ABSTRACT

Title of dissertation: RESOURCE ALLOCATION IN COMPUTER VISION

Daozheng Chen, Doctor of Philosophy, 2013

Dissertation directed by: Professor David Jacobs
Department of Computer Science

We broadly examine resource allocation in several computer vision problems. We consider human resource or computational resource constraints. Human resources, such as human operators monitoring a camera network, provide reliable information, but are typically limited by the huge amount of data to be processed. Computational resources refer to the resources used by machines, such as running time, to execute the programs. It is important to develop algorithms to make effective use of these resources in computer vision applications.

We approach human resource constraints with a frame retrieval problem in a camera network. This work addresses the problem of using active inference to direct human attention in searching a camera network for people that match a query image. We find that by representing the camera network using a graphical model, we can more accurately determine which video frames match the query, and improve our ability to direct human attention. We experiment with different methods to determine from which frames to sample expert information from humans, and discover that a method that learns to predict which frame is misclassified gives the best performance.

We approach the problem of allocating computational resource in a video processing task. We consider a video processing application in which we combine the outputs from two algorithms so that the budget-limited computationally more expensive algorithm is run in the most useful video frames to maximize processing performance. We model the video frames as a chain graphical model and extend a dynamic programming algorithm to determine on which frames to run the more expensive algorithm. We perform experiments on moving object detection and face detection to demonstrate the effectiveness of our approaches.

Finally, we consider an idea for saving computational resources and maintaining program performance. We work on a problem of learning model complexity in latent variable models. Specifically, we learn the latent variable state space complexity in latent support vector machines using group norm regularization. We apply our method to handwritten digit recognition and object detection with deformable part models. Our approach reduces latent variable state size and performs faster inference with similar or better performance.

RESOURCE ALLOCATION IN COMPUTER VISION

by

Daozheng Chen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor David Jacobs, Chair/Advisor
Professor Dhruv Batra
Professor Hal Daumé III
Professor Larry Davis
Professor Lise Getoor
Professor Min Wu, Dean's Representative

To my parents, Keming Chen and Yun Zhang:

for their care over all these years.

Acknowledgments

I want to thank my advisor, David Jacobs, for providing me an environment that allowed me to pursue my own research ideas and guided me through all the research obstacles over the years. David is always patient to listen to my half-baked ideas, active to help me to obtain internships and future career opportunities, and cares about my life at UMD. I cannot imagine what my graduate study would be without David's kind support! I also thank Lise Getoor who is always willing to provide me with research ideas and warmly encourages me to keep working despite temporary slow research progress. I appreciate Dhruv Batra who guided my research with many technical details and allowed me to quickly pick up a large amount of material in continuous optimization. I am also grateful to Mustafa Bilgic who is always available for discussion and suggested useful ideas when we worked on the camera network project together. I also would like to thank William Freeman who kindly provided support and research advice during my stay at MIT. I also thank Kimo Johnson who suggested many practical skills that allowed me to work efficiently and also guided me to behave gently. I would like to thank my supervisors, Shaolei Feng and Kevin Zhou, at Siemens Corporate Research, where I gained very valuable industrial experience under their guidance. I also thank the technical supporting staff at UMIACS, MIT, TTIC, Siemens, and the CS department at UMD. Without their professional help to maintain data and machines, I could not imagine how many more problems I would have had. Lastly, I want to show my appreciation to my Father, Keming Chen, and my Mother, Yun Zhang, who diligently raised me up and supported my college education in the United States over the years. Thank you all and thank God!

Table of Contents

List of Figures

Chapter 1

Introduction

New technology is giving rise to large stores of digital images and videos. Their size has increased much faster than the computational resources needed to effectively process them. For example, Flickr, an image hosting and video hosting website, reports that it was hosting more than 6 billion images in August 2011 and this number continues to grow steadily [68]. At the same time, as we develop increasingly more sophisticated and accurate vision algorithms, they also demand greater computational resources. Consequently, it is important to develop strategies for applying vision algorithms with greater efficiency to image and video data.

In this thesis, we broadly examine this resource allocation in several computer vision problems. Specifically, we consider human resource and computational resource constraints. Human resources, such as human operators monitoring a camera network, provide very reliable information. Such resources are typically limited by the huge amount of data it is necessary to process. For example, in the terrorist attack in London in 2005, British police were required to examine 80,000 CCTV tapes from the city's camera network system [60] to discover the image of a bomber after the [80]. It is very useful to direct human attention to portions of videos, which are most likely to be informative.

We approach this constrained setting with a frame retrieval problem in a camera network. Given a target image for an object, we want to retrieve all video frames in the

camera network, which contain this object. We consider an active inference approach, in which the model can collect expert information, such as human annotation, over this camera network at inference time [76]. The goal is to direct the budget limited human information in such a way so that an objective, such as prediction accuracy, is optimized.

Computational resources refer to the resources used by the machines, such as memory and running time, to execute programs for the algorithms. Sophisticated vision algorithms can handle many vision tasks with good performance guarantees. However, they typically require longer running times. It can therefore be impractical to run such algorithms if a large amount of data is given, such as the scenario that occurred during the London bombings in 2005 [60, 80].

We approach this problem in a video processing applications in which we combine outputs from two algorithms to analyze videos in which one algorithm is computationally more expensive and, therefore, can be run only a limited number of times. We approach this problem by modeling the output from both algorithms with a chain graphical model. The inference output from this model decides where to run the more resource-consuming algorithm.

In the above two applications, we impose a budget on the resources we want to allocate. We also consider another idea for saving computational resources and maintaining program performance without explicitly specifying the budget. Specifically, we show that by reducing variable state space dimensionality, we can build models that consume less computational resources during inference with similar prediction performance. We ground this framework in latent structural support vector machines. We consider learning the model regularized by group norm so that the dimensionality of the state space of

latent variables is reduced and faster inference with similar accuracy is performed. We apply our approach to handwritten digit recognition and object detection to demonstrate the effectiveness of our method.

The contribution of this thesis is as follows. Chapter 2 describes the approach using active inference to direct human attention to perform monitoring tasks over a camera network. We approach this by first mapping video frames in a camera network onto a graphical model. This allows us to perform more effective inference than when each frame is independently analyzed. More importantly, as a human operator examines frames of the video, her input can improve classification of portions of the video that have not yet been analyzed. In addition, we can use active inference algorithms to direct attention towards the most useful portions of the video. Our primary contribution is therefore to show that by modeling video frames using a graphical model, we can perform collective classification and active inference to perform more effective video analysis in camera networks.

Chapter 3 describes the work of combining two algorithms to analyze videos. Our primary contribution is a new algorithm that determines where in a video to apply the expensive algorithm. We build on prior work by Krause and Guestrin [48], which shows that one can use a dynamic programming algorithm to determine the optimal places at which to make observations in a first-order Markov chain. We modify this algorithm so that it can be run efficiently over a Markov chain with thousands of nodes. Our final contribution is to experimentally demonstrate the value of this algorithm in two very different vision tasks: motion and face detection. We show that our algorithm can be used to significantly improve performance.

Chapter 4 describes work to learn the complexity of a latent variable state space. In this work, our primary contribution is to propose the use of structured sparsity inducing norms like ℓ_1-ℓ_2 to estimate the parameters of a latent-variable model, thereby regularizing the complexity of the latent space. We apply our approach to latent support vector machines, for both the binary and structured output case. We perform two sets of experiments: handwritten digit recognition on MNIST and object detection with deformable part models on the PASCAL VOC 2007 dataset [28]. Our first set of experiments shows that our approach is indeed able to prune the complexity of latent space, resulting in a model that allows significantly faster inference at test time without a drop in accuracy over a complete (non-sparse) model. Our second set of experiments shows that our approach is able to learn a better model by adapting the complexity of the latent variable space to the category being trained. Finally, Chapter 5 concludes this thesis and describes future work.

Chapter 2

Active Inference for Retrieval in Camera Networks

The work from this chapter was published in the IEEE Workshop on Person-Oriented Vision (POV) in January 2011, [17].

2.1 Introduction

In this chapter, we show how we can discover and exploit spatial and temporal relationships among frames in a camera network, and we study the use of active inference, which can be used to direct human labeling efforts to portions of videos whose labels will provide the biggest performance improvements. We consider a frame from a camera network to be "relevant" if it contains a queried person; the retrieval task is to identify all of the relevant frames.

We achieve this goal by first mapping video frames in a camera network onto a graphical model. This allows us to perform more effective inference than when each frame is independently analyzed. More importantly, as a human operator examines video frames, her input can improve the classification of portions of the videos that have not yet been analyzed. In addition, we can use active inference algorithms to direct attention towards the most useful portions of the videos. Our primary contribution is, therefore, to show that by modeling video frames using a graphical model, we can perform collective classification and active inference to produce more effective video analysis in camera

networks.

Specifically, our contributions are as follows:

- We describe how to model retrieval in a camera network using a graphical model.
- We develop active inference techniques to prioritize frames for human annotation.
- We empirically show that, among the several active inference approaches we consider, the technique that most heavily exploits the network structure gives the best performance.

This chapter is organized as follows. We discuss related work in Section 2.2. Then, we formulate the problem in Section 2.3. We describe two probabilistic frameworks for video analysis in camera networks in Section 2.4. Next, we describe the active inference techniques in Section 2.5. Section 2.6 discusses the experimental setup and results, and we conclude in Section 2.7.

2.2 Prior Work

Person reidentification, in which a person seen in one surveillance video is located in later ones, closely resembles the query problem we address. Wang et al. [102] use shape and appearance context to model the spatial relationships of object parts to do appearance-based person reidentification. Gray and Tao [36] design a feature space and use Adaboost to learn person reidentification classifiers. Lin and Davis [53] reidentify people by a pairwise comparison-based learning and classification approach. Loy et al. [58] facilitate human reidentification by using cross canonical correlation analysis to learn the spatial

temporal relationship of video frames in a camera network. In contrast, local descriptors have been widely used in object recognition. In particular, Sivic and Zisserman [86] consider video retrieval using a bag-of-words model.

Other work has used graphical models to represent camera networks. Loy et al. [57] performs abnormal event detection by modeling regions from different camera views using a time delayed probabilistic graphical model. Chen et al. [15] use a conditional random field (CRF) to model a camera network identify a known set of people.

Tracking over camera networks has also been widely addressed (eg., [29, 42, 62, 87, 91]). Typical problems include inferring the topology of the camera network [29, 62, 91] and finding correspondences between trajectories from multiple cameras [42, 87].

The key difference between our work and these is the use of collective classification and active inference to handle queries to a camera network. In a very different context, some of these issues have been addressed in interactive segmentation. For example, Rother [78] extends the graph-cut method [10] with substantially simplified user interaction to achieve superior image segmentation quality. Wang et al. [100] interactively extract foreground objects from a video using user painted strokes. While this work focuses on minimizing the need for human labeling over still images or a single video, we focus on active inference methods that can direct human attention over camera networks.

Krause and Guestrin [48] did a theoretical analysis of active inference for graphical models and they showed that the optimal solution is tractable for Hidden Markov Models, but it is $\mathbf{NP}^{\mathbf{PP}}$-hard for graphical models even with a polytree structure. Rattigan et al. [76] performed active inference on networks of arbitrary structure by first grouping the nodes of the network into clusters and then acquiring the labels of the central nodes

in the clusters. Finally, the active learning work [84] is very related to active inference, and it has been applied to visual recognition [96, 97]; however, the biggest difference is that active learning acquires labels to construct training data to learn a model, whereas active inference performs label acquisition for an already learned model to guide the probabilistic inference to achieve better accuracy and precision.

2.3 Problem Formulation

Let $\mathcal{C}$ denote a network of cameras and let $\mathcal{F}_C$ represent the set of frames taken by camera $C \in \mathcal{C}$. Each frame $F \in \mathcal{F}_C$ is represented by a feature vector $\vec{X}_F = \langle X_F^1, X_F^2, \ldots, X_F^p \rangle$ and class label Y_F pair, $F = \langle \vec{X}_F, Y_F \rangle$. Here, the $\vec{X}_F$ are continuous variables; these variables can depend on the specific query that is being processed, and indicate the similarity between the query image and a video frame (described further in Section 2.6.3). Each Y_F is binary, indicating whether F is *relevant* or *irrelevant* to a query.

