A - B||_2^2 \). Without orthogonality constraints, this reduces to a straightforward convex optimization problem. To guard against the trivial solution here, we can fix \( P_{ii} \) to be the identity. If we stack the problems back together, we can rewrite the problem as a linear system solved with least squares:

\[
\begin{bmatrix}
X_1 & \ldots & X_K
\end{bmatrix}
\begin{bmatrix}
-I_{d_1} & \ldots & P_{1K} \\
\vdots & \ddots & \vdots \\
P_{K1} & \ldots & -I_{d_K}
\end{bmatrix}
\begin{bmatrix}
X_1 \\
\vdots \\
X_K
\end{bmatrix}
\]

\[= 0\]

Let us define \( M \) to be the combined projection matrix in this equation. This formulation shows that we are essentially trying to construct a matrix \( M \) which has the data distribution as its null space. We call this matrix \( M \) the redundancy matrix. Group-sparse regularization here translates to block-sparse regularization of \( M \). We expect that, by optimizing this loss, the sparsity structure of \( M \) reveals information about the directed relationship between different views.

So far, we have assumed linear projections \( P_{ij} \) but we can also pick classes of projection functions \( P_{ij}(X_{ij}) \) which we can optimize over. For example, deep neural networks are straightforward to incorporate into this formulation.

**Global latent Representation** After computing these individual projections with their respective group-sparsity structures, we have uncovered directed relationships between the views. The group-sparsity regularization encourages using only the most important remaining views to learn the projections from. While this does not uncover all possible local relationships, it does try to recover the more apparent and prominent ones.
2.3. APPROACHES

With these directed relationships, we can construct a "relationship" graph. We can also build a multi-fidelity version of this by tweaking the group-sparse penalty to tune the number of represented other-views for any given views. The latent representation here is implicit in the learned projections and the relationship graph. In the case of linear projections, we can extract a global embedding by looking at low-rank decompositions of the matrix $M$; this would give a basis for the relationships themselves, and a latent space can be recovered through these.

2.3.2 Robust Multi-view AutoEncoder

The typical strategies for training a Multi-view AutoEncoder (MVAE) have the same concerns outlined in the previous section; they try to directly learn a shared embedding space which best reconstructs all views ( [139], [126]). This often means learning a single bottle neck representation shared across all views. Such an architecture implicitly captures the intersection of information across all views. We train an MVAE which learns to be robust to missing views by trying to capture the union of information across the views instead.

In our proposed architecture [121], called the Robust MVAE, as shown in 2.1, we have two levels of encoding. The first is at the view level, where every view has its own individual encoder network. These encoders produce the latent embeddings $L_i$ for their respective views. Then, we compose this with an additional encoder, called the meta-encoder, which operates on top of these embeddings to produce the final global latent representation $L_S$. This is then used as input to the decoders for the reconstruction of different views.

Here, our idea for "robustness" of a representation is the ability of faithful reconstruction of all views given that an arbitrary subset of views are missing at input. For this, we borrow from the idea of dropout; every batch, we drop a different, random subset of views while forcing the reconstruction of all views. In this way, the training encourages the latent representation to exploit redundancy of information across different views.

This is similar to the Variational Auto-Encoder with Arbitrary Conditioning (VAEAC) [49], which is a generative model for estimating arbitrary missing feature values in data (eg. in-painting). While theirs is a single-view approach, similar to RMAE, they also consider sampling "dropped" features from some prior distribution. However, our approach allows us to learn view-specific encoders, since we can exploit the view-structure in our data. In our proposed work, we will look at generative modeling of multi-view data in a similar fashion to the VAE-AC, as well as using flow-based models.

To emulate dropout more appropriately, we perform a relative scaling of the input to the encoders based on number available views. In dropout with probability $p$, the output of the used units are scaled by $\frac{1}{p}$ to compensate for the missing units. Similarly, we scale the available views by $\frac{K_v}{K_a}$ where $K_a$ is the number of available
Figure 2.1: This is the outline of the Robust Multi-view AutoEncoder for 5 views. Each bottom arrow represents the encoder for its respective view, and each upper arrow represents the decoder. $L_i$ is the encoding for view $i$ and $L_S$ is the global latent representation. The bottom arrows are dotted to represent potential drop-out during training time.

views during that iteration. However, unlike in unit-level dropout, we also do this during test time when we have missing views.

We can also change where the view-dropout takes place; we can either zero out the input $X_i$ or we can zero out the latent encoding $L_1$. The former method is similar to encouraging every individual view encoder to output an informative "mean" embedding which works well in lieu of missing data. The latter localizes the "robustness" of the encoding to the meta-encoder level.

While the Robust MVAE is relatively straightforward, it learns a latent representation jointly over all views. Our previous approach initially considers each view individually and then aggregates the learned projections.

2.4 Experiments

2.4.1 Synthetic Data

Here, we describe a simple approach for the construction of a multi-view dataset with redundant inter-view relationships. Figure 2.2 shows a Venn diagram for the case with three views $X_1, X_2$ and $X_3$. 
2.4. EXPERIMENTS

We can treat each individual partition as having an independent data distribution which can contribute to the overall latent space of the data. Any partition which lies within the circle of a given view is accessible by it; thus, view $X_1$’s latent space contains the contributions of the four partitions contained within its circle. For example, the three views could be cameras with the circles corresponding to their fields of view, and the individual partitions are regions which are accessible to a specific subset of the cameras.

To construct a redundant multi-view dataset, each of these partitions can be turned "on" or "off" to allow or restrict their contribution to the latent space. For example, if we only turn on $X_1 \cap X_2$, $X_2 \cap X_3$ and $X_3 \cap X_1$, then, this would give us a dataset where any two views are enough to reconstruct the entire latent space. This allows us to customize the redundancy-relationships offered by the different views, and study our methods under different levels/forms of this redundancy.

2.4.2 Results

Multi-view One-vs-Rest Embedding Learning

4-view problem with simple redundancy Here, we consider the simple case where any two views are enough to reconstruct all other. We extend the three view description in the previous description as follows: We consider four underlying independent feature partitions $A, B, C, D$. The four views were created as $X_1 = [BCD]$, $X_2 = [ACD]$, $X_3 = [ABD]$ and $X_4 = [ABC]$, with some noise added.

For the Multi-view One-vs-Rest algorithm, we used the reconstruction loss, without the CCA constraints. Figures 2.3 shows the redundancy matrix $M$ learned.
with using group-sparse regularization and 2.4 shows $M$ with only least-squares. The group-sparsity encourages removing views if their contribution does not add much to the information provided by the rest of the views.

**Robust Mulvi-view AutoEncoder**

For the Robust MVAE, we ran a similar experiment but with the goal of understanding how reconstruction error of all views varies with number of available input views. Our synthetic data was similar to that used in the previous experiment, but we looked at multiple experiments with varying number of input views.

Here is a plot of a curve for average error of reconstruction given number of available views for a 5-view system. Figures 2.7, 2.9 shows the trend we would expect; the average reconstruction error of all output views goes down the more input we have. This plot does not display a more detailed analysis of reconstruction where we can fix the input subset to be helpful/unhelpful for a particular output view. In general, the trend we notice is that the error of reconstruction of an output view decreases with more input views which are locally redundant with it. Again, this is something which we would expect.

![Redundancy Matrix:](image)

Figure 2.3: Redundancy matrix after optimizing using group-sparse regularization. The learned projections largely ignore some redundant views in the reconstruction.

### 2.4.3 Studying performance under different levels of redundancy

Next, we consider the problem of reconstruction using the learned robust latent representation of the RMAE. The synthetic dataset family we look at are those
2.4. EXPERIMENTS

Figure 2.4: Redundancy matrix after optimizing using unregularized least squares. The learned projections do not exploit the redundancies.

where every numbered view intersects in information with only the ones next to it (indexed 1 less and 1 more in this case). The algorithms we compare here are the RMAE, the Concatenated unimodal AE (ConcatMAE) and the Intersection MAE (IntersectMAE). We show training and test-set errors for average reconstruction of all views as a function of available input views.