Given training data $\mathcal{D}^{tr} = \{\langle \vec{X}_F^{tr}, Y_F^{tr} \rangle\}$ for $F \in \mathcal{F}_C, C \in \mathcal{C}$, a test set $\mathcal{D}^{te} = \{\langle \vec{X}_F^{te}, Y_F^{te} \rangle\}$, and a budget B determining the number of labels a human annotator can provide, our objective is to determine the best set of labels $\mathcal{A} \subseteq \mathcal{Y}^{te}$ to acquire as follows:

$$\underset{\mathcal{A} \subseteq \mathcal{Y}^{te}, |\mathcal{A}| \leq B}{\operatorname{argmax}} \; Reward(S^{te} \mid \mathcal{X}^{te}, \mathcal{Y}^{te}, \mathcal{A})$$

where $\mathcal{Y}^{tr}$ and $\mathcal{Y}^{te}$ represent the set of labels for the frames from the training and testing data respectively, $S^{te} = \{L_1, L_2, \cdots, L_N\}$ is the set of random variables for the label of each of the N testing instance, and $\mathcal{X}^{tr}$ and $\mathcal{X}^{te}$ have similar meaning for sets of features. In practice, this reward function is based on the conditional probabilities of the

labels given observed, acquired and inferred information. We use a probabilistic model to estimate these probabilities. We consider two types of $Reward$ functions in this paper; the first one is accuracy, measuring the percentage of frames in $\mathcal{Y}^{te}$ that are correctly classified. The second one is average precision, measuring how well the model can rank the frames in order of relevance.

2.4 Probabilistic Models

We will contrast two probabilistic models for video retrieval in a camera network.

2.4.1 Local Models (LM)

In the simplest case, we assume that, given the parameters of the underlying model, the labels of all frames in the network are independent of one another, given the features of the frame. Thus, in this model, we assume that each Y_F depends only on $\vec{X}_F$ and nothing else. Because this model uses only the local information about the current frame, we call it a *local model* (LM).

For estimating $P(Y_F \mid \vec{X}_F)$, any probabilistic classifier that can be trained discriminatively, such as logistic regression, can be used. In our experiments, we use a visual bag-of-words model [86], which has been shown useful for video image retrieval. The query image and video frames are represented as vectors of visual word frequencies. We then compute cosine similarity between these frequency vectors to represent $\vec{X}_F$, which is then used as input to the probabilistic classifier. We provide more details in Section 2.6.3.

2.4.2 Relational Models (RM)

Because one person is typically present or not present for a duration of time in a camera, and because cameras have overlapping and non-overlapping fields of view, we expect the above independence assumption to miss important relationships in the data. So we also consider a *relational model* (RM), where the information from *neighboring* frames is integrated. Specifically, to predict the label Y_F, we use the label information from three types of neighbors, which we define below.

1. **Temporal neighbors $\mathcal{N}_{Y_F}^{T^k}$:** These are the labels of the frames that appear k time steps before frame F and k time steps after it in the same camera C.

2. **Positively correlated spatial neighbors $\mathcal{N}_{Y_F}^{P}$:** These are the labels of the frames from other cameras at the same time step that tend to have the same label as F. Such neighbors may correspond to cameras with overlapping fields of view and can be discovered from the training data.

3. **Negatively correlated spatial neighbors $\mathcal{N}_{Y_F}^{N}$:** These are the labels of the frames from other cameras at the same time step that tend to have labels different from Y_F. For example, when cameras have non-overlapping fields of view, a person can be present in at most one camera. Such neighbors can also be discovered automatically.

The set of neighbors of Y_F is then defined as $\mathcal{N}_{Y_F} = \mathcal{N}_{Y_F}^{T^k} \cup \mathcal{N}_{Y_F}^{P} \cup \mathcal{N}_{Y_F}^{N}$. Relational models use both $\vec{X}_F$ and $\mathcal{N}_{Y_F}$ to predict Y_F. However, because the neighbor labels are also often unobserved, the labels in the test data need to be inferred collectively. *Collective*

classification is the process of using a relational model to infer the labels in a network simultaneously, exploiting the relationships between the network entities (see [83] for an overview). In this paper, we use Iterative Classification Algorithm (ICA). We describe it below.

ICA uses two models, a local model and relational model, to infer the labels of related entities iteratively. It learns a local model that uses only $\vec{X}_F$ to bootstrap the labels, and then applies a relational model that uses both $\vec{X}_F$ and $\mathcal{N}_{Y_F}$ to propagate the labels to neighboring frames in the network iteratively. Specifically, the relational model component of ICA represents each frame F as a vector that is a combination of $\vec{X}_F$ and features that are constructed using $\mathcal{N}_{Y_F}$.

Because frames from different cameras can have varying numbers of neighbors, the combined feature vector $\langle \vec{X}_F, \mathcal{N}_{Y_F} \rangle$ will be of different length for different frames. To get a fixed-length vector representation, we use an aggregation function `aggr` over the neighbor labels. For example, `count` aggregation constructs a fixed-size feature vector by counting the number of neighbors with each label. With this aggregation, we build the following combined feature vector: $\vec{X'}_F = \langle \vec{X}_F, \texttt{aggr}(\mathcal{N}^{T^k}_{Y_F}), \texttt{aggr}(\mathcal{N}^{P}_{Y_F}), \texttt{aggr}(\mathcal{N}^{N}_{Y_F}) \rangle$. Once the features are constructed, then an off-the-shelf probabilistic classifier can be used to learn $P(Y_F \mid \vec{X'}_F)$. Despite its simplicity, ICA has been shown to be quite effective and efficient [59, 66].

2.4.2.1 Choosing Neighborhoods for Cameras

In this paper, we use the explicit time information in each camera to define the temporal neighborhood. Let $F^t_{C_i}$ represent the frame from camera C_i at time step t. Then,

$$\mathcal{N}^{T^k}_{Y_{F^t_{C_i}}} = \{Y_{F^{t-k}_{C_i}}, Y_{F^{t-k-1}_{C_i}}, \ldots, Y_{F^{t-1}_{C_i}}, Y_{F^{t+1}_{C_i}},$$

$$Y_{F^{t+2}_{C_i}}, \ldots, Y_{F^{t+k}_{C_i}}\}$$

We learn the positive-spatial and negative-spatial neighborhoods from the data as follows. Let $\mathcal{Y}_{C_i}$ represent the temporally ordered set of all frames from camera C_i in the testing data. Then,

$$\mathcal{N}^{P}_{Y_{F^t_{C_i}}} = \{Y_{F^t_{C_j}} \mid corr(\mathcal{Y}_{C_i}, \mathcal{Y}_{C_j}) > \sigma_p\}$$

and

$$\mathcal{N}^{N}_{Y_{F^t_{C_i}}} = \{Y_{F^t_{C_j}} \mid corr(\mathcal{Y}_{C_i}, \mathcal{Y}_{C_j}) < \sigma_n\}$$

where $corr(.,.)$ measures the correlation between two ordered sets, and σ_p and σ_n are threshold parameters that define whether a camera should be included as a neighbor.

2.5 Active Inference

We allow the underlying retrieval algorithm to request the correct labels for some frames at inference time. This setup is called "active inference" meaning that the underlying model can actively collect more information at inference time [76]. The goal is to make the most of available human resources. We would like to determine for which frames the probabilistic model should request labels so it can label the remaining frames

as well as possible. In this section, we describe a general framework for active inference and several possible algorithms for video analysis.

We considered the following active inference techniques:

1. **Random sampling (**RND**):** Sample frames randomly across different cameras and time steps.

2. **Uniform sampling (**UNI**):** Sample frames uniformly over time, for each camera.

3. **Most relevant (**MR**):** Sample frames whose probability of being relevant is the highest, where the probability is based on the output of the probabilistic model.

4. **Uncertainty sampling (**UNC**):** Sample the frames whose entropy value is the highest, where the entropy is defined using the probability estimates of the probabilistic model.

5. **Most likely incorrect (**MLI**):** Sample the frames that are most likely to be incorrectly predicted. For this, we adapt the reflect-and-correct algorithm (RAC) [9], which uses a separately trained classifier to predict which instances in a general network classification problem are likely to be misclassified. Below we describe how we adapt RAC for the purposes of retrieval in a camera network.

The first four methods can be applied to both LM and RM, and our experiments demonstrate that RM with relational information outperforms LM significantly. Because MLI is based on RAC, which specifically targets collective classification, it can be applied with only RM.

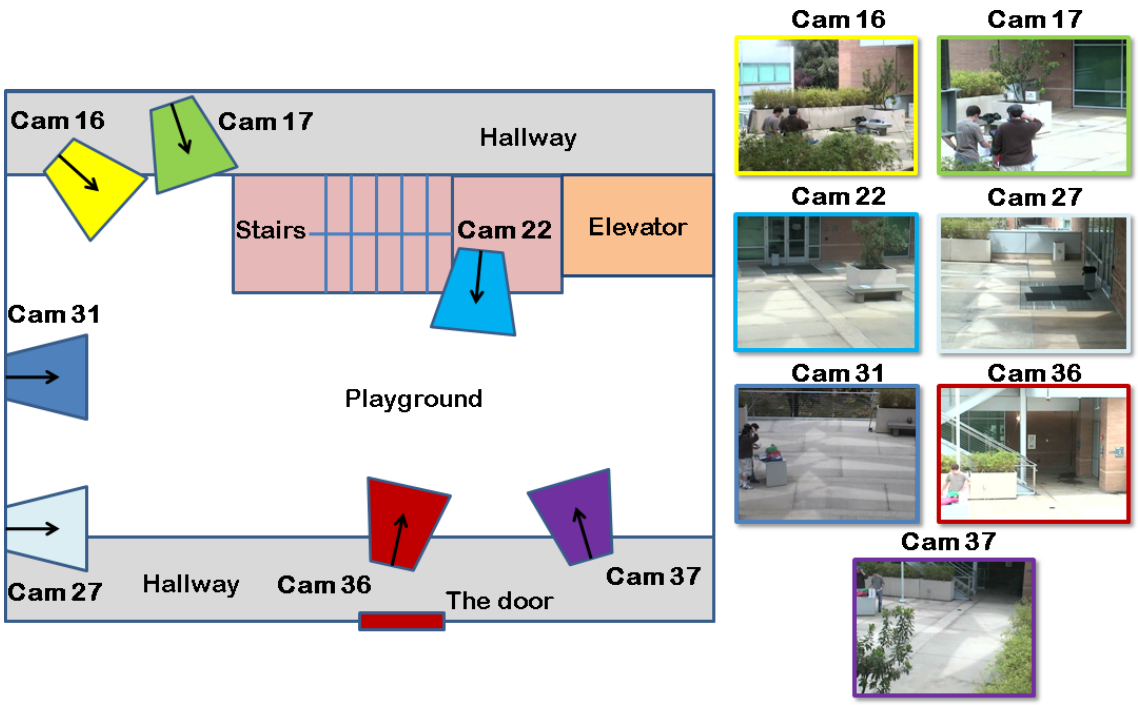


Figure 2.1: Camera layout and sample frames from each of the 7 cameras. The camera ID above each frame is the actual ID used in the UCR Videoweb dataset.

2.5.1 Adapting RAC for Retrieval in Camera Networks

RAC is an active inference technique that works as follows. At training time, a separate classifier is trained to predict whether or not an instance is misclassified. Then, at inference time, RAC uses this classifier to predict at what instances RM is likely to make a mistake, and suggests acquiring the label of a central instance in a region where most of the instances are predicted to be misclassified [9]. To learn and predict whether an instance is misclassified, RAC utilizes a few features that are indicative of misclassification. At a higher level, these features are based on the RM prediction, LM prediction, and some global statistics about the data. RAC learns the classification function using the trainind data used for training RM.

In this paper, we introduce two important modifications of the original RAC framework. These address i) what features to use for RAC in camera networks and ii) how to train RAC. To distinguish this adapted version from the original, we refer to our version as *Most Likely Incorrect* (MLI).

2.5.1.1 Features for MLI

We used the following 10 features as possible indicators of misclassification:

1. Four features based on the probability estimates of RM. We use the entropy of the probability estimate for the single label Y_F and average entropy values over $\mathcal{N}_{Y_F}^{T^k}$, $\mathcal{N}_{Y_F}^{P}$, and $\mathcal{N}_{Y_F}^{N}$. These features capture the uncertainty of the frame and uncertainty of its neighborhood.

2. Four features based on the probability estimates of RM and LM. We use the KL divergence between the RM probability and LM probability for label Y_F, and the average of this value for $\mathcal{N}_{Y_F}^{T^k}$, $\mathcal{N}_{Y_F}^{P}$, and $\mathcal{N}_{Y_F}^{N}$. These features provide a way to measure the likelihood a frame and its neighborhood are misclassified, since disagreement between RM and LM is a sign of misclassification.

3. Whether Y_F is predicted to be relevant by RM. This feature captures whether there is any bias of the model toward one class. This feature is expected to be especially useful for domains where there is a class skew in the data.

4. Percentage of $\mathcal{N}_{Y_F}^{P}$ and $\mathcal{N}_{Y_F}^{T^k}$ that agree with the label Y_F. $\mathcal{N}_{Y_F}^{P}$ and $\mathcal{N}_{Y_F}^{T^k}$ both have positive correlation with Y_F. A lower percentage value indicates higher likelihood

of misclassification.

2.5.1.2 Training MLI

Because MLI predicts whether a frame is misclassified, it cannot use the labels in the training data directly. Rather, it needs to be trained on data that specifies the features described above for each frame and whether the frame is misclassified. To construct this training data, we split the original training data into k folds. We train RM and LM on $k-1$ folds and test them on the k^{th} fold. For each training frame F for MLI, we construct the features described above using these RM and LM. The label of F for MLI is *misclassified* if RM (trained on the $k-1$ folds) predicts Y_F incorrectly and *not-misclassified* otherwise. We repeat this process for each fold.

2.6 Experimental Evaluation

We performed video retrieval on the Videoweb dataset from UC Riverside [22], where various people are recorded for short periods of time, called the *scenes*. Our video retrieval task is: given training data for a number of people in a number of scenes, retrieve the frames for a new query person (whose image is given) in a new scene. We train our probabilistic models, LM and RM, on the training data, and perform active inference on the test data, where a human annotator can provide the labels of a small number of frames, and the probabilistic models are expected to utilize the annotated frames to perform better on the remaining frames. We next describe the dataset, constructing the local features from the query image and video frames, our evaluation strategy, and experimental results in

detail.

2.6.1 Dataset

The dataset contains a number of scenes recorded over four days. Each scene is recorded by a camera network and the videos from different cameras in the network are approximately synchronized and contain several types of activities over a number of people.

We arbitrarily choose scenes 20 to 25 for our experiments. In these scenes, seven cameras overlook the playground. Scene 21 does not include data from one of the seven cameras, so we use it to generate queries. All other scenes are used for retrieval. The time period for scene 24 is approximately twice as long as those of other scenes. We split it into two parts with equal time periods, and refer to them as scene 24.1 and 24.2. This gives us six scenes of approximately equal length. Each camera has about half an hour of video over all scenes, and we use a frame rate of one frame per second. Figure 2.1 shows the camera layout and example frames from these seven cameras.

2.6.2 Queries

We use a set of query images from four persons. These images are from scene 21, which is not included in the scenes used for retrieval. We consider three query images for each person from the front, back, and side view. Since people in the dataset can easily be occluded and are mainly characterized by the patterns of their clothes, we manually crop each query image to highlight their distinctive clothing regions. These cropped images

Figure 2.2: The 12 query images from 4 people. The parts inside the red bounding boxes are the cropped portions used to compute similarity measure.

are used as queries. Figure 2.2 shows the query images and their cropped results.

2.6.3 Similarity Features

Both LM and RM need the local feature vector $\vec{X}_F$ for each frame, query and scene. We adopt a commonly used, bag-of-words [86] approach to derive a feature that measures the similarity between the query and regions of interest in each frame. This involves computing image descriptors at keypoints in a region of interest. These descriptors are quantized into codewords, which are created by applying k-means clustering to training examples. Then histograms of the codewords in two regions of interest are compared using cosine similarity. In our implementation, the entire query is one region of interest, while we use the background subtraction algorithm of Zivkovic [113], which is based on a standard method using Gaussian mixture model [88], to determine regions of interest in the video. We densely sample keypoints in the region of interest, and build descriptors using a color histogram over RGB space. For each video frame, descriptors from all detected regions of interest are considered as a whole to represent the frame. In preliminary experiments, the color histogram is more effective than some other descriptors, such as

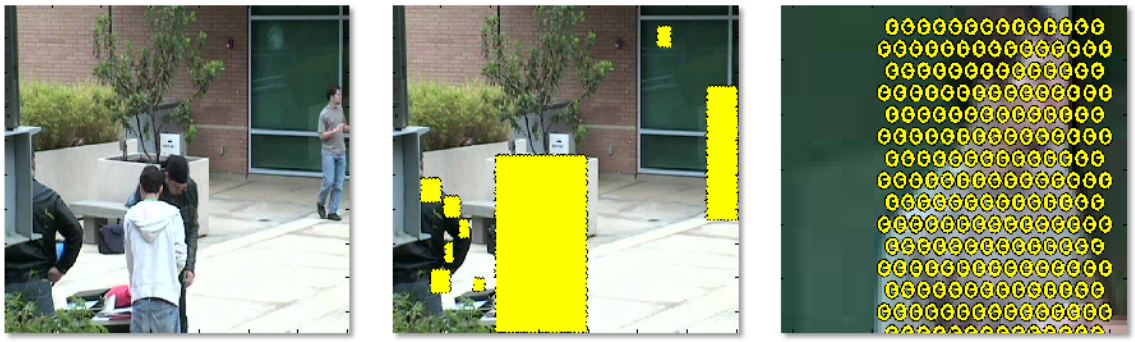


Figure 2.3: An example showing densely sampled points over regions of interest computed by background subtraction. The right-most figure is the enlarged view of the left-most region where key points are densely sampled. The detected region is of square shape, because we run morphological operations after background subtraction and extract non-overlapping bounding boxes over connected components. In addition, the reason that the person with a black coat is only partially sampled is because he has been present over a long period of time with little motion.

SIFT [56] and OpponentSIFT [94]. Figure 2.3 shows an example of densely sampled key points from video frames. Using k-means clustering on a random subset of descriptors, we form 500 visual words. By comparing histograms, we obtain features that encode the similarity between a query and video frames.

2.6.4 Training `LM`, `RM`, and `MLI`

When testing for a particular query in a given scene, the neighborhood structure of the camera network, the probabilistic models `LM` and `RM`, and the active inference technique `MLI` are learned on data from other persons and other scenes. For computing the temporal neighborhood, $\mathcal{N}_{Y_F}^{T^k}$, we set $k = 1$ for `RM`, and $k = 4$ for `MLI`. We have three queries for each person, and they all share the same structure, probabilistic models, and

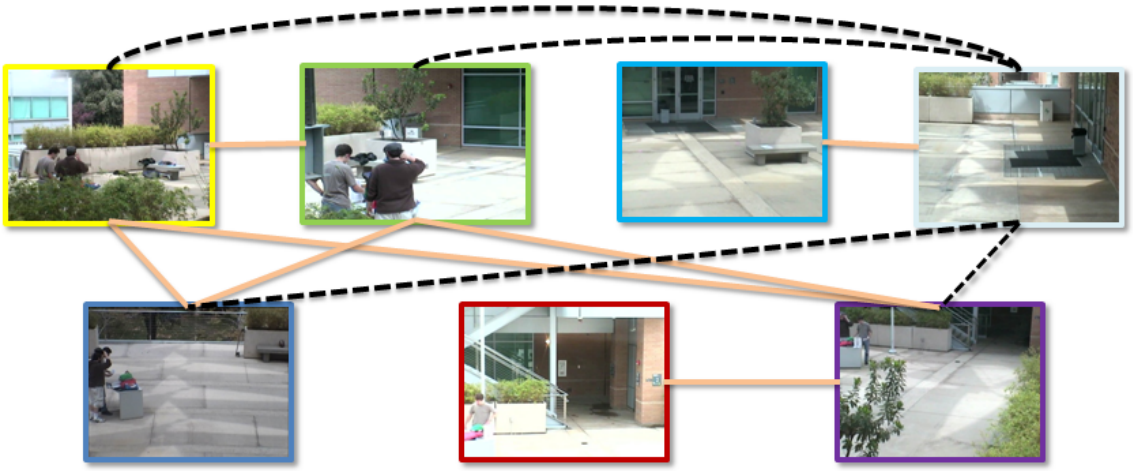


Figure 2.4: An example of learned topology. Light gold edges with solid lines denote positive correlation and black edges with dashes denote negative correlation. Temporal edges are not shown because they are fixed.

MLI. The threshold σ_p for learning positive-spatial neighbors is 0.6 and σ_n for negative-spatial neighbors is -0.15. We use logistic regression in WEKA with default parameters [37] to learn LM, RM, and MLI. We generated the ground truth for each person by manually labeling the frames. Figure 2.4 shows an example of the learned network structure.

2.6.5 Non-incremental and Incremental Sampling

The sampling locations for RND and UNI do not depend on the output of any probabilistic models. Thus, for them sampling is carried out independently, in a non-incremental fashion, and the locations sampled for a smaller budget are not a subset of those sampled for a larger budget. On the other hand, sampling for MR, UNC, and MLI is dependent on the output of probabilistic models. Because RM inference is based on the acquired labels, the labels acquired at lower budget levels can change the predictions

of RM. Thus, for these active inference techniques we perform incremental sampling, first acquiring the labels of a small subset of k frames, then incorporating these acquired labels into the prediction, and running the acquisition algorithm again to sample the next set of k frames. We do this until the budget is used up. In our experiments we used $k = 10$.

2.6.6 Evaluation Methods

We perform active inference for both LM and RM with a budget (the number of frames for which the human annotator provides the labels during inference) varying from 0% to 50% of all frames. For UNC−RM, MLI, and MR−RM we repeatedly perform inference to update the predictions whenever ten new labels have been acquired. In these methods, the use of inference can allow the results of partial labeling to guide the system in determining locations for additional labeling.

Given that we have six scenes, four people, and three queries per person, we train on five scenes with nine queries, and test on a scene for three queries, and we repeat this process for each scene and each person, giving us 72 different test cases. We trim the scenes so that each one is 270 seconds (4.5 minutes).

For each active inference technique, we plot two performance measures on the Y axis as a function of the budget on the X axis. The first performance measure is accuracy, measuring the percentage of frames predicted correctly. The second measure is the 11-point average precision [63] of the precision-recall (PR) curves over all frames. For those frames whose labels are acquired, we set their probabilities to either 0 or 1 based on their actual labels. However, in three out of 72 cases, the queries are completely absent from

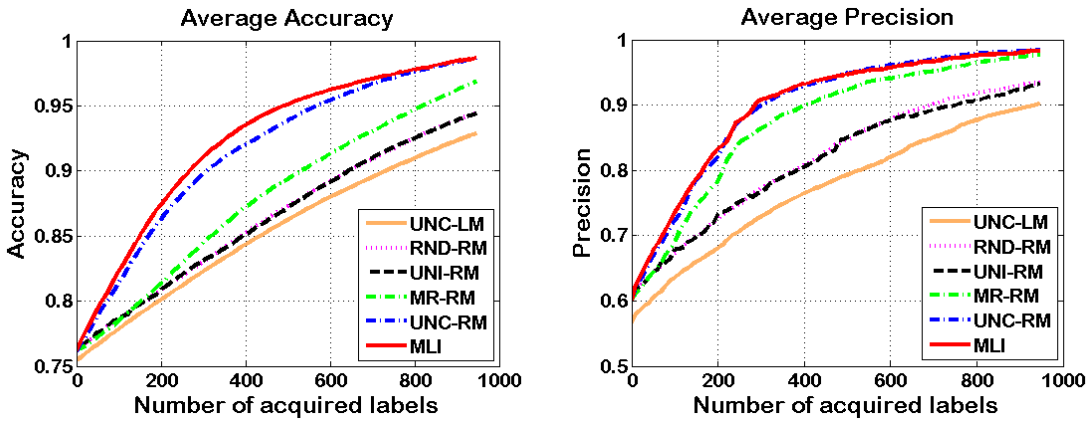


Figure 2.5: Left: Average accuracy as budget increases. **Right**: Average precision as budget increases.

the scene and the PR curve is undefined for these three cases. We ignore these three scenes for calculating the 11-pt precision measure. Significance claims are based on a paired t-test at the 0.05 level.

2.6.7 Results and Discussion

We compare the performance of four active inference methods described in Section 2.5 using `LM` and `RM`, while considering `MLI` only with `RM`. For `MLI`, we use temporal neighbors within four time steps for constructing the features that are based on temporal neighbors. The left side of Figure 2.5 shows performance using average accuracy, while the right side shows 11-pt average precision. For `LM`, `UNC` has the best performance when compared with `RND`, `UNI`, and `MR`. Therefore, we show results for only `UNC` for `LM` in order to increase readability in the graphs. For `RM`, however, we show the results for all active inference techniques, as they are better than `UNC` using `LM`.

Based on a statistical analysis of the results, we draw the following conclusions.

First, whenever we apply the same algorithm using LM and RM, RM performs significantly better. Comparing UNC−LM and UNC−RM in Figure 2.5 provides a typical example of the large magnitude of this difference. Second, we find that UNC−RM and MLI always perform significantly better than all other methods. Third, MLI has a statistically significant advantage over UNC−RM in terms of accuracy up to 32% budget (600 labels), and the two methods are comparable afterwards. When we measure 11-pt average precision, MLI is significantly better than UNC−RM in a few spots, and never significantly worse. Based on this result, we conclude that the use of graphical models and collective classification provides large improvements in performance for active inference. In addition, MLI, our adaptation of RAC, provides the best performance, especially at low budget levels, which are more likely to be used in practice.