We also try to show an empirical estimation of "usefulness" of a single view under each algorithm by plotting the one-to-one reconstruction error for each view to each other view. We would hope that the methods are able to recognize which views have local intersections of information and are able to reconstruct those views better.

The plots we show here are for number of views = 4, 5, 6. "Relative error" in figures 2.5, 2.7 and 2.9 refers to error relative to the largest single-view reconstruction error across all methods.

The overall performance of the different methods is as expected. Since IntersectMAE directly learns a bottleneck representation which is common to all views, the training implicitly tries to learn the intersection of information across the different views, as opposed to the union. This is evident from the training vs. test error curves as well as the single-view error matrices.
2.5 Conclusion

We have described two approaches for robust multi-view representation learning: (i) Multi-view One-vs-Rest CCA with group sparse penalization and (ii) Ro-
2.5. CONCLUSION

Figure 2.7: [5-view problem] Train and test reconstruction error vs. number of views for different AE competitors. This plot also shows the 1-sigma confidence interval for different choices of available views.

Figure 2.8: [5-view problem] One-to-one single-view reconstruction for different AE competitors.

bust Multi-view AutoEncoder with randomized view-dropout. Our objective with these approaches is to uncover local relationships/redundancies between subsets of views, to help better understand the underlying structure of the data. Our results on simple synthetic data demonstrate that the approaches are a viable means to achieving this.
CHAPTER 2. UNDERSTANDING MULTI-VIEW RELATIONSHIPS

Figure 2.9: [6-view problem] Train and test reconstruction error vs. number of views for different AE competitors. This plot also shows the 1-sigma confidence interval for different choices of available views.

Figure 2.10: [6-view problem] One-to-one single-view reconstruction for different AE competitors.
Chapter 3

Exploiting Learned Multi-View Representations for Down-Stream Tasks

3.1 Introduction

In Chapter 2, we discussed methods toward robust representation learning of multi-view data. Here, we discuss the application of these methods to real-world problems to demonstrate their effectiveness. As mentioned before, the primary application domains we will focus on are medical patient-health monitoring and predictive maintenance.

We will evaluate our representation learning methods over these datasets, as well as the viability of the learned representations in terms of performance down-stream tasks. The approaches we consider are the same: (i) Robust Multi-view Auto-Encoder as well as (ii) One-vs-Rest Multi-view Embedding Learning. The second approach described does not give an explicit representation, but rather a model of the relationship structure between the multiple-views. This approach is primarily suited to the reconstruction of missing view data; we also demonstrate how to use this approach to perform other meaningful down-stream tasks.

Within the domains we mentioned, the datasets are typically temporal in nature. While we do intend to make use of this temporal structure for our learning approaches, for now, we use techniques such as windowed featurization to circumvent this. We will leave the explicit modeling of such dynamical/temporal systems to our proposed work, which will be discussed in later chapters.

The datasets we consider are:

- 3 Sources News Dataset\(^1\) This dataset consists of featurized news articles from three sources: BBC, Guardian, Reuters.

\(^1\)http://mlg.ucd.ie/datasets/3sources.html
• **NUS-Wide-Lite** [18] This is an image dataset where each image is associated with one or more of 81 concepts like "lake" or "person"; the views are different image featurizations.

• **N-MNIST** [8] This dataset consists of three noisy versions of the original MNIST dataset as the different views.

We will describe these datasets in detail in Section 3.3.

### 3.2 Approaches

We first briefly revisit the relevant methods introduced in the previous chapter for representation learning, and then describe the application of this learned representation in down-stream tasks.

#### 3.2.1 Revisiting Multi-view Representation Learning

**Robust Multi-view Auto-Encoder**

Given the nature of the RMAE [121] as used for representation learning, we can contrast the learned representation with several baseline alternatives to evaluate the robustness and efficacy of the learned latent space.

Here are some baseline alternatives. In general, missing views are treated as just an appropriately sized vector of 0's.

- **Multi-view Feature Concatenation (CAT)**
  A simple alternative method for representation learning is to simply concatenate the multiple available views into a single feature vector. While this approach preserves all the information as contained by the different views, it is not robust to missing views, since it does there is no inter-relationship modeling.

- **Intersection Multi-view Auto-Encoder (IMAE)**
  This baseline is an alternative multi-view AutoEncoder approach which forces all the views to share a single bottle-neck representation. The final code is represented as the average code from all the views. This method tends to extract the intersection of the information contained between the latent spaces, and is thus not as robust to relationships and redundancies local to only a subset of views.

- **Concatenation AutoEncoder (CMAE)**
  Here, we first concatenate the multi-view features just as the first baseline, but train an AutoEncoder above this to learn inter-view relationships. This
approach is the middleground between the RMAE and just simple feature concatenation; and can be seen as skipping the first level of view-specific encoders in the RMAE framework.

While this approach is the closest in spirit to the RMAE, it lacks the initial feature transformation/encoding which often helps unravel inter-view relationship structure.

**Application of Learned Representation** Each baseline method provides an intermediate representation which can then be used to train a down-stream task. All the tasks we describe here are classification, so we train a logistic regression classifier using scikit-learn.

We can then stress-test the learned representations of the different methods for robustness to missing views by producing an encoding with only a subset of the views for the down-stream tasks.

## 3.3 Experiments

### 3.3.1 Datasets

Here, we briefly describe the datasets we consider:

- **3 Sources News Dataset**\(^2\) This dataset consists of news articles as collected from three well-known news sources: BBC, Reuters and The Guardian. Each view news-source is considered a view, and the articles are samples from them. Each article corresponds to some real world event; every news source that publishes an article on the event has a sample for that data point. Not all sources write articles for every event, so we have missing data from some views.

- **NUS-Wide-Lite**\(^3\) This dataset consists of various images and 81 categories that they may contain. These categories are objects, people, types of landscape, etc. so multiple such categories can be present in each image. We take a subsampled version of this dataset with 550 training and test images, and look at detecting three different categories: lakes, people and sunsets. Sub-sampling is done to preserve category prevalence from the main dataset. The views are five different image featurizations: (i) color histogram (CH), (ii) color correlogram (CORR), (iii) edge direction histogram (EDH), (iv) wavelet texture (WT) and (v) block-wise color moments (CM).

- **N-MNIST**\(^4\) This dataset consists of three noisy versions of the original

\(^2\)[http://mlg.ucd.ie/datasets/3sources.html]  
\(^3\)[http://mlg.ucd.ie/datasets/3sources.html]  
\(^4\)[https://csc.lsu.edu/ saikat/n-mnist/]


MNIST dataset. The noise functions are: (i) additive white gaussian noise (AWGN), (ii) motion blur (MB) and (iii) reduced contrast with additive white gaussian noise (RCAWGN). We take a sub-sampled version of the dataset with 6000 training and 600 test points, while preserving digit prevalences.

3.3.2 Evaluation

Down-stream tasks

Here, we look at the performance of the RMAE as a representation learning method for down-stream tasks. We compare the representation learned by RMAE with those learned by CAT, IMAE and CAE in classification/regression performance.

3.3.3 Results

Here, we show plots for training and testing error curves using the different methods. We notice over the different experiments that there is no clear winning method among the ones tested. But RMAE is always either the best or the second best among the testing errors, when all views are available. RMAE’s consistent performance shows that it is able to extract meaningful information about views and their interactions.

Figure 3.1: [3-Source News] Plots for accuracy vs. number of available views for the different approaches.
3.3. EXPERIMENTS

3.3.4 Conclusion

In this chapter, we demonstrated the efficacy of our multi-view relationship learning approaches, both for representation learning and for down-stream tasks.

Figure 3.2: [NUS-WIDE-Lite: Lake] Plots for accuracy vs. number of available views for the different approaches.

Figure 3.3: [NUS-WIDE-Lite: Person] Plots for accuracy vs. number of available views for the different approaches.

In this chapter, we demonstrated the efficacy of our multi-view relationship learning approaches, both for representation learning and for down-stream tasks.
Figure 3.4: [NUS-WIDE-Lite: Sunset] Plots for accuracy vs. number of available views for the different approaches.

Figure 3.5: [N-MNIST] Plots for accuracy vs. number of available views for the different approaches.

In the remaining chapters, we will discuss extending these approaches to respect the temporal structure of the data, as well as additional approaches to tackle
3.3. EXPERIMENTS

these problems.
CHAPTER 3. EXPLOITING LEARNED MULTI-VIEW REPRESENTATIONS FOR DOWN-STREAM
Chapter 4

Toward Improving Multi-view Relationships: Active Search

4.1 Introduction

In the previous chapters, we took our first steps into understanding multi-view relationships and leveraging this understanding for down-stream applications. In this chapter, we will take a step back and once again consider the nature of inter-view relationships. But this time, we will look at ways in which we can manipulate these relationships ourselves to produce meaningful behavior. This could be in the form of favorable properties in these relationships which help us attack a given problem.

One way to produce informative inter-view relationships is through the design or selection of the views themselves. For example, sensor placement on a system such as a self-driving car could be done to optimize for coverage of the surroundings as well as redundancy and overlap for de-noising sensor measurements. However, we are not always at liberty to design such a system ourselves; we are often simply given multi-view data and an associated learning problem.

We may still have some degree of control over inter-view relationships, since our choice of model influences the nature of the relationships uncovered. For example, consider the approach One-Vs-Rest Embedding Learning as described in Chapter 2. Our choice of loss function (eg. reconstruction, correlation) affects the properties of the relationships found between views. In general, we can change how we model the relationship graph between views depending on the problem we are trying to solve. We can also influence inter-view relationships through view selection.

In this chapter, we look at view selection as a means of producing favorable inter-view relationship structure, and discuss Active Search ([127], [122]) a candidate approach.
4.2 Scalable Active Search for View Selection

View selection is similar to directly designing the views, albeit with a lesser degree of control. Instead of directly modeling/modifying the relationships between views, we just choose the views which have favorable/informative relationships between them. This would be useful in applications with a large number of views, where we would want to select only a small subset of views which are relevant to us. These views could be selected to optimize for coverage over the underlying data space, or redundancy across views for reducing noise and dealing with missing view-data. An example of such an applications could be patient medical records, where we can consider the human body as the “underlying system” with different patients as views into this system. Then, we can consider a disease or medical condition as a “data point” and patients recorded with the condition as available views for this point. Another such example would be social media networks, where individual users can be taken as views and their posts as samples; here, real-world events being spoken about can be the “data points” (similar to the news dataset in Chapter 3).

Let us take a closer look at the example of social media networks, like Twitter, as a large multi-view system. As described before, if we consider every user of Twitter as a view, we can take their tweets as samples from the view. Locating influential nodes in social networks and understanding the spread of information are well studied problems [38]. But these problems usually consider the relationship structure between as induced directly by the social network, while we want to leverage the data itself as produced by the users to construct this structure.

Again, we can take real-world events as the true underlying data points. Twitter already provides an easy way to associate tweets to real-world events, namely through hashtags. Given a set of hashtags connected to an event, we can take tweets using those hashtags as corresponding samples from different user-views. With this structure in mind, we can consider view selection as a means to tackle different problems. For one, we can select a subset users to study and contrast inter-view relationships with other characteristics (eg. geographical, occupational, etc.) they may have.

We can also aggregate users who share sentiments on different events into a bigger view, possibly to deal with noise or to augment the data from any one user. This could be done by selecting an initial user and searching for additional users who are similar. Graph-based search approaches are good candidates for such a view-selection strategy. In this case, we do not want to use the social network itself for our graph structure, but rather uncover such a structure in a data-driven fashion. The immediate concern is that social networks are usually prohibitively large to create our own inter-user relationship structure. Most graph-based methods succumb to large amounts of data, given that they usually have a quadratic memory and time dependency on number of points.
In this chapter, we consider Active Search [127] as a candidate algorithm for view-selection. Active Search comes under the Active Learning framework; the algorithm interactively recovers relevant samples, given a small initial set of target samples. To remedy the scalability concern mentioned before, we describe a scalable extension [122] of *Active Search on Graphs* [127].

### 4.2.1 Related Work

Over the past few years, there has been significant research done in semi-supervised active learning. Most of this research is driven towards learning good classifiers given a limited labeled data, as opposed to recovering target points.

Guillory et al. [39] propose methods for selecting labeled vertex sets on a graph in order to predict the labels of other points. Cesa-Bianchi et al. [14] explore an active version of this where they consider the optimal placement of queries on a graph to make minimal mistakes on the unlabeled points.

Zhu et al. [146] propose a method to perform semi-supervised learning on graphs. They formulate their problem in terms of a Gaussian random field on the graph, and efficiently compute the mean of the field which is characterized by a harmonic function. They extend this in [148] to make it active: given the above graphical construction, they query points using a greedy selection scheme to minimize expected classification error. Zhu et al. [147] describe a scalable method to perform inductive learning using harmonic mixtures, while preserving the benefits of graph-based semi-supervised learning.

There has also been some work on optimization-based approaches for semi-supervised classification. Melacci et al. [80] propose a method they call LapSVM, which builds an SVM classifier using the graphical structure of the data. Zhang et al. [144] describe the Prototype Vector Machine which solves a similar objective as above, by approximating it using “prototype” vectors which are representative points in the data. Liu et al. [74] introduce an approach which also considers representative samples from the data called Anchors. They construct an “Anchor Graph”, and make predictions in the main graph based on weighted combinations of predictions on Anchors.

Ma et al. [76] describe new algorithms which are related to the multi-armed bandit problem to perform Active Search on graphs. Their algorithms are based on the $\Sigma$-optimality selection criterion, which queries the point that minimizes the sum of the elements in the predictive covariance as described in [75]. Kushnir [67] also incorporate exploration vs. exploitation in their work on active transductive learning on graphs. They do this by considering random walks on a modified graph which combines the data distribution with their label hypothesis, allowing them to naturally switch from exploring to refinement.

There have also been Active Search approaches which focus on recall instead of classification. Garnett et al. [32] perform Active Search and Active Surveying
using Bayesian Decision theory. Active Surveying seeks to query points to predict the prevalence of a given class.

Closely related to our work is that of Wang et al. \[127\] where they perform Active Search on graphs. They select points by minimizing an energy function over the graph. They also emulate one-step look-ahead by a score reflecting the impact of labeling a point. Our work extends this with crucial modifications allowing us to scale to much larger data sets.

### 4.2.2 Problem Statement

We are given a finite set of \( n \) points \( X = \{x_1, \ldots, x_n\} \), and their unknown labels \( Y = \{y_1, \ldots, y_n\} \) where \( y_i \in \{0, 1\} \). We are also given a similarity function \( \mathcal{K}(\cdot, \cdot) \) between points. We consider the case where this function is linear over some explicit feature transformation \( \phi \):

\[
\mathcal{K}(x_i, x_j) = \phi(x_i)^T \phi(x_j).
\]

This is analogous to the explicit kernel-space representation of some finite-dimensional kernel. This induces a graph over the data: the edge weight between \( x_i \) and \( x_j \) is given by \( \mathcal{K}(x_i, x_j) \).

Initially, we are given a small set of labeled points \( \mathcal{L}_0 \), while the remaining points are in the unlabeled set, \( \mathcal{U} \). Every iteration, we query one point in \( \mathcal{U} \) for its label and move it to the labeled set \( \mathcal{L} \). The goal is to find as many positive points as possible after \( T \) iterations, where \( T \) is a fixed budget for labeling points.