2.7 Summary

Our work addresses the problem of using active inference to direct human attention in searching a camera network for people that match a query image. We first use local information to measure the similarity between the query and each frame. We find that by representing the camera network using a graphical model, we can more accurately determine whether video frames match the query, and improve our ability to direct human attention. We experiment with five methods of determining which frames should be labeled. We find that the value of the graphical model is very strong, regardless of which algorithm is used to select frames for human labeling. In comparing these active inference methods, we find that there is an advantage in labeling those frames that are

most likely to contain errors. This can be captured by a simple method that measures the entropy of the probability distribution that indicates our uncertainty about the labels of each frame. However, we find that we do somewhat better by adapting an approach that uses a classifier to predict which frames are in error. Overall, we demonstrate that we can adapt tools developed for active inference in graphical models to improve the capacity of humans to effectively search or annotate video from camera networks.

Chapter 3

Dynamic Processing Allocation in Video

The work from this chapter was published in the IEEE Transactions on Pattern Analysis and Machine Intelligence in November 2011, [16].

3.1 Introduction

In this chapter, we explore methods to direct computational resources to analyze videos. Instead of analyzing multiple videos from a camera network, we consider video from a single camera. We develop a new method for controlling processing, so that available resources are directed to the most relevant portions of the video. In our proposed approach, we initially perform some inexpensive processing of a video by applying a cheap but less accurate algorithm combined with a sparse application of a more expensive and accurate algorithm. We then use an inference algorithm to determine which frames we should apply further expensive processing.

Our work makes two critical assumptions. First, that expensive algorithms exist that can perform a task quite accurately (e.g., $>$90% accuracy). While many real-world vision tasks are still too challenging for this, recent growth in the number of vision companies and applications illustrate that high accuracy is often achievable in simpler tasks (eg., face detection in cameras [45]) or in controlled environments (eg., detection and tracking in stores [72]). Moreover, in vision systems with a human in the loop, a human analyst may

be regarded as a very accurate and very expensive algorithm. Second, we assume that a much cheaper, but less accurate algorithm is available, and that it is desirable to use the output of this algorithm to direct the attention of the expensive algorithm most profitably. We want to stress that our work does not aim to solve vision problems that are beyond the reach of existing algorithms, but rather to speed up the solution to problems that are currently solvable, albeit only at considerable cost.

To combine information from features produced by cheap and expensive algorithms, we present a graphical model for video analysis. We use a second-order Markov model with a node for each frame, and a state variable that indicates whether this frame is relevant to a query. For example, the state might indicate whether the frame contains a visible face. Each state has two potential observations. The first observation is always given; it is obtained by running a cheap algorithm on all frames. For example, cheap background subtraction might provide a clue as to whether people are currently visible. The second observations is only obtained if a more expensive and accurate algorithm is applied to that frame (in this example, a face detector). As in a Hidden Markov Model (HMM), each observation directly depends on the current state. In addition, in our model each observation directly depends on the previous observation. This captures the phenomenon that errors made by an algorithm are often correlated from one state to the next. This model allows us to effectively combine information from cheap and expensive algorithms to improve performance.

Our primary contribution is a new algorithm that uses this model to determine where in a video to apply the expensive algorithm. We build on prior work by Krause and Guestrin [48] that shows that one can use a dynamic programming algorithm to determine

the optimal places at which to make observations in a first-order Markov chain. While this work is readily extended to our graphical model, it requires $\Theta(B^2n^3)$ computation time, where n is the number of nodes in the Markov chain, and B is the number of places at which we will apply the expensive algorithm. In our setting n is the number of frames in the video and B is also $\mathrm{O}(n)$, so this algorithm is not practical for video analysis.

We solve this problem with a new algorithm that produces an approximately optimal answer efficiently. More precisely, we make an additional assumption about the concavity of the reward from observations as the budget increases, a law of diminishing returns that we show is generally valid in our setting. Then, we show that by applying part of the total budget to make observations at a uniform step size, we can find an allocation of the remaining observations that will be at least as good as the optimal batch allocation, and that requires a modest amount of computation. This allocation makes use of rewards computed by Krause and Guestrin's algorithm, applied to small sections of the video.

Our approach is quite general, and can be applied to a wide range of scenarios in which multiple algorithms are combined into a single system. Our final contribution is to experimentally demonstrate the value of this algorithm in two very different vision tasks: motion and face detection. To detect motion efficiently we combine a very cheap and a more expensive background subtraction algorithm. For the second task, we use background subtraction to trigger face detection. We show that our algorithm can be used to significantly improve performance.

The chapter is organized as follows. In Section 3.2 we discuss related work. In particular, we describe a dynamic programming algorithm [48] that determines the optimal place to make observations. We then describe our new algorithm in the context of

Markov Chains in Section 3.3. Then, we introduce our graphical model for video analysis and describe how to apply the new algorithm to this model in Section 3.4. In Section 3.5, we show experiments.

3.2 Prior Work

We first review background subtraction and face detection algorithms. Next, we describe work on visual computing that deals with issues of resource constraint. We then discuss work on feature and label acquisition. Finally, we describe the algorithm by Krause and Guestrin [48] in detail.

3.2.1 Background Subtraction

Background subtraction detects moving objects in video, usually taken by static cameras. This typically involves building a background model. Wren et al. [104] use a Gaussian estimate of the background distribution of each pixel. Lo and Velastin [55] use the median of the previous n frames as the background model. Elgammal et al. [27] propose a non-parametric model based on kernel density estimation to approximate the pdf of each pixel. In Kim et al. [46], background values are quantized into codebooks to handle periodic motion. Rittscher et al. [77] represent the background using a hidden Markov model, which can discriminate between foreground, background, and shadow. Cheung et al. [19], Piccardi [71], and Yilmaz [109] give a general review of this problem. We describe in more detail two methods we use in our experiments.

In frame differencing (FD), the background model of the frame at time t, f_t, is

the frame in the previous time step, f_{t-1} (Jain and Nagel [41]). Given a threshold, Th, $|f_t - f_{t-1}| > Th$ gives the foreground region of f_t.

Stauffer and Grimson [88] use a mixture of Gaussians (MoG) for the background model. With the assumption that a more compact distribution with a higher mode is more likely to be the background, MoG selects background components whose ratio between its peak value and standard deviation is greater than a certain threshold. Finally, it uses recent pixel values to update the model parameters. This method is much more sophisticated than the FD method, and requires significantly more computational resources. Zivkovic [113] improves this work by using recursive equations that can also simultaneously select the appropriate number of components in the mixture model. We call this method the improved adaptive Gaussian mixture model (IAGMM), and we use it in our experiments.

3.2.2 Face Detection

Yang et al. [107] provide a comprehensive survey of face detection methods and organize them into four major categories. First, top-down knowledge based methods represent a face using human knowledge, which usually captures relationships between facial features such as eyes, nose, and mouth. Yang and Huang [106], for example, use a hierarchical knowledge-based method to detect faces. Second, bottom-up feature based methods seek invariant features, such as eyebrows, hair texture, and skin color, for detection. Hsu et al. [39] propose a skin-tone color model which they use to generate face candidates for verification by facial features (see also Jones and Rehg [44]). The third category is template based methods, in which correlation between an input image and the

template is used to detect faces. Sinha [85], for example, builds a face template by capturing the invariance between the relative brightness of facial regions. The last category includes appearance based methods, which in general use statistical analysis and machine learning techniques. Various methods, such as support vector machines [69] and neural networks [79], and naive Bayesian classifiers [82] have been proposed.

Viola and Jones [99] achieve a breakthrough in performance that has been widely adopted. With integral images for fast computation, their scheme uses a set of features that are similar to Haar wavelets. They then construct classifiers by selecting a small number of important features using Adaboost. Finally, the scheme detects faces inside an image region by applying classifiers in a cascade. At each level of the cascade, one uses a classifier with a very low false negative rate, although the false positive rate might be high. Subsequent classifiers are run only when previous classifiers indicate a positive result. Lienhart and Maydt [52] extend this work by adding an efficient set of 45° rotated features to the original feature set and by using a new post-optimization procedure for a given boosted classifier. Their work shows significantly lower false alarm rates, and we use this method to detect faces in our experiment.

There have been many other extensions to the Viola-Jones method. For example, Huang et al. [40] extend the cascade of classifiers structure to a Width-First-Search (WFS) tree structure. Mita et al. [64] introduce a new feature, called the joint Haar-like feature, for detection. Xiao et al. [105] use a boosting chain to integrate historical knowledge of successive learning of strong classifiers.

3.2.3 Resource Constraints in Computer Vision

Many methods have been developed to handle resource constraints in computer vision. Weiss and Taskar [103] generalize the approach of Viola and Jones and apply it to a range of applications, including handwriting recognition. Felzenszwalb et al. [30] develop cascades for object detection using deformable models such as pictorial structures. Vijayanarasimhan [98] recently introduce a novel framework for object detection and classification in still images under resource constraints. They design a grid based model that is used to determine the best image regions to look at and the best features to be extracted. This process is guided by the principle of value-of-information (VOI) to find the most evidence at the least cost.

In video processing, performance is often an issue, as many effective algorithms are too slow to run in real-time, and even fast algorithms may require enormous amounts of time when used to perform retrospective analysis of large quantities of video. One common strategy is to run cheap, lower level algorithms such as motion detectors to determine when something interesting might be happening. When these detect motion, higher level algorithms are then deployed. While this approach is used heuristically, but very effectively in a wide range of applications, we will mention two representative works that formalizes this. First, Krishna et al. [49] propose an algorithm switching approach to handle background subtraction. The system starts by processing each frame with a unimodal model. When the system shows poor segmentation quality, it switches to use the MoG model. Second, Barotti et al. [4] use algorithm switching to handle lighting changes and solve bootstrapping problems in motion detection. When the system detects sudden

global illumination variation, the motion detection switches from background subtraction using a single Gaussian to FD.

3.2.4 Feature and Label Acquisition

The machine learning community has looked at the problem of determining which features to acquire in order to correctly classify instances. In this setup, instances are described by a set of features each of which has an associated acquisition cost, and a total budget limits feature acquisition. Some example strategies are [5, 93]. The biggest difference between this line of work and ours is that the feature-value acquisition community treats each instance as independent. However, in our case, the information we want to extract in nearby frames of video is highly correlated, and we should be able to do better if we take these correlations into account.

Another related area of work is label acquisition: instead of obtaining features, we can query an oracle to determine an instance's label directly. Given a network of instances, such as a sequence of frames, a network of friends, etc, acquiring the label for an instance helps in correctly classifying the rest of the network. The question is then which instances should be queried in order to get the best performance on the remaining ones. Rattigan et al. [76] queries the instances that are structurally important, e.g. highly connected instances, central instances, etc. Bilgic and Getoor [8] build a classifier that can predict which instances might be misclassified and query a central instance only if it is predicted as misclassified. Active learning work [84] is very related to this problem, and it has been applied to visual recognition [96, 97]. However, active learning acquires

labels to construct training data to learn a model, whereas label acquisition described here is applied to an already learned model to guide probabilistic inference.

These methods have been quite successful in practice, but they are heuristic approaches and have no theoretical guarantees (partly because they are applicable to general networks). However, for the class of chain graphical models such as HMMs, Krause and Guestrin [48] show how to solve the label acquisition problem optimally. We describe their work next.

3.2.5 Optimal Observation Plans

Krause and Guestrin [48] present VoIDP, an Dynamic Programming to optimize Value of Information, for selecting observations for the class of chain graphical models. Since we build on this method, we now describe it in some detail, although we consider a special case of their work that is suitable to our problem.

They optimize an objective function based on a class of reward functions, R, that are defined using the probability distribution of a set of random variables $S = \{X_1, \cdots, X_n\}$. This set of variables forms a chain graphical model, that is, $i < j < k$, implies that X_i is conditionally independent of X_k given X_j. For example, consider a HMM unrolled for n time slices. Then the n hidden state variables form a chain graphical model. Suppose that for each of these variables, it is possible to observe its hidden state at a fixed cost. This corresponds, in our problem, to the supposition that an expensive algorithm is extremely accurate, and reveals the hidden state. Let O be the set of observed variables and o be the values of these variables. $O = o$ is used to denote each variable in O takes its

corresponding value in o.

The reward function R is built upon a local reward R_j, which is a functional on the probability distribution $P(X_j|O = o)$. While this reward could be quite general, in this chapter we consider only X_j that are binary variables, and use

$$\begin{aligned} &R_j(P(X_j|O = o)) = \\ &\quad max(P(X_j = 1|O = o), P(X_j = 0|O = o)) - \\ &\quad min(P(X_j = 1|O = o), P(X_j = 0|O = o)) \end{aligned} \tag{3.1}$$

That is, given a set of observations, we receive a greater reward as we become more certain of the value of each state. This reward is equivalent to considering the expected number of correct classifications. We will also use the notation:

$$R_j(X_j|O) \triangleq \sum_o P(O = o)R_j(P(X_j|O = o)), \tag{3.2}$$

That is, given the choice of a set of variables, O, to observe, $R_j(X_j|O)$ denotes the expected reward we will receive from these observations.

Assume that there is a fixed budget B for selecting observations, we then must select observations O to

$$\begin{aligned} &\text{maximize } J(O) = R(O) = \sum_j R_j(X_j|O), \\ &\text{subject to } \|O\| = b \leq B, \end{aligned} \tag{3.3}$$

where j is the index over the state variables S, b is the number of observed state variables O, and B is the total budget for the whole chain. Observations can include variables at any time step in the chain since we consider processing a video after it is recorded. This corresponds to the "smoothing" version of the problem [48].

The conditional independence property in the chain graphical model simplifies the local reward. With this property, the local reward $R(X_j|O) = 1$ in the case that $X_j \in O$. In the case that $X_j \notin O$, we have $R(X_j|O) = R(X_j|O_j)$, where O_j is a subset of O containing two observations. These are the last observation preceding X_j and the first observation in O that follows X_j.

Furthermore, Krause and Guestrin [48] consider both a conditional planning setting of this problem, in which the best observation is made and then the optimal next observation is computed, and this is repeated k times, and a subset selection setting of the problem, in which one decides on the locations of the best observations with a total cost of k first, and then makes these observations. Our algorithm uses the conditional planning variant since it in general produces the best performance.

They solve this problem by noting that once an observation is made, it splits the problem of determining future observations into conditionally independent components before and after the observations. This allows for a dynamic programming solution. They define a value, $J_{a:b}(x_a, x_b; k)$, which denotes the reward produced by the optimal plan with a budget of k over the interval from variables X_a to X_b, given that these variables have been observed to have states x_a and x_b. Then they note that $J_{a:b}(x_a, x_b; k)$ can be recursively computed given the value of $J_{c:d}(x_c, x_d; l)$ for all $a \leq c \leq d \leq b$ and $l < k$. The recursive formula is

$$\begin{aligned} J_{a:b}(x_a, x_b; k) = \max\{&J_{a:b}(x_a, x_b; 0), \max_{a<j<b}\{ \\ &\sum_{x_j} P(X_j = x_j | X_a = x_a, X_b = x_b)\{R_j(X_j|X_j = x_j) + \\ &\max_{0 \leq l \leq k-1}[J_{a:j}(x_a, x_j; l) + J_{j:b}(x_j, x_b; k - l - 1)]\}\}\}, \end{aligned} \tag{3.4}$$

where the base case is

$$J_{a:b}(x_a, x_b; 0) = \sum_{j=a+1}^{b-1} R_j(X_j | X_a = x_a, X_b = x_b). \tag{3.5}$$

The recursive formula basically iterates over each split point j between a and b to find one that returns the highest reward (or performs no further observations if they do not increase the reward). For each split point, the reward is the expectation taken over all possible assignments of value to the split point. All possible budget allocations between the two split subsequences are considered when the value of split point is fixed.

To initialize, the algorithm adds two independent dummy variables, X_0 and X_{n+1}, which have no reward and observation cost but have default states, to the head and tail positions of the chain. Thus the optimal reward for a chain of n variables with a budget of B is computed as $J_{0:n+1}(x_0, x_{n+1}; B)$.

This algorithm corresponds to the smoothing version of VoIDP in a conditional plan setting, and we refer it as VoIDP-SCP in this chapter. According to Theorem 2 in [48], the complexity of this algorithm in terms of number of evaluations of local rewards for our binary state variables is

$$\Theta(B^2 n^3). \tag{3.6}$$

3.3 Efficient Observation Plans

We now present a novel algorithm, Dynamic Progamming Allocation (DPA), that is efficient enough to apply to problems with very large values of n. DPA approximates the optimal algorithm and is much faster. In the next sections, we will show how this algorithm can be applied to video processing. DPA first uses B' observations from the total

budget, B, to make uniform observations. This splits the Markov chain into $M = B' + 1$ consecutive intervals where the first and last variables of each interval are observed. This breaks our problem up into a series of smaller problems of the same size. These problems are not independent, however, since the remaining $B'' = B - B'$ budget must be parceled out between all these intervals. We show that, with an additional, reasonable assumption, this can be done optimally. Once the budget is allocated to intervals, observations can be allocated within intervals using VoIDP-SCP.

DPA maximizes the sum of rewards over all intervals to allocate budget between them. That is, let k_i be the budget allocated to interval i. Let $J_i(k_i)$ be the reward of VoIDP-SCP for interval i with a budget of k_i. We want to find $k_1, k_2, \cdots, k_M$ such that they

$$\text{maximize } \sum_{i=1}^{M} J_i(k_i), \text{ subject to } \sum_{i=1}^{M} k_i = B''. \tag{3.7}$$

To perform this optimization efficiently, we rely on the empirical observation that for each interval, the optimal reward typically forms a concave curve as the budget increases. That is, the plot of $J_i(k_i)$ against k_i as k_i increases is concave in general. This is a kind of law of diminishing returns property. We denote these kinds of curves as Reward-Budget (RB) curves, and the assumption that these curves are concave will allow us to optimally allocate our budget. We will experimentally verify that this assumption is reasonable. Given this assumption we can compute the k_i with our proposed algorithm, Dynamic Progamming Allocation (DPA).

It remains to describe how we perform the second step of this algorithm. Let N_i be the maximum possible budget for interval i, typically the number of unobserved state

Algorithm 1 Dynamic Processing Allocation (DPA)

1: Use B' observations to make uniform observations to break the chain model into consecutive disjoint intervals separated by the observed variables;

2: Allocate the remaining budget B'' to these intervals. No observation is taken in this step;

3: Use VoIDP-SCP to determine the observation locations within each interval; observations are taken in a conditional mode.

variables in the interval. We first compute the reward curve for each subsection up to its maximum budget. That is, we compute $J_i(k)$, for $1 \leq k \leq N_i$, using VoIDP-SCP. Next, we define the reward increment as

$$\Delta J_i(k) \equiv J_i(k) - J_i(k-1), \tag{3.8}$$

where $k = 1, \cdots, N_i$. Assuming concavity, we have

$$\Delta J_i(k) \leq \Delta J_i(k-1) \tag{3.9}$$

for all ks. Next, we sort all the reward increments in descending order. Finally, the budget for each interval is set to the number of increments it has in the top B'' positions of the sorted list. We call this allocation method batch budget allocation. Intuitively, we can see that this always assigns observations to the sequences where they will create the most incremental benefit. We prove this in Section 3.3.2 and discuss the complexity of the algorithm in Section 3.3.3.

We expect this algorithm to do much better than an optimal batch algorithm, since the B' observations we use to break the problem into intervals also provide very useful information, and because we can use an optimal conditional plan within each interval,

Algorithm 2 Incremental Budget Allocation - Step 2 of Algorithm 1

1: Initialize the budget of each interval to be zero;

2: Compute the reward increment $\Delta J_i(1)$ for $i = 1, 2, \cdots, M$;

3: Select the highest increment, and add one to the budget of the corresponding interval I;

4: If the total budget, B'', has been used, terminate and use the current budget allocation for each interval as the final budget allocation;

5: If not, compute the next reward increment for interval I, and use it to replace the current reward increment for this interval. Go back to step 4.

which can be much better than the optimal batch plan.

However, we note that it is possible to improve the running time of budget allocation at the cost of some additional memory. This is because DPA requires us to compute $J_i(k)$ for all k, while in practice, most intervals are allocated small budgets, and we only need to compute the RB curve up to this budget. This leads to the algorithm incremental budget allocation, which performs step 2 of DPA a bit differently. This method returns the same output as the batch allocation method. In Section 3.3.3, we show that it is asymptotically faster, especially when we have a small budget. In addition, when the subsection size is large, we can avoid computing the RB curve up to its maximum budget for each subsection, this will save a significant amount of processing time. We use DPA with incremental budget allocation in our experiments.

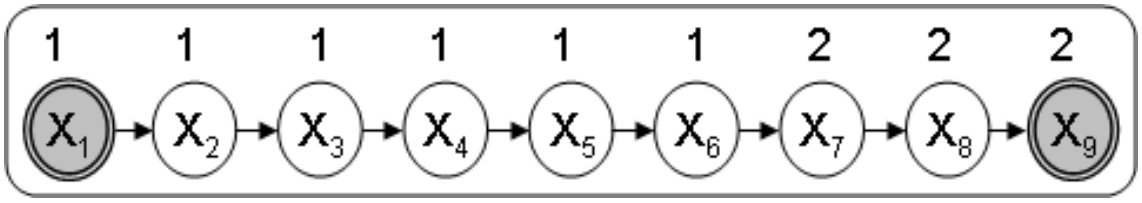
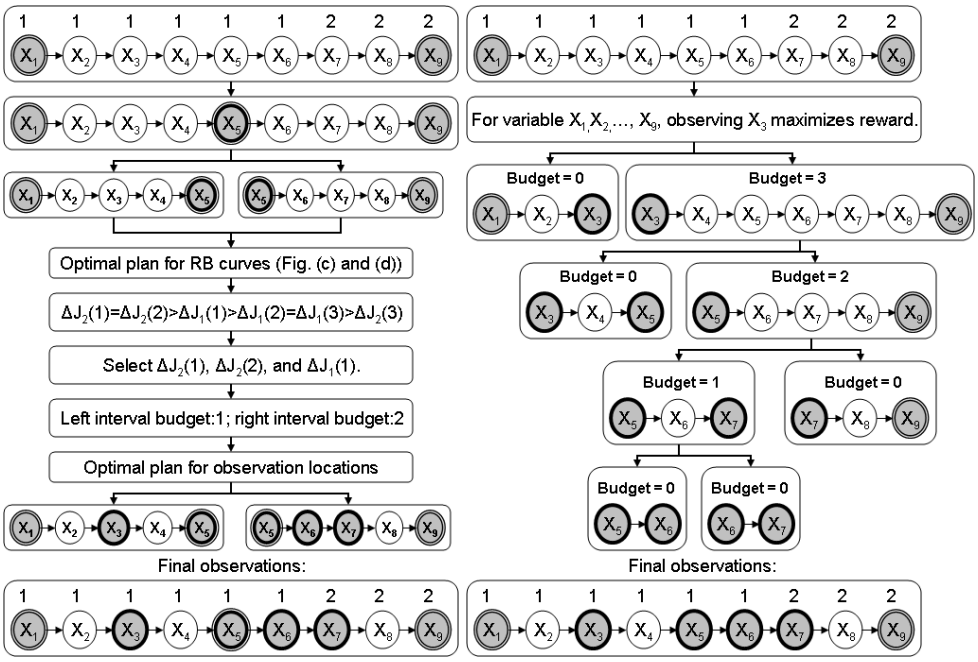


Figure 3.1: The example chain graphical model. We assume that X_1 and X_9 are observed in advance, shown with gray shading. The state of a node can be 1 or 2, and we show the actual state on the top of each node.

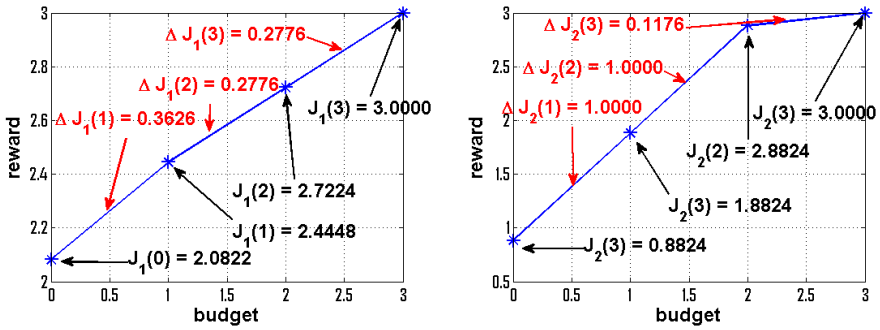
3.3.1 An Example

We use an example to illustrate how DPA works and how VoIDP-SCP is different. Consider a chain graphical model with nine state variables whose value can be either 1 or 2. The prior probability of being in each state is 0.5. The transition probability of switching from one state to the other is 0.2. Figure 3.1 shows this model and displays one set of states generated by it on the top of each variable. For simplicity, we assume the first and last states are known in advance. Fig. 3.2 shows how DPA and VoIDP-SCP determine the observation locations with a budget of 4.

First, we note that for this example, it is most likely that the initial states have a value of 1, and that at some point in the chain there is a single transition from 1 to 2. To correctly determine the state values, the main goal is to find the location of this transition. It is also possible that there are really three transitions, and a secondary goal will be to check on that. Next, we note that VoIDP-SCP is able to perform a binary search to find such transitions. It turns out that the optimal strategy for this situation involves first determining the value of X_3. If this state is 1, then the remaining budget is sufficient to allow a binary search to be performed on states X_4-X_8, to find the transition from 1 to



(a) DPA-batch (b) VoIDP-SCP



(c) RB curve - interval 1 (left) (d) RB curve - interval 2 (right)

Figure 3.2: (a) shows how DPA with batch budget allocation determines the observation locations. We use gray to highlight observed nodes. The interval size is 5, so the initial observation is X_5, which is highlighted by a double-line boundary with a thick inner line. (c) and (d) show the RB curves for left and right intervals respectively. The observation locations for each of these two intervals are determined by VoIDP-SCP, and are highlighted by a single-line thick boundary. (b) shows how VoIDP-SCP determines observations locations, which are also highlighted by a single-line thick boundary. It shows how the chain is split into sub-chains by sequential observations.