### 4.2.3 Approach

**Background: Active Search on Graphs [ASG]**

We briefly describe the algorithm introduced by Wang et al. \[127\]. They interpret the data as a graph where the edge-weights between points is given by the similarity \( \mathcal{K} \). Their method then uses a harmonic function \( f \) to estimate the label of data points, inspired by the work done by Zhu et al. \[146\]. This is done by minimizing the energy:

\[
E(f) = \sum_{i \in \mathcal{L}} (y_i - f_i)^2 D_{ii} + \lambda \left( w_0 \sum_{i \in \mathcal{U}} (f_i - \pi)^2 D_{ii} + \sum_{i,j} (f_i - f_j)^2 A_{ij} \right) \tag{4.1}
\]

where \( A_{ij} = \mathcal{K}(x_i, x_j) \), \( D_{ii} = \sum_j \mathcal{K}(x_i, x_j) \), and the regularizing constants \( \lambda \) and \( w_0 \) depend on transition probabilities into pseudo-nodes. Explicitly, if \( \eta \) and \( \nu \) are transition probabilities into labeled and unlabeled pseudo-nodes respectively, then \( \lambda = \frac{1-\eta}{\eta} \) and \( w_0 = \nu \). The minimizer is:

\[
f^* = (I - BD^{-1}A)^{-1}(I - B)y', \tag{4.2}
\]
4.2. SCALABLE ACTIVE SEARCH FOR VIEW SELECTION

\[ B = \begin{bmatrix} \frac{\lambda}{1+\lambda} I_L & 0 \\ 0 & \frac{1}{1+w_0} I_U \end{bmatrix}, \quad y' = \begin{bmatrix} y_L \\ \pi \end{bmatrix} \]

For simplicity of notation, \( f^\star \) will simply be denoted by \( f \) moving forward.

To pick points for label queries, ASG uses a heuristic called the Impact Factor which looks at the change of \( f \) values if a given unlabeled point was labeled as positive.

\[ IM_i = f_i \sum_{j \in \{U \setminus i\}} (f_j^+ - f_j) \]

The final selection criterion is \( \arg \max_i f_i + \alpha IM_i \). With this, ASG iteratively queries labels and updates \( f \) and \( IM \). ASG has an \( O(n^3) \) time initialization and \( O(n^2) \) time per-iteration.

Linearized Active Search [LAS]

Here, we describe our algorithm. We now require feature vectors for our points. The similarity function is then assumed to be be linear in these features (or some explicit transformation of them). This requirement is often not too restrictive; in fact, some popular kernels can be approximated using a linear embedding into some feature space. For example, the RBF kernel can be approximated by Random Fourier Features [98]. For simplicity, let \( x_i \) itself represent the feature vector. The similarity between two points is then \( K(x_i, x_j) = x_i^T x_j \).

\textbf{Note:} As mentioned before, ASG requires purely graphical data as input, i.e. the graph adjacency matrix. LAS works with a different class of data, which lives in some multi-dimensional feature space. A graph is induced over the data by the similarity function. If the input to ASG and LAS is the same, the results will be identical. By “the same”, we mean the adjacency matrix for ASG is the same as the one of the induced graph for LAS. In this case, \( f, IM \) and the point queried will be identical at every iteration.

\textbf{Algorithm 1} LAS: Linearized Active Search

<table>
<thead>
<tr>
<th>Input: ( X, L_0, w_0, \lambda, \pi, \alpha, T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U \leftarrow {x_1, \ldots, x_n} \setminus L_0 )</td>
</tr>
<tr>
<td>Initialize ( K^{-1}, f, IM )</td>
</tr>
<tr>
<td>( \text{for } i = 1 \rightarrow T \text{ do} )</td>
</tr>
<tr>
<td>( \text{Query: } x_i \leftarrow \arg \max_{x \in U} (f + \alpha IM) )</td>
</tr>
<tr>
<td>( \text{Update } K^{-1}, f, IM \text{ with } x_i, y_i )</td>
</tr>
<tr>
<td>( \text{Remove } x_i \text{ from } U )</td>
</tr>
<tr>
<td>( \text{end for} )</td>
</tr>
</tbody>
</table>

The pseudo-code is given in Algorithm 1. We now discuss how a linear similarity function helps us update \( f \) efficiently.
Initialization The adjacency matrix is $A = X^T X$ where $X = [x_1 \ldots x_n]$, with $n$ points and $r$ features. Then, $D = diag(X^T X)$. This gives us:

$$f = (I - RX^T)^{-1} q$$

$$R = BD^{-1}, \quad q = (I - B)y'$$

Using the matrix inversion lemma, we get\(^1\):

$$f = q + RX^T K^{-1} Xq$$

$$K = I - XR X^T$$

(4.3)

(4.4)

This converts an $O(n^3)$ time matrix inverse in ASG into the $O(r^3)$ time inverse of $K$. For large datasets, we can expect $r \ll n$. Below, we show that we only need to invert $K$ once; its inverse can be efficiently updated every iteration.

The initialization runs in $O(nr^2 + r^3)$ time for computing $K^{-1}$ and $O(nr^2)$ for computing $f$. Next, we describe our efficient updates to $K$ and $f$ given a new label.

Updates to $f$ on receiving a new label We have $K^{-1} = (I - XR X^T)^{-1}$ at the previous iteration. Only one element in $R$ changes each iteration. Take superscript $\dagger$ to mean the updated value of a variable. We have:

$$R^\dagger = R - \gamma e_i e_i^T$$

where $\gamma = - \left( \frac{\lambda}{1+\lambda} - \frac{1}{1+w_0} \right) D^{-1}_{ii}$ and $e_i$ is the $i^{th}$ standard basis vector. Using the matrix inversion lemma:

$$(K^\dagger)^{-1} = K^{-1} - \frac{\gamma (K^{-1} x_i)(K^{-1} x_i)^T}{1 + \gamma x_i^T K^{-1} x_i}$$

(4.5)

Only one element in $q$ changes: $q^\dagger_i = y_i \frac{1}{1 + \lambda}$. Thus, the update to $f$ can be calculated as\(^2\):

$$f^\dagger = q^\dagger + R^\dagger X^T (K^\dagger)^{-1} Xq^\dagger$$

This takes $O(r^2 + rn)$ time per-iteration as it just involves cascading matrix-vector multiplications.

Impact Factor LAS also includes appropriate modifications for the initialization and updates of the Impact Factor which adhere to the improved running time. We do not describe these here as they are much more involved than those above, while not being fundamentally complicated.\(^3\) We also slightly changed the Impact Factor from ASG: we scaled $IM$ so that it has the same mean as the $f$ vector. This allows us to tune $\alpha$ without worrying about the magnitude of values in $IM$, which varies based on the dataset.

---

\(^1\) Derived in the Appendix.

\(^2\) Updates derived in the Appendix.

\(^3\) Derived in the Appendix.
4.2. **SCALABLE ACTIVE SEARCH FOR VIEW SELECTION**

**Weighted Neighbor Active Search [WNAS]**

Here, we briefly describe a simple and intuitive alternate approach for query selection which also scales well with large amounts of data. This approach is similar to the Nadaraya-Watson kernel regressor:

\[
    f_i = \frac{\sum_{j \in L} y_i \cdot K(x_i, x_j)}{\sum_{j \in L} |K(x_i, x_j)|}
\]

The updates for \( f \) for this approach are simple. We keep track of the numerator and denominator individually for each unlabeled point. Each time we get a new labeled point \( x_i \), we can compute its similarity to all other unlabeled points efficiently as the following vector:

\[
    K(X_U, x_i) = X_u^T x_i
\]

We can then update the numerator and denominator of all unlabeled points directly from this vector. The numerators would be updated by adding \( y_i K(X_U, x_i) \) and the denominators would be updated by adding \( |K(X_U, x_i)| \). These computations require \( O(nr) \) time for initialization and iteration.