2. This strategy therefore guarantees that the transition from 1 to 2 will be found, and maximizes the chances that any additional transitions will be found.

Suppose instead we run DPA, with a budget of 4, and $B' = 1$. The algorithm begins by determining the state of X_5, to break the problem into two equal parts. When this state is found to be 1, the algorithm then allocates its budget between these two subsequences. To allocate the remaining budget to each interval, it computes the RB curves up to its maximum budget for both intervals, as shown in Fig. 3.2(c) and 3.2(d). Notice that both curves satisfy the concavity property. In addition, the reward increment under a small budget for the right interval is higher than those for the left interval. This is because the state of the first and last nodes for the right interval indicate a state transition. As a result, the right interval obtains a higher budget. In fact, DPA allocates two observations to the right side of the chain, which is enough to perform a binary search for the state transition, and one observation to the left side. This final observation on the left side is more likely to find something interesting than if allocated to the right side, once the binary search has occurred.

We now consider how incremental budget allocation works in this example. After the observation of X_5, the chain is split into two intervals as shown in Fig. 3.2(a) and the remaining budget becomes 3. The incremental algorithm then initializes the budget for both intervals to be 0 and computes $J_1(0), J_1(1), J_2(0)$, and $J_2(1)$ for $\Delta J_1(1)$ and $\Delta J_2(1)$. Since $\Delta J_1(1) = 0.3626 < \Delta J_2(1) = 1.0000$, it increases the budget for the right interval by 1. The remaining budget becomes 2 and it computes $J_2(2)$ for $\Delta J_2(2)$. It again increases the budget for the right interval by 1 because $\Delta J_1(1) < \Delta J_2(2) = 1.0000$. The remaining budget becomes 1 and it computes $J_2(3)$ for $\Delta J_2(3)$. After this,

because $\Delta J_1(1) > \Delta J_2(3) = 0.1176$, it increases the budget for the left interval by 1. All the budget has been allocated, and the algorithm terminates with a budget of 1 for the left interval and 2 for the right interval. Notice that compared with batch allocation, incremental allocation does not compute $J_1(2)$ and $J_1(3)$.

DPA is not optimal in two ways. First, an optimal set of observations may not include X_5. Second, allocating one observation to the left subsequence and two to the right subsequence may not be optimal; future observations could determine that a different allocation would be better. On the other hand, in VoIDP-SCP shown in Fig. 3.2(b), the initial observation X_3 is determined after computation and comparison of the expected reward of observing $X_1, \cdots, X_9$ with all possible budget distributions. This is a considerable amount of computation. In DPA, this interval is split into two shorter intervals, and reward can be more cheaply computed for each subchain separately.

3.3.2 Proof of Correctness

We now provide a proof that the batch budget allocation and incremental budget allocation is correct based on the problem formulation in (3.7).

We use the following new notations, definitions, and facts. With a budget of B, we let $\hat{k}_1^B, \hat{k}_2^B, \cdots, \hat{k}_M^B$ be an optimal budget allocation, where M indicates the number of subsequences among which we must divide the budget. Let $\bar{k}_1^B, \bar{k}_2^B, \cdots, \bar{k}_M^B$ be the budget allocation by the algorithm. Since the algorithm picks only the top B reward increments from Z to distribute budget, it is clear that $\sum_{i=1}^{M} \bar{k}_i^B = B$. The following theorem proves the correctness of the allocation algorithm by showing that summation

of reward from each subsection under the batch budget budget allocation is equal to that under the optimal budget allocation.

Theorem 1.

$$\sum_{i=1}^{M} J_i(\bar{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B). \tag{3.10}$$

To prove Theorem 1, we introduce Lemma 1 and Lemma 2. Lemma 1 shows that the sum of all reward increments for each subsection using batch budget allocation is equal to the sum of the top B reward increments in Z. Lemma 2 proves the the sum of all reward increments for each subsection under the optimal allocation cannot be greater than that under batch allocation. Finally, we prove Theorem 1 by adding the reward increment to the reward with zero budget for each subsection to establish the equality.

Denote the sorted list of rewards gained by an additional observation as Z and the sum of the top B reward increments in Z as ΔL. In addition, we define $\Delta J_i(0) \equiv 0$ to handle the case that some interval has a budget of 0. Then we define $\Delta \hat{J}_i^B \equiv \sum_{j=0}^{\hat{k}_i^B} \Delta J_i(j)$, which means $\Delta \hat{J}_i^B$ is the sum of all reward increments for interval i with a budget of $\hat{k}_i^B$. Similarly, we define $\Delta \bar{J}_i^B \equiv \sum_{j=0}^{\bar{k}_i^B} \Delta J_i(j)$.

Lemma 1.

$$\sum_{i=1}^{M} \Delta \bar{J}_i^B = \Delta L. \tag{3.11}$$

Proof. For an interval i, by the procedure of the algorithm, there must be $\bar{k}_i^B$ reward increments from interval i in ΔL. Let the sum of these increments be ΔL_i. In case that no such increment exists, we let $\Delta L_i \equiv 0$. Suppose $\Delta \bar{J}_i^B \neq \Delta L_i$. By concavity, we know that ΔL_i must include some reward increment $\Delta J_i(x)$ such that $x > \bar{k}_i^B$ and $\Delta J_i(x) < \Delta J_i(\bar{k}_i^B) \leq \Delta J_i(\bar{k}_i^B - 1) \leq \cdots \leq \Delta J_i(1)$. Thus, there must be at least

$\bar{k}_i^B + 1$ reward increments in the top B positions from interval i. But this conflicts with the fact that the top B positions only contain $\bar{k}_i^B$ such increments. Thus, it can only be that $\Delta\bar{J}_i^B = \Delta L_i$. Finally, because $\sum_{i=1}^M \bar{k}_i^B = B$, we have $\sum_{i=1}^M \Delta L_i = \Delta L$. Therefore, $\sum_{i=1}^M \Delta\bar{J}_i^B = \sum_{i=1}^M \Delta L_i = \Delta L$. □

Lemma 2.

$$\sum_{i=1}^{M} \Delta\bar{J}_i^B \geq \sum_{i=1}^{M} \Delta\hat{J}_i^B. \tag{3.12}$$

Proof. Any reward increment not in the top B positions of Z must be less than or equal to any reward increment in the top B positions because Z is sorted. So by Lemma 1, $\sum_{i=1}^M \Delta\bar{J}_i^B = \Delta L$ must be greater than or equal to the sum of any B reward increments from Z. Because $\sum_{i=1}^M \hat{k}_i^B = B$ and Z contains all reward increments from all intervals, we know that $\sum_{i=1}^M \Delta\hat{J}_i^B$ is also the sum of B reward increments from Z. Therefore, $\sum_{i=1}^M \Delta\bar{J}_i^B \geq \sum_{i=1}^M \Delta\hat{J}_i^B$. □

By the definition of $\Delta J_i(k)$, we know that $J_i(k) = \Delta J_i(k) + J_i(k-1)$. Using induction, it is trivial to show that $J_i(k) = \sum_{j=1}^k \Delta J_i(j) + J_i(0)$. Because we define $\Delta J_i(0) \equiv 0$, then

$$J_i(k) = \sum_{j=0}^{k} \Delta J_i(j) + J_i(0). \tag{3.13}$$

With this formula, we can finally prove Theorem 1.

Proof. By equation (3.13), we have

$$\begin{aligned}\sum_{i=1}^{M} J_i(\bar{k}_i^B) &= \sum_{i=1}^{M}[\sum_{j=0}^{\bar{k}_i^B} \Delta J_i(j) + J_i(0)] \\ &= \sum_{i=1}^{M}[\Delta\bar{J}_i^B + J_i(0)] = \sum_{i=1}^{M} \Delta\bar{J}_i^B + \sum_{i=1}^{M} J_i(0).\end{aligned} \tag{3.14}$$

Similarly,

$$\sum_{i=1}^{M} J_i(\hat{k}_i^B) = \sum_{i=1}^{M} \Delta\hat{J}_i^B + \sum_{i=1}^{M} J_i(0). \tag{3.15}$$

Then by lemma 2, it follows that $\sum_{i=1}^{M} J_i(\bar{k}_i^B) \geq \sum_{i=1}^{M} J_i(\hat{k}_i^B)$. Finally, because the budget allocation, $\hat{k}_1^B, \hat{k}_2^B, \cdots, \hat{k}_M^B$, is optimal, we have $\sum_{i=1}^{M} J_i(\bar{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B)$.

□

Now we show the correctness of Incremental Budget Allocation. Our main insight is that the sum of reward increments selected by algorithm 2 is equal to ΔL, the sum of the top B reward increments in list Z in algorithm 1. We will use the following similar notations, definitions, and facts. Let $\tilde{k}_1^B, \tilde{k}_2^B, \cdots, \tilde{k}_M^B$ be the budget allocation by algorithm 2, and we define $\Delta\tilde{J}_i^B \equiv \sum_{j=0}^{\tilde{k}_i^B} \Delta J_i(j)$. In addition, according the procedure of the algorithm, it is clear that step 4 is the place the select reward increments. Let the sum of all selected reward increments be ΔR. Furthermore, the algorithm selects $\tilde{k}_i^B$ reward increments from interval i and B increments in total. Thus, $\sum_{i=1}^{M} \tilde{k}_i^B = B$. To prove the correctness of algorithm 2, we only need to show that $\sum_{i=1}^{M} J_i(\tilde{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B)$. We now use Lemma 3, Lemma 4, and Theorem 2 to prove this.

Lemma 3.

$$\sum_{i=1}^{M} \Delta\tilde{J}_i^B = \Delta R. \tag{3.16}$$

Proof. Let ΔR_i be the sum of all selected reward increments from interval i by the algorithm. We now show that $\Delta R_i = \tilde{J}_i^B$. Suppose $\Delta R_i \neq \tilde{J}_i^B$. Then by the concavity property, ΔR_i must include some reward increment $\Delta J_i(x)$ such that $x > \tilde{k}_i^B$ and $\Delta J_i(x) < \Delta J_i(\tilde{k}_i^B)$. However, because the algorithm always selects the highest increment and compute the next increment for the corresponding interval before the total bud-

get is used up, it must also select all reward increments $\Delta J_i(y)$ with budget $y \leq x$ from interval i. And there are at least $\tilde{k}_i^B + 1$ of them. However, this conflicts with the fact that it only selects $\tilde{k}_i^B$ reward increments. Therefore, the assumption is false, and $\Delta R_i = \tilde{J}_i^B$. Finally, because $\Delta R = \sum_{i=1}^{M} \Delta R_i$, we have $\Delta R = \sum_{i=1}^{M} \Delta R_i = \sum_{i=1}^{M} \tilde{J}_i^B$. □

Lemma 4.

$$\sum_{i=1}^{M} \Delta \tilde{J}_i^B \geq \sum_{i=1}^{M} \Delta \hat{J}_i^B. \tag{3.17}$$

Proof. We first show that $\Delta R = \Delta L$. From the proof of Lemma 2, we know that ΔL must be greater than or equal to the sum of any B reward increments from list Z. Since Z contains all possible reward increments from all intervals, we have it also contains all selected rewards increments in algorithm 2. Because ΔR includes B reward increments, it follows that $\Delta R \leq \Delta L$.

Suppose $\Delta R < \Delta L$. Because ΔL is the sum of the top B reward increments in Z. It follows that ΔR must include an increment $\Delta J_i(x)$ from interval i such that it is outside the top B positions. In addition, there must be another reward increment $\Delta J_{i'}(x')$ within the top B positions from interval i' such that ΔR does not include it and $\Delta J_{i'}(x') > \Delta J_i(x)$.

Consider the moment that algorithm 2 selects $\Delta J_i(x)$. It must be greater than or equal to the reward increment $\Delta J_{i'}(y')$ from interval i' at that time. We know that the algorithm does not compute $\Delta J_{i'}(y')$ unless it has selected all possible reward increments with budget less than y' from interval i'. Because the algorithm does not select $\Delta J_{i'}(x')$, we have $x' \geq y'$. By the concavity property, $\Delta J_{i'}(x') \leq \Delta J_{i'}(y')$. Therefore, $\Delta J_{i'}(x') \leq \Delta J_{i'}(y') \leq \Delta J_i(x)$. However, this conflicts with the fact that $\Delta J_{i'}(x') > \Delta J_i(x)$. Thus,

it follows that $\Delta R = \Delta L$. Finally, by Lemma 1, Lemma 2, and Lemma 3, we have

$$\sum_{i=1}^{M} \Delta \tilde{J}_i^B = \Delta R = \Delta L = \sum_{i=1}^{M} \Delta \bar{J}_i^B \geq \sum_{i=1}^{M} \Delta \hat{J}_i^B. \tag{3.18}$$

□

Finally, the next theorem completes the proof.

Theorem 2.

$$\sum_{i=1}^{M} J_i(\tilde{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B). \tag{3.19}$$

Proof. Similar to the proof of Theorem 1, we have

$$\sum_{i=1}^{M} J_i(\tilde{k}_i^B) = \sum_{i=1}^{M} \Delta \tilde{J}_i^B + \sum_{i=1}^{M} J_i(0), \tag{3.20}$$

and

$$\sum_{i=1}^{M} J_i(\hat{k}_i^B) = \sum_{i=1}^{M} \Delta \hat{J}_i^B + \sum_{i=1}^{M} J_i(0). \tag{3.21}$$

by equation (3.13). By lemma 4, it follows that $\sum_{i=1}^{M} J_i(\tilde{k}_i^B) \geq \sum_{i=1}^{M} J_i(\hat{k}_i^B)$. Finally, because the budget allocation, $\hat{k}_1^B, \hat{k}_2^B, \cdots, \hat{k}_M^B$, is optimal, we have $\sum_{i=1}^{M} J_i(\tilde{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B)$. □

3.3.3 Complexity Analysis

We now determine the complexity of DPA in terms of the number of local reward evaluations.

Theorem 3. *With batch budget allocation, the number of local reward evaluations computed with DPA is:*

$$\mathrm{O}\big(\frac{1}{\epsilon^4} n\big), \tag{3.22}$$

and with incremental budget allocation, it is

$$\mathrm{O}\big(\frac{1}{\epsilon^4}B\big), \tag{3.23}$$

where ϵ is a number less than 1 such that $\epsilon = \frac{M}{n}$.

The inverse of ϵ reflects the subsection length. This complexity is a vast improvement over $\Theta(B^2n^3)$ for VoIDP-SCP. A brief outline of the proof is as follows.

Proof. Let the total budget be $B = B' + B''$, where B'' is the remaining budget allocated to each subsection after the initial uniform sampling with a budget of B'.

The proof with batch allocation is straight forward. In the budget allocation stage, VoIDP-SCP is run with a budget up to the number of unobserved state variables, which is $\mathrm{O}\big(\frac{1}{\epsilon}\big)$ for each subsection. Step 1 of the algorithm requires no computation of rewards, and Step 3 can reuse the rewards computed in Step 2. With $M = \epsilon n$ and using the complexity of VoIDP-SCP, we have the complexity of DPA as:

$$\mathrm{O}\big(M \cdot \mathrm{O}\big(\mathrm{O}\big(\frac{1}{\epsilon}\big)^2 \cdot (\frac{1}{\epsilon})^3\big)\big) = \mathrm{O}\big(\epsilon n(\frac{1}{\epsilon})^2(\frac{1}{\epsilon})^3\big) = \mathrm{O}\big(\frac{1}{\epsilon^4}n\big) \tag{3.24}$$

For incremental budget allocation, the complexity is similarly determined by the budget allocation step. However, VoIDP-SCP in this case is run on each subsection with a budget up to its allocated budget plus one. For analysis, we can ignore the constant one and the complexity is

$$\mathrm{O}\big(\sum_{j=1}^{M}(\tilde{k}_j^2(\frac{1}{\epsilon})^3)\big) = \mathrm{O}\big(\frac{1}{\epsilon^3}\sum_{j=1}^{M}\tilde{k}_j^2\big) \tag{3.25}$$

where $\tilde{k}_j$ is the allocated budget for interval j. Because $\sum_{j=1}^{M}\tilde{k}_j = B''$ and $\tilde{k}_j \leq \frac{1}{\epsilon}$ for all j, $\sum_{j=1}^{M}\tilde{k}_j^2$ reaches its maximum by letting as many intervals have budget $\frac{1}{\epsilon}$ as possible. Each of the other intervals either has 0 budget or has the remainder of the total budget.

As a result, the number of intervals that have nonzero budget is $\lceil B''/\frac{1}{\epsilon} \rceil = \mathrm{O}\big((B''/\frac{1}{\epsilon})\big) = \mathrm{O}\big((\epsilon B'')\big)$. Thus, we conclude that the number of local reward evaluation is in

$$\mathrm{O}\big(\frac{1}{\epsilon^3}\sum_{j=1}^{M}\tilde{k}_j^2\big) = \mathrm{O}\big(\frac{1}{\epsilon^3}\cdot\epsilon B''\cdot(\frac{1}{\epsilon})^2\big) = \mathrm{O}\big(\frac{1}{\epsilon^4}B\big) \tag{3.26}$$

□

3.4 Data Fusion for frame sequences

In this section, we first describe how we model a video as a graphical model. This model is a Markov chain which emits cheap and expensive features. Therefore, we can apply DPA to this model and Section 3.4.2 describes this.

3.4.1 A Graphical Model for Video

The graphical model is a Markov model, where each frame of video corresponds to a node. Each node contains a state variable that represents the property we wish to infer, such as whether a face or a moving object is present. Each node can emit two observable quantities, corresponding to cheap and expensive features extracted from the frame. This is similar to a hidden Markov model (HMM), but here we also model dependencies between observations. In our model, the value of a cheap feature at time t is not conditionally independent of the rest of the model given the state at time t, but is also dependent on the cheap feature at times $t-1$ and $t+1$. We do this to capture the fact that when an algorithm makes an error in one frame, it is quite likely to make a similar error at an adjacent frame. This model can be considered a type of autoregressive hidden Markov model [65]. We have experimentally verified in Section 3.5.5 that if

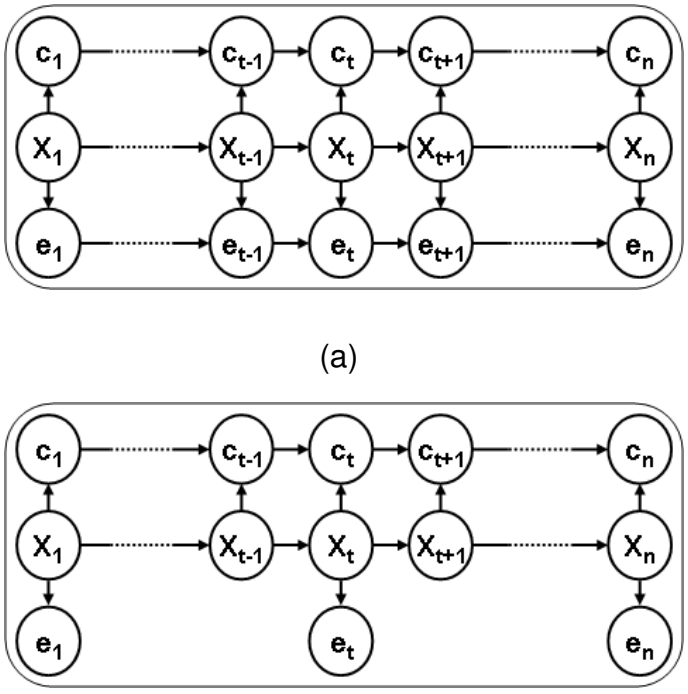


Figure 3.3: Markov models for video sequences. State variables are labeled "X", cheap observations are labeled "c", and expensive observations are labeled "e". They all have numbered subscripts indicating their time steps. (a) The model we use when expensive features are not available at every frame; (b) The model we use when all frames have expensive features.

we assume conditional independence between consecutive cheap features, the model will become overconfident about the evidence of the cheap features, resulting in less accurate inference.

We typically have expensive features for a small fraction of frames, so it is less important to model the dependency between them. In addition, we assume the expensive feature is accurate when predicting the state. Therefore, we assume expensive features depend only on the state. However, in cases where we have an expensive feature at every frame, we model their dependencies the same way we do with cheap features (see Figure 3.3).

3.4.2 Applying DPA

We have described DPA for the case of simple, chain graphical models. However, it is straightforward to apply it to the model in the previous section, since it has a chain structure and the same conditional independence property. We assume that the expensive feature is accurate in predicting the state of a node. This allows an expensive feature to play the role of an observation in our algorithm. In practice, expensive features do make mistakes. This means that the states before and after the frame at which we apply an expensive feature are not truly conditionally independent, but only approximately so. We experimentally evaluate the consequences of this approximation in the next section.

Finally, cheap features also provide useful information. Thus, we make the recursive formulas for computing the optimal reward be conditioned on applicable cheap features. Denoting $c_{a:b}$ as the cheap feature over the interval from variable X_a and X_b, the

formula (3.4) and (3.5) for computing the optimal reward become

$$\begin{aligned} J_{a:b}(x_a, x_b; k) \quad = \quad & \max\{J_{a:b}(x_a, x_b; 0), \\ & \max_{a<j<b}\{\sum_{x_j} P(X_j = x_j | X_a = x_a, X_b = x_b, c_{a:b}) \\ & \{R_j(X_j | X_j = x_j, c_{a:b}) + \max_{0 \le l \le k-1}[J_{a:j}(x_a, x_j; l) + \\ & J_{j:b}(x_j, x_b; k - l - 1)]\}\}\}, \end{aligned} \tag{3.27}$$

where the base case is

$$J_{a:b}(x_a, x_b; 0) = \sum_{j=a+1}^{b-1} R_j(X_j | X_a = x_a, X_b = x_b, c_{a:b}). \tag{3.28}$$

Finally, when predicting the state of frames and using formula (3.27) and (3.28) to determine where to sample expensive features, we need to determine the probability distribution of each state based on observations. We can easily extend the standard Forward-Backward algorithm [74] to do this.

3.4.3 Forward-backward Algorithm

When using formula (3.27) and (3.28) to determine where to sample expensive features, and using our model to do inference to predict the state of each frame, we need to determine the probability distribution of each state based on observations. We can use the Forward-Backward algorithm [74] to do this. Using our model, this algorithm works a little differently from the standard version, due to dependencies in the observations, and we describe it below. The derivation uses the ideas in [65] and [34].

Let N be the number of states, and denote individual states as $S = \{s_1, s_2, \cdots, s_N\}$. Let the sequence have T time slices. Also let cheap feature observations from time 1 to

T be $c_1, c_2, \cdots, c_T$ respectively, and expensive feature observations from time 1 to T be $e_1, e_2, \cdots, e_T$. Let f_i be $\{c_i, e_i\}$ at time i and O be all the observations. Similarly, we use X_t to denote the state variable at time t. Then,

$$\begin{aligned} P(X_t = s_i|O) &= \frac{P(O, X_t = s_i)}{P(O)} = \frac{P(f_{1\cdots T}, X_t = s_i)}{P(O)} \\ &= \frac{P(f_{t+1\cdots T}|X_t = s_i, f_{1\cdots t})P(X_t = s_i, f_{1\cdots t})}{P(O)} \\ &\propto P(f_{t+1\cdots T}|X_t = s_i, f_t)P(X_t = s_i, f_{1\cdots t}). \end{aligned} \tag{3.29}$$

Given the forward variable defined as $\alpha_t(i) = P(X_t = s_i, f_{1\cdots t})$ and the backward variable as $\beta_t(i) = P(f_{t+1\cdots T}|X_t = s_i, f_t)$, we have

$$P(X_t = s_i|O) \propto \alpha_t(i) \cdot \beta_t(i). \tag{3.30}$$

To compute $\alpha_t(i)$, we have

$$\begin{aligned} \alpha_t(i) &= P(X_t = s_i, f_{1\cdots t}) = P(X_t = s_i, f_t, f_{1\cdots t-1}) \\ &= P(X_t = s_i, f_t|f_{1\cdots t-1})P(f_{1\cdots t-1}) \\ &= \sum_{j=1}^{N} P(X_t = s_i, f_t|X_{t-1} = s_j, f_{1\cdots t-1})P(X_{t-1} = s_j|f_{1\cdots t-1})P(f_{1\cdots t-1}) \\ &= \sum_{j=1}^{N} P(f_t|X_{t-1} = s_j, f_{1\cdots t-1}, X_t = s_i)P(X_t = s_i|X_{t-1} = s_j, f_{1\cdots t-1})P(X_{t-1} = s_j, f_{1\cdots t-1}) \\ &= \sum_{j=1}^{N} P(f_t|f_{1\cdots t-1}, X_t = s_i)P(X_t = s_i|X_{t-1} = s_j)P(X_{t-1} = s_j, f_{1\cdots t-1}) \\ &= \sum_{j=1}^{N} P(f_t|f_{1\cdots t-1}, X_t = s_i)P(X_t = s_i|X_{t-1} = s_j)\alpha_{t-1}(j), \end{aligned} \tag{3.31}$$

where $P(f_t|f_{1\cdots t-1}, X_t = s_i) = P(c_t|c_{t-1}, X_t = s_i)P(e_t|X_t = s_i)$ for the left model and $P(f_t|f_{1\cdots t-1}, X_t = s_i) = P(c_t|c_{t-1}, X_t = s_i)P(e_t|e_{t-1}, X_t = s_i)$ for the right model in Fig. 3.3. This gives the recursive formula to compute the forward variable, where

the base condition is the same as that in [74]. The only difference from the standard forward algorithm is that the probability of the observation needs to be conditioned on the observation in the previous time step. To compute $\beta_t(i)$, we have

$$\begin{aligned}
\beta_t(i) &= P(f_{t+1\cdots T}|X_t = s_i, f_t) = \sum_{j=1}^{N} P(f_{t+2\cdots T}, X_{t+1} = s_j, f_{t+1}|X_t = s_i, f_t) \\
&= \sum_{j=1}^{N} P(f_{t+2\cdots T}|X_{t+1} = s_j, f_{t+1}, X_t = i, f_t) P(X_{t+1} = s_j, f_{t+1}|X_t = s_i, f_t) \\
&= \sum_{j=1}^{N} P(f_{t+2\cdots T}|X_{t+1} = s_j, f_{t+1}) P(f_{t+1}|X_{t+1} = s_j, f_t) P(X_{t+1} = s_j|X_t = s_i) \\
&= \sum_{j=1}^{N} \beta_{t+1}(j) P(X_{t+1} = s_j|X_t = s_i) P(f_{t+1}|X_{t+1} = s_j, f_t), \qquad (3.32)
\end{aligned}$$

where $P(f_{t+1}|X_{t+1} = s_j, f_t) = P(c_{t+1}|X_{t+1} = s_j, c_t)P(e_{t+1}|X_{t+1} = s_j)$ for the left model and $P(f_{t+1}|X_{t+1} = s_j, f_t) = P(c_{t+1}|X_{t+1} = s_j, c_t)P(e_{t+1}|X_{t+1} = s_j, e_t)$ for the right model in Fig. 3.3. Again, the only difference from the standard backward algorithm is that probability of the observation is conditioned on the observation in the previous time step.