### 4.2.4 Experiments

We performed experiments on the following datasets: the CoverType and Adult datasets from the UCI Machine Learning Repository and MNIST.

The Covertype dataset contains multi-class data for different forest cover types. There are around 581,000 points with 54-dimensional features. We take the class with the lowest prevalence of 0.47% as positive. The data is unit normalized across features and a bias feature is appended to give 55 in total. Then, we project these onto a 550-dimensional space using Random Fourier Features [98] to approximate an RBF Kernel.

The Adult dataset consists of census data with the task of predicting whether a person makes over $50k a year or not. It contains 14 features which are categorical or continuous. The continuous features are made categorical by discretization. Each feature is converted into a one-hot representation with \( m \) binary features for \( m \) categories. The features are then unit normalized. The positives are those making more than $50k a year. We modified the dataset size to make the target prevalence 5%. The final dataset has around 63,500 points.

For the MNIST dataset, we combine the training, validation and testing sets into one. The 28x28 pixel images give us 784 features which are then unit normalized. We take the positive class to be the digit 1, and modified its prevalence to be 1%. The final dataset has around 63,500 points.

We compare LAS and WNAS to **Anchor Graph Regularization** with Local Anchor Embedding [AGR] as described in [74]. Their approach creates a proxy

---

This was re-implemented in Python for our experiments.
graph called the Anchor Graph which approximates the larger dataset; the labels
given to points are then a weighted combination of the labels of the anchor points.
Since this is a semi-supervised classification approach, we retrain it every iteration
with all the data and known labels. We then use the confidence values for each
unlabeled point to be positive as the $f$ value. This algorithm requires anchors to be
computed beforehand. For this, we generated k-means over the transformed data
points, with $k = 500$ for each dataset.

Our main experiment measured recall (number of positives found) over a fixed
number of iterations for each dataset. For each dataset, 10 runs were performed
starting with one randomly chosen positive as initialization. For LAS, we took $\alpha$
(the coefficient for the Impact Factor) to be the best from empirical evaluations.
This was $10^{-6}$ for CoverType and Adult, and 0 for MNIST. $\pi$ was taken as the true
positives prevalence.

We also carried out smaller experiments over each dataset where we studied
the predictive performance of LAS vs. WNAS immediately after initialization. Here,
we randomly sampled 100 pairs of one positive and one negative point to initialize.
Then, we reported the number of positives in the top 100 unlabeled points accord-
ing to their $f$-values. These 100 pairs did not include "bad" initializations, where
neither approach found any positives.

**Note:** We did not compare our approach vs. purely graph based methods as in
the [127] Since our results are identical to ASG given the "same" data as described
before, we only considered data with feature vectors.

<table>
<thead>
<tr>
<th></th>
<th>CoverType</th>
<th>MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250</td>
<td>500</td>
</tr>
<tr>
<td>LAS</td>
<td>198.7 ± 32.0</td>
<td>377.8 ± 55.7</td>
</tr>
<tr>
<td>WNAS</td>
<td>188.8 ± 21.5</td>
<td>375.7 ± 37.9</td>
</tr>
<tr>
<td>AGR</td>
<td>27.2 ± 11.2</td>
<td>43.5 ± 11.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Adult</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>LAS</td>
<td>53.7 ± 11.7</td>
</tr>
<tr>
<td>WNAS</td>
<td>46.1 ± 16.3</td>
</tr>
<tr>
<td>AGR</td>
<td>23.1 ± 18.5</td>
</tr>
</tbody>
</table>

Table 4.1: This table shows mean recall ± standard deviation at the middle and
last iteration for each algorithm and dataset.
4.2. SCALABLE ACTIVE SEARCH FOR VIEW SELECTION

Figure 4.1: These plots show recall vs. iteration averaged across 10 runs for LAS, WNAS and AGR, along with ideal and random recall. The left image is for CoverType, the middle image is for MNIST and the right image is for Adult.

<table>
<thead>
<tr>
<th>Dataset (pos%)</th>
<th>LAS</th>
<th>WNAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoverType (0.47%)</td>
<td>4.19</td>
<td>1.66</td>
</tr>
<tr>
<td>MNIST (1.00%)</td>
<td>94.25</td>
<td>60.68</td>
</tr>
<tr>
<td>Adult (5.00%)</td>
<td>27.25</td>
<td>17.29</td>
</tr>
</tbody>
</table>

Table 4.2: This table shows the average positives in the top 100 unlabeled points from the $f$-values of LAS and WNAS.

Results

Figure 4.1 shows plots of the recall per iteration of LAS, WNAS and AGR for the different datasets. Table 4.1 shows mean recall and standard deviation of these experiments in the mid and final iteration. LAS and WNAS both have good performance in all three experiments. The CoverType dataset has high variance in estimates, likely because the data has many scattered positives which are not very informative during initialization. The algorithms would then take longer to
discover the remaining positives. The MNIST data-set showed particularly good performance across the different approaches; all three approaches have near ideal recall. This is likely because the targets are tightly clustered together in the feature-space. The performance of AGR in the CoverType, though much better than random choice, is poorer than the other approaches. This is because AGR incurs significant overhead in the initialization of the algorithm. Computing k-means, followed by the weights and the reduced Laplacian of the Anchor Graph takes a few hours for CoverType. Furthermore, any change in the feature function used between the data points requires recomputation of the Anchor Graph. Due to this, we only used 500 Anchors even though it is a larger data-set. This poorer approximation of the data likely led to worse performance.

Table 4.2 shows the comparison between LAS and WNAS given a single positive and negative point for initialization.

Note: We also conducted experiments on much larger datasets from the UCI Repository: the HIGGS dataset (5.5 million points) and the SUSY dataset (2.5 million points). We have not reported these results. These experiments were not any more informative than those above; they just served as a demonstration of scale.

4.3 Conclusion

In this chapter, we looked toward improving multi-view relationships through view-selection. We described a scalable extension to Active Search, which allows us to interactively select views similar to some initial target subset we may have. Using this, we can find other views of interest given an initial view, for the purpose of de-noising and/or augmenting data from that single view.

Scalability is an important aspect of such an approach, given applications where we have an extremely large number of views. Social media networks is one such application where every user can be interpreted as a view, and their posts as samples from this view.

Further, we can consider metric learning over the similarity function used by Active Search as an additional means to control inter-view relationships. In this case, we would simultaneously search for additional views while improving our model of the relationship induced by the similarity function.

Another point to note here is the ability to re-interpret data-points as views, and vice-versa. We were able to cast social media networks into multi-view system through this interpretation, allowing us to use approaches which are originally tailored toward single-view data. Depending on how we choose to model our views and their relationships, we can use this interpretation to our advantage and employ traditionally single-view techniques to tackle multi-view problems. We will look into this in more detail in the coming chapters.
Part III

Proposed Work
Chapter 5

Revisit to Understanding Multi-view Relationships

5.1 Introduction

In this chapter, we will look at extending and augmenting our approaches for understanding multi-view relationships from Chapter 2. We will continue to look primarily at representation learning as a means to characterizing multi-view relationships.

In Chapter 2, we looked at two approaches, (i) One-vs-Rest Embedding Learning (OVREL) and the (ii) Robust Multi-view AutoEncoder (RMAE). In this chapter, we will first begin with proposing a generative modeling extension of the RMAE, and then consider augmentations to OVREL to better interpret and utilize the models learned.

We will also look at characterizing different views and their relationships in terms of intuitive and meaningful metrics, which may be useful in different applications. For example, we might want to measure the robustness, reliability and/or informativeness of a given view in a given setting.

5.2 Generative Modeling Extensions to RMAE

Our initially proposed approach for robust representation learning involved applying the well-studied idea of "drop-out" in neural networks to views as a whole. While simple, pairing this idea with a suitable architecture showed us promising results in learning representations robust to missing views. Perhaps the more intuitive approach toward learning such a representation would be to model the data distribution itself.

If we can consider the underlying data-distribution as a data-generating process, we can more naturally look at different views as applying "observation" functions to this process. This interpretation also allows us to deal with missing views
gracefully; we can think of missing views as performing a marginalization step.

5.2.1 Flow-based Approaches

The philosophy behind flow-based generative modeling approaches is that a good representation of data is one in which the data distribution is simple and easily modeled. Flow-based approaches ([23], [24]) learn a sequence of invertible transforms over the input data to a latent state where the base distribution has a simple form. These transforms are typically required to all have upper triangular Jacobians, giving a simple relationship between the data distribution and the base distribution, using the change-of-variables formula. Even with this assumption, however, we can still use certain powerful and expressive function classes (including neural networks) to represent these invertible transforms. Compositions of such transforms also satisfy the assumption, which further increases the expressiveness and flexibility of the modeled transforms.

![Diagram of Robust Multi-view AutoEncoder](image)

Figure 5.1: This is the outline of the Robust Multi-view AutoEncoder for 5 views, as shown in Chapter 2. Each bottom arrow represents the encoder for its respective view, and each upper arrow represents the decoder. \( L_i \) is the encoding for view \( i \) and \( L_{all} \) is the global latent representation. The bottom arrows are dotted to represent potential drop-out during training time.

Sampling from this model is very straightforward as long as we can efficiently sample from the base distribution – we just apply the inverses of the learned transforms to samples from the base distribution. Most flow-based approaches typically look at modeling the transforms ([23], [24]), the base distributions or both [90].
We propose to utilize this flow-based framework for our task of robust multi-view representation learning. There have already been extensions of flow-based models to deal with arbitrary subsets missing of data [71], though primarily as missing features in a single-view problem (e.g., missing pixel values for in-painting). We can naturally extend such an approach to multi-view problems, treating views as structured feature subsets of the data.

The same architecture we described in Chapter 2 (shown again in Figure 5.1), can be re-used for this approach. Flow-based compositions of transforms can be used to model the joint encoder which produces the shared latent state $L_{all}$, while the individual view-specific encoders can be independently or jointly optimized auto-encoders.

Simple experiments with synthetic multi-view data show promising signs for using flow-based models to effectively model and sample from the underlying data distribution. However, these initial experiments did not account for missing subsets of views. We propose to extend such flow-based methods to reliably model multi-view data distributions, while gracefully dealing with missing views.

5.2.2 Variational Approaches

We will also look at extending variational approaches, such as the VAE [60], to the generative modeling of multi-view data. Variational methods typically maximize variational lower bounds of the log-likelihood by introducing a parametrized proposal distribution to approximate the posterior of the latent variable given the data. In the case of the VAE, this proposal distribution is essentially the encoder.

This idea can be extended to multi-view data as well, similar to the discussion of flow models above. Since we have two tiers of encoders, we can look at modeling one or both of those tiers as parameterized proposal distributions. Arbitrary conditioning on missing views would follow a similar approach to the VAE-AC [49].

5.3 Relationship Graphs

Another concept that we have brought up several times in the first few chapters is the notion of a relationship graph. Here, we will try to formalize the concept and discuss meaningful applications for such a structure. We give the following tentative definition: A relationship graph is a graph where the nodes are views, or sets of views, and the edges represent a mathematical relationship between them.

Here, a mathematical relationship can be defined by an inter-view metric (e.g., reconstruction error, canonical correlation, etc.) or by a learned multi-view model. In the case of the OVREL, this graph can be represented through the redundancy matrix learned from optimizing over the relevant loss. Here, we would have hyper-
nodes and hyper-edges in the graph since the modeled relationships typically involve more than just two views.

We also looked at the one-vs-one reconstruction error using the RMAE in Chapter 2; we constructed a matrix where each element $(i, j)$ represented the reconstruction error of view $j$ using only view $i$ as input. This matrix can directly be interpreted as the adjacency matrix of a directed graph (or the negative of one). While this is more of an empirical and indirect measure of relationship, such a graph does give us an intuitive understanding of the connections between individual views; the lower the reconstruction error, the more closely related are the views.

Such relationship graphs allow us to operate over explicit connections between views, which opens up analysis through graph-based, spectral and other algorithms. We already described an application of Active Search over such a graph for the purpose of view selection, but we can also consider other problems such as view-clustering.

### 5.3.1 OVREL: Multi-fidelity Relationship Graphs

As mentioned before, a relationship graph induced by projections learned using OVREL would likely contain hyper-edges, since these projections generally involve more than two views. This is because, for every view $i$, we jointly learn projections from all other views to view $i$ to optimize for the objective (eg. reconstruction, correlation, etc.). As we discussed in Chapter 2, we require all of these jointly learned projections together; if we are missing any of the contributing views, we will not produce an embedding. Thus, the projection learned (and consequently, the graph edge) from some view $j$ to view $i$ only makes sense within the context of having all other projections to view $i$.

Further, all these edges are directed, since the projections are not symmetric, i.e. the contribution of view $j$ to the overall projection to view $i$ (edge $j \rightarrow i$) is not necessarily the same as that from view $i$ to view $j$ (edge $i \rightarrow j$). Without using a hyper-graph (i.e. a graph with hyper-edges), we can only represent a single incoming relationship for view $i$ through the projections learned to it; this would be given by all the incoming edges to view $i$ taken together. Here, the term incoming refers to the direction of contribution – the projections learned by optimizing the objective in Equation 2.2 for view $i$ would constitute the incoming relationship to view $i$.

A simple alternative to using a hyper-graph to representing multiple relationships would be to use multiple ordinary graphs. Of course, there could be an exponential number of such relationships given each unique subset of views. But OVREL was primarily designed to tackle this issue by using the group-sparse penalty ($\lambda$ from Equation 2.2) as a means to extract only "important" relationships. Tuning $\lambda$ gives us indirect control over how many other views are considered as contributors for view $i$. At $\lambda = 0$, our learned projections would likely involve all
other views; as we raise $\lambda$, we can expect the number of views contributing to view $i$ to decrease, till only a single other "best" view is used. With this strategy, we can expect there to be at most as many uncovered "important" relationships view $i$ as there are other available views to learn projections from, depending on our choice of $\lambda$.

We can then interpret the number of contributing views (as a proxy for values of $\lambda$), as different tiers of fidelity in the learned projections. Each such tier would be associated with its own relationship graph, representing only a single incoming relationship for a given view. Since the graph is directed, we can use a single graph to represent the incoming relationships for all views of a given tier.

**Greedy view selection**

An alternative to tweaking $\lambda$ would be to directly search over all combinations of a given number of views and choose the best subset based on the final objective value. Of course, this is has the same scalability issues which we wanted to remedy by using group-sparsity. Even with a greedy step-wise approach where we iteratively minimize the residual by adding new views, we would have to do a quadratic (in #views) number of computations.

But we can get away with a brute-force search if we only consider 1 or 2 other views. We can also keep the learned projections from the sub-optimal choices for other views. This helps remedy the problem discussed above where we cannot produce an embedding without any of the constituent projections. If we compute and store all 1-view projections, we will be able to produce an embedding if we have at least one view available.

**5.3.2 Interpreting Relationship Graphs**

As discussed previously, a relationship graph between different views opens us up to applying graph-based techniques for further analysis and modeling. We already discussed one such application in Chapter 4 where we looked at Active Search as a candidate method for view-selection. We can also look at clustering techniques to produce view-clusters which are closely related to each other.

Sensor Networks are another application where Machine Learning has seen an increasing amount of use [1]. Alsheik et al. survey different machine learning algorithms that have been employed over large sensor networks. The use of graph-based methods is typically limited to routing problems over the communication channels connecting various sensors. Using a data-driven relationship graph, we can look at alternative solutions for some of the other problems they mention, such as data compression, anomaly detection, etc.
5.4 Metrics for Qualitative Characterization of Views

It may be helpful to measure the "usefulness" of a particular view within the context of a given multi-view system and a given multi-view task. In this section, we consider some qualitative metrics to characterize views and their relationships. These metrics are generally probabilistic in nature, since they are usually statistics computed over the available data.

We can typically take two approaches to evaluate a view given a metric: (i) compute the metric taking the given view as the only available view (compared to the same for other views) or (ii) compute the metric taking everything but the view (compared to the value of the metric using all views).

1. **Task-affinity**: For tasks with clear and computable objectives such as reconstruction error or classification accuracy, we can directly use this objective itself as a measure of usefulness. I.e., we take the objective value with only (or everything but) a given view and treat this value (or the change in it) as our metric.

2. **Mutual-information**: In the case of a learned latent representation, we can compute the mutual information between a given view and the latent representation produced using all views.

These metrics could be computed online as well, in applications where the system is dynamically changing and the model’s dependence on any given view changes.

5.5 Conclusion

In this chapter, we took a look at special cases for data within application domains, namely (i) temporal structure within views and (ii) asynchronicity between views. We proposed extensions and alternatives to our methods to handle these cases, as well as additional problems which become relevant given the context.

In the last chapter, we will take a brief look at improving multi-view relationships from the perspective of system design.
Chapter 6

Revisit to Exploiting Multi-view Relationships

6.1 Introduction

In many applications, we may have additional information or structure within the problem which we can exploit. In other cases, we may even have to contend with the loss of structure which we assumed had existed. In this chapter, we will look at some specific scenarios and propose strategies to deal with them, either through appropriate changes to our existing methods or new methods altogether.

The two main cases we consider are:

1. **Temporal data**: In this case, we have additional sequential structure within the multi-view data which we can exploit, like in eg. video data, bed-side patient monitoring. This also gives rise new problems which are only relevant with temporal data, such as System Identification and Forecasting. We consider the cases where this temporal structure can apply to all views or only a subset of views.

2. **Asynchronous data**: Here, we look at the case where we the correspondence structure between views, or at least a part of it. We considered "missing" views before in the input data which falls under a similar category, but the implicit assumption was that we had enough synchronized data to model the inter-view relationships. But we may have little or even no synchronization, i.e. every data point is associated with a single view alone; eg. converting horses to zebras and vice-versa [145].
6.2 Temporal Data

In this section, we will take a look at multi-view temporal data and dynamical systems. Multi-view temporal data is very common in robotic and vision, especially with the various multi-sensor and multi-actuator robotic systems we have seen over the past few decades. We also looked at examples in the medical domain with patient health monitoring, medical history, etc. The capability to understanding and/or manipulating such dynamical systems allows us to tackle crucial problems over a wide variety of domains, from robotics control to patient care.

In this section, we will look at both our existing and some new proposed approaches for dealing with the additional structure that temporal data provides. We will also look at some of the new learning problems that come to light when dealing with such systems.

6.2.1 Related Work

There are a few well-established domains which have traditionally looked at multi-modal time-series data, such as Audio-Visual Speech Recognition (AVSR) ([19], [47], [96]), machine translation ([5], [55], [125]), video captioning ([102], [123]), etc. In general, a lot of work has been done in multi-modal translation with sequential data, mainly looking at a few specific types of data: (i) acoustic ([15], [43]), language ([81], [5]) and video ([102], [120]). These approaches are generally heavily tailored to fit the domain, and are usually meant for well-defined problems where the input and output modalities are known ahead of time. For example, the output in machine translation and video captioning is text in a specific language.

As we described in Chapter 1, multi-modal alignment has also seen progress in specific domains. For example, Dynamic Time Warping ([10]) is an well-studied unsupervised method for aligning modalities ([4], [10], [118]), but is limited to two modalities. Other generative modeling ([20], [33], [141]) and neural-network based ([5], [15], [149]) are also used for alignment, but are again typically between two modalities and not easily generalizable outside their application domains.

There has been some interest in more general multi-modal dynamical systems. Boots et al. [11] describe using two-subspace PCA to learn a joint embedding over two views of data, and then apply this to non-linear dynamical systems. In their case, they look at unimodal dynamical systems where the two views are fixed length sequences of features of (i) the future observations, and (ii) the corresponding past observations; but this can be applied to general 2-view problems as long as the noise is uncorrelated.

Hefny et al. [42] use a similar decomposition of past and future observations to model dynamical systems through a sequence of supervised regression problems. These and other similar approaches ([25], [107]) generally use the idea of Predicitcive State Representations (PSRs) [72], where the state of the system is...
expressed in terms of (features of) future observations. These are typically applied for unimodal systems, where they learn a probabilistic model over features of future observations given features of past observations.

6.2.2 Representation Learning

Multi-view representation learning has been a core component of this proposal so far. Here, we will discuss extensions to our approaches to better suit them to temporal data.

Featurizing Sequential Data

First, we will take a look at pre-processing steps to individual views to make it easier to learn multi-view representations.

Windowed Featurization An initial simple approach would be to convert the temporal data into (approximately) i.i.d. data and leaving the models as is. This can be achieved by featurizing fixed-length windows of individual views, aligning corresponding windows across views, and treating the windows themselves as data points. Converting time-windows into features allows us to circumvent directly dealing with the temporal structure, leaving that as a pre-processing step. Featurizations of fixed-length time-windows are quite common; frequency spectra (e.g. using fourier transforms) are quite effective for simple waveform-like time-series, especially since they can be easily converted back into time-window. Another approach for featurizing windows could be using RNNs to learn fixed-length representations, which we will describe next.

Featurizing with RNNs Many recent methods do this for both fixed-length and variable-length sequences ([5], [100], [99], [101], [123]). This is usually done by running an encoder RNN model over an input sequence and taking the last hidden state as the final feature representation; then this representation can be treated as the initial state (or additional input) to feed into another decoder RNN model to reproduce the sequence. These approaches typically also have the advantage of being inverted (or decoded) back to the time-sequence.

Learning Representations over Featurized Time-Sequences

With featured windows, we can just use our methods as-is. In the general case where we would like to produce a state/featurization at any point of a sequence, we can also use the same strategy as with the RNNs and hidden state. We can represent the state of a time-sequence at any time-stamp by running an RNN over it and just using the corresponding hidden state as the feature/state representation.
In this case, this featurization is no longer i.i.d., so our models will need to be modified appropriately.

The RMAE model as described in Chapter 2 can be modified by replacing the individual and shared encoders/decoders with Recurrent Auto-Encoders (eg. [21], [26]). These are typically straightforward extensions of sequence-to-sequence models, usually by either trying to predict the next element in the sequence or by encoding into (and decoding from) a fixed length sequence as described before.

We can have two types of “missing” views in this case, where either an entire view is missing or only a subset of time-stamps are missing for a view. Both these can be modeled using the RMAE framework, by applying either dropping strategy while training, to encourage robustness of representation.

### 6.2.3 Multi-view Tasks with Temporal Data

Here, we quickly describe a few down-stream tasks which become relevant with temporal view data. Generally, we can look at these problems in two steps: First, we use our multi-view representation learning approaches to build a shared representation for the data. Second, use the learned representation as a single view to apply the relevant unimodal techniques.

The alternative is to jointly learn a representation and optimize the task objective. We will compare and contrast these strategies using the proposed methods.

**System Identification**

This is essentially the same as representation learning itself, since we are simply modeling dynamical system. Our learned representation can be directly used as a characterization of the state of the system at a given point.

**Forecasting**

The learned representation is generally also a time-sequence, just one which is shared across all views. We can use standard forecasting methods, from AR models to RNNs, to forecast within the learned representation space. Individual view-decoders can then be used to convert the forecasted learned representations into forecasts for the different view.

### 6.3 Asynchronous Data

In this section, we look at the case where we have little to no correspondence between data points from different views. We have looked at views with missing
data, but we generally assumed that we had enough to still be able to model inter-view relationships. But when this assumption breaks down, we need to provide some scaffolding to our methods to make up for the loss in the inter-view structure.

6.3.1 Adversarial Training

GANs have seen recent success for translating between domains without having paired training data ([145], [45], [132]). The general idea for using adversarial training in this context is to simultaneously train a generator to convert samples from one domain to another and a discriminator to distinguish between domains. Cycle consistency is also optimized for, which tries to ensure that the generators trained are invertible. This has also been extended to more than two domains ([17]), and even for specific multi-view time-series problems like voice conversion ([56]).

These approaches have generally seen effective use for unpaired image-to-image translation, outside of specific problems such as voice conversion. Since domain translation itself is the primary task, they typically do not learn an explicit latent representation, and instead translate points directly between domains. If we consider cases where there are other downstream applications which use the data, it can be useful to introduce the intermediate step of representation learning.

6.3.2 Asynchronous Representation Learning

We propose to take a look at adversarial training as a means to build a representation model of multi-view data, with application to other downstream tasks. We can do this by looking at (encoder, generator, discriminator) triplets for each view, where we have the inclusion of an explicit latent state to related the different views. For each view:

- The encoder takes a data point in the given view and produces a view-agnostic latent representation.
- The generator takes this representation and generates a data-point from the given view.
- The discriminator tries to identify whether any given sample is a true sample from the view, or a synthetic one generated by one of the view-generators.

We can enforce cycle consistency just as in CycleGANs, by training the generators to produce samples which give back the latent state when encoded.

We will look at this as a viable representation learning model for asynchronous multi-view data, and discuss applications, and modifications for special cases such as dynamical systems.
Chapter 7

Improving Multi-view Relationships

7.1 Introduction

In Chapter 4, we looked at how we can use a graph-based method, namely Active Search, to help with the task view-selection. View-selection was one way we looked to improve inter-view relationships in multi-view data. In this chapter, we look at additional methods which work to improve the relationships between views.

We first describe how we can reinterpret views as data-points, and use this reinterpretation to use other, standard methods to understand multi-view relationships. Then, we discuss a few ways in which we can use this idea to model and improve multi-view relationships, for example, through graph-based methods and metric learning.

Lastly, we look at how we can apply such techniques online, to build an adaptive multi-view system. This could be useful in low-bandwidth settings, (e.g., space probes) where we might need to choose the most relevant sensor measurements to transmit, given the current environment.

7.2 View Duality

In Chapter 4, we saw how we could interpret views themselves as data-points. This allowed us to use Active Search, a graph-based method, for view-selection. Here, we will try to formalize this notion, and look at ways we can apply this reinterpretation to analyze and learn over multi-view relationships. First, we will distinguish between the views themselves and the samples from the views. Let us revisit our notation as described in Chapter 1; We will use $X_i$ to denote the view $i$ itself, and $x_i^j$ to denote the $j^{th}$ sample from view $i$. The multi-view dataset contains $N$ points, with corresponding samples in each view. We assume that no samples are missing in each view, though this will not affect most of our discussion since we mainly look at statistics over the samples.
So far, we have discussed representation learning from the perspective of modeling the samples, or sample distribution, from the views. Now, we take a step back and describe a framework over modeling the views directly. We can consider modeling views through statistics as calculated over the samples of the views. Writing these as functions over the samples, we would have the following form for view $i$:

$$f(X_i) = \phi(\{x_{i1}^1, \ldots, x_{iN}^i\})$$  \hfill (7.1)

for some feature function $\phi$ over sets. We can also model the interactions between views similarly through such sample statistics. For view $i$ and view $j$, we can model their interaction as:

$$g(X_i, X_j) = K(\psi(\{x_{i1}^1, \ldots, x_{iN}^i\}), \psi(\{x_{j1}^j, \ldots, x_{jN}^j\}))$$  \hfill (7.2)

where $\psi$ is another feature function over sets, and $K$ is a kernel function or some other measure of similarity. This formulation allows us to model views and their relationships through statistics of their samples. For example, if we can approximate $K$ to be linear for some choice of $\psi$, we can apply our Scalable Active Search [122] method as described in Chapter 4. In general, we can look at approaches for learning over sets and distributions as candidate models.

7.2.1 Modeling over Sets and Distributions

Learning over distributions has received a lot of attention in recent years. Sutherland et al. [117] discuss approximate kernel embeddings which can be computed in linear-time; this is applicable to Scalable Active Search as mentioned before. Maundet et al. [84] review various approaches for producing kernel mean embeddings. For modeling sets, Zaheer et al. [142] discuss necessary and sufficient conditions required for functions over sets and propose deep models which satisfy these conditions.

7.2.2 Learnable Relationships

Given tools to model sets, and their interactions, we have what we need to model learnable inter-view relationships. We can apply kernel learning as a metric learning problem [66], or look at multi-kernel learning [37], [112] techniques to model relevant relationships between views. These approaches also allow natural integration of prior knowledge, for example, in the form of triplet constraints. We can also use the Deep Sets framework [142] to have trainable set-functions which can optimized given the context of the problem.

Another paradigm to analyze and model the relationship between views is through causal discovery and causality-based modeling. Glymour et al. [36] review various methods for causal discovery and causality-based modeling based on
graphical models. Huang et al. [46] even propose an approach to causal discovery over non-stationary data distributions, which is often the case with dynamical systems.

7.3 Online Improvement

Lastly, we briefly discuss applying such multi-view relationship learning techniques in an online fashion. This is especially relevant in the case where the underlying system is constantly changing and our model of the multi-view system needs to be updated often. For example, if we consider a space probe or surface rover for exploration, it is likely that we will encounter environments which we may not be familiar with. That, paired with the low-bandwidth setting of extremely long-distance communication, might require use to adapt our choice of views and their relationships on the fly.

Online metric learning is a natural extension of our discussion in the previous section, and has been studied before for various applications ([50], [31]). Online view selection and anomaly detection are also relevant to consider for an application with a heterogeneous underlying data distribution.

7.4 Conclusion

In this chapter, we looked at improving multi-view relationships through modeling and optimizing their interactions directly. We first introduce the notion of view-data duality where we argue that we can interpret views as data points and vice-verse, which allows us to use different algorithms in our analysis of multi-view data. Such a formulation is conducive to applying metric learning and other such methods to form relationships between views.

We also briefly look at the case of online multi-view relationship learning, which is relevant in the case of a heterogeneous underlying system and changing application objectives.
Part IV
Timeline
Chapter 8

Timeline

This is an estimated 16 month time-line from July 2020 to October 2021 for the completion of the various tasks. Here, Part I, Part II and Part III refer to Understanding (Chapter 5), Exploiting (Chapter 6) and Improving (Chapter 7) interview relationships.

We will spend the first 6 months looking at generative modeling extensions of RMAE and as well as Relationship Graph construction and analysis. We expect the application to temporal and asynchronous data to take the most time, so expect an 8 month time frame split between the two. We will spend the last 3 months looking at relationship learning and online improvement.

I plan to use the three deadlines groups of D1: ICLR/AAAI/AISTATS (Sep/Oct 2020), D2: ICML/UAI/IJAI (Jan/Feb 2021) and D3: NeurIPS (May 2021) as progress checkpoints over this timeline.

I expect to take a month at the end to assemble my dissertation, with my thesis defense around October 2021.
CHAPTER 8. TIMELINE

MONTHS: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

**Part I**
- Gen. RMAE
- Rel. Graph Analysis

**Part II**
- Temporal data
- Async. data

**Part III**
- Rel. Learning
- Online Improvement
- Document prep.

D1 D2 D3 Defense


[130] Wöllmer, M., Metallinou, A., Eyben, F., Schuller, B., and Narayanan, S. Context-sensitive multimodal emotion recognition from