3.5 Experiments

We now apply DPA to two vision tasks involving motion detection and face detection. Our main goal is to show that inference can be used to efficiently allocate processing in two very different tasks. We begin by first describing some common characteristics of our experiments in the next section. We then present the results for the two tasks in Section 3.5.2 and 3.5.3. Section 3.5.4 and 3.5.5 discuss how the expensive and cheap features can affect DPA. Section 3.5.6 shows how the subsection size affects the performance and

running time of the algorithm. Finally, Section 3.5.7 discusses how the concavity assumption holds for our sampling method.

3.5.1 General Experiment Setup

We use DPA described above to determine the locations at which to run the expensive algorithm. We uniformly sample the expensive feature at every 20 frames to break the sequence, while also running the cheap algorithm at every frame. We compare to several baseline algorithms. For all algorithms, when all the cheap and the necessary expensive observations have been made, we predict the state of each frame using the inference model in Fig. 3.3(a) since expensive features are not available in every frame. Competing algorithms are always provided the same total budget. We describe the budget in terms of its percentage of the total number of frames.

- The first baseline method is *uniform sampling*, which runs the expensive algorithm at a uniform step size. This method is in essence equivalent to running the expensive algorithm at a lower frame rate.

- The second baseline method is *most-relevant sampling*. We first run the cheap algorithm at each frame and perform inference using the model in Fig. 3.3(a) to obtain the conditional probabilities of all state variables. We then run the expensive algorithm on the frames that are most likely to satisfy our query. This is equivalent to using the cheap algorithm to prune the least interesting frames.

- The third and last baseline method is *most-uncertain sampling*. Similar to the most-relevant sampling method, we again run the cheap algorithm at each frame and then

perform inference to obtain conditional probabilities. Then, we run the expensive algorithm on frames that have the greatest uncertainty, measured by the entropy of the conditional probability.

- Finally, to calibrate the performance of algorithms on different tasks, we compared to an idealized method, *ceiling sampling*, in which we run the cheap and expensive algorithms at all the frames, i.e, the budget is equal to 100%. We use Fig. 3.3(b) to model the frame sequence because the expensive feature is available everywhere. This method should provide an upper bound on performance.

To compare these methods, we used 4-fold cross validation on each video. We recorded each video at 30 frames per second. We then uniformly sampled 3 frames per second to generate the training and testing sequences, as this is a reasonable frame rate for real world surveillance videos [58]. By beginning sampling at different locations, we produced 10 different sequences for training and also for testing. All ten sequences are used as training data. We also use all ten for testing, helping to smooth the results a bit. The performance measure we used was the 11-point average precision of the precision recall (PR) curves [63]. That is, we take the average precision for 11 uniformly spaced levels of recall. This is averaged over all 40 testing sequences from the 4 folds. We varied the total budget from 5% to 25% of n. Because the subsection size is 20, DPA and uniform sampling have the same performance at a budget of 5%. We next provide more details about the experiments for both tasks.

In addition to these three baseline methods, we also considered a greedy selection scheme. This acquires one expensive observation at a time [8, 38], choosing the obser-

vation that provides the greatest increase in overall expected reward. That is, given that we have already applied the expensive algorithm to all frames in O, we next choose the expensive observation, E_j, that maximizes $R(O \cup E_j) - R(O)$. Unfortunately, this approach is not suitable for our problem. First, to implement this approach, at each iteration we must perform inference for each possible value of E_j over the whole video to determine $R(O \cup E_j)$, and repeat this for each j. This requires an impractical amount of run time. We have performed preliminary experiments in which we choose ten observations at a time after each round of inference. However, not only is this still slow, but this approach performs poorly compared with other baseline methods. So we omit this method from further experiments.

3.5.2 Motion Detection

We first evaluated these algorithms in a simple background subtraction task. We collected three half hour videos at thirty frames per second, for a total of $n \approx 55,000$ per video, with each frame at 240×320 resolution. We hand-labeled each frame as "interesting" if it contained a moving object, such as a person or car, "uninteresting" otherwise.

As a cheap algorithm, we used FD [41] and we used IAGMM [113] as the expensive algorithm. For FD the feature was the number of foreground pixels in a frame after applying a threshold of 10. This avoided postprocessing, saving a significant amount of time. It is an interesting question for future work to determine how best to build a background model suitable for the expensive algorithm based on sparsely sampled frames.

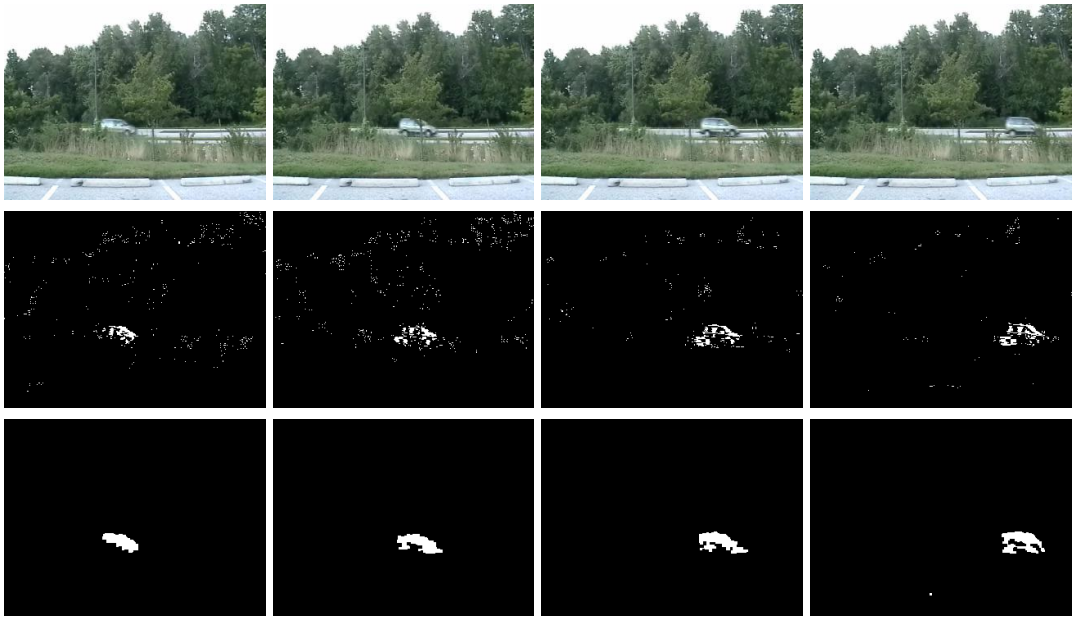


Figure 3.4: Top: examples from video used in the motion detection task. Middle: output of FD. Bottom: output of IAGMM.

However, in this experiment we wish to focus on the effectiveness of our algorithm in directing application of IAGMM. Therefore, we build a background model using all recent frames and then apply IAGMM only at sampled locations. After applying IAGMM, we performed an opening and a closing morphological operation. We then extracted the area of the largest connected component to generate features, which were then discretized. We had tried 8 different features, the area of the component, the width of its Bounding Box, and the diameter of a circle with the same area as the component. They all produced similar performance, and we chose the area of the component to show results. Figure 3.4 shows some examples; the other video and output are similar.

Next, we tested our concavity assumption on these videos, since DPA assumes that the reward curves were concave. Table 3.1 shows the results. We can see that over a half of all intervals produce concave reward curves, while most of the non-concave ones have

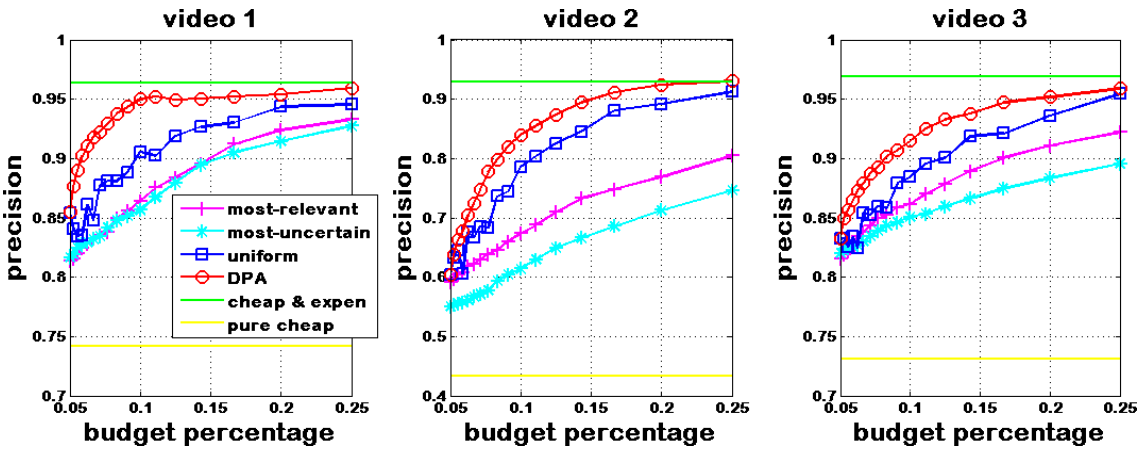


Figure 3.5: 11-point average precision values for the background subtraction task.

very small convexities.

We show 11-point average precision results on the three videos in Fig. 3.5. In all three videos, our method outperforms the baseline methods. We also observe from the plots that uniform sampling outperforms both most-relevant sampling and most-uncertain sampling. We postulate that the reason may be that the cheap algorithm does not produce high quality features and so decisions based purely on the cheap algorithm are unreliable. In these videos, FD faces difficulties because leaves often move in the background.

3.5.3 Face Detection

Next, we applied our approach to the problem of identifying frames containing a face. As with the last task, we collected three half-hour videos. We hand-labeled each frame as "interesting" if there is a frontal or profile face in it and labeled it as "uninteresting" otherwise.

For the cheap algorithm, we used IAGMM with the area of the largest connected

Table 3.1: Concavity of RB curves for the motion detection tasks.

Video	**Video 1**	**Video 2**	**Video 3**
Number of intervals[a]	2720	2720	2720
concave (%)	56.21	67.39	59.93
concave or $\approx$ concave[b](%)	94.34	97.28	94.45
Median of the rest	0.0468	0.0181	0.0834

[a] The total number of intervals over the 40 testing sequences.

[b] RB curves which are not concave but with a nonconcavity measure not greater than 0.01.

component as a feature since it is relatively good at detecting the motion of a human and is still computationally cheap compared to the face detector. The expensive algorithm was the face detection algorithm based on OpenCV [11], using the scheme in [52]. We used both frontal and profile face detectors and the expensive feature was a binary indicator of whether the detectors found a face. Fig. 3.6 shows examples from one of these videos, the others are similar.

We again first measured the concavity of the reward curves for the face detection videos. The results are given in Table 3.2. Again, the reward curves are largely concave. We show the 11-point average precision results on the three videos in Figure 3.7. Our

Figure 3.6: Top: frames from video used in face detection. Middle: output of IAGMM. Bottom: output of the face detector.

method outperforms the baseline methods in two videos, video 4 and 5, under all budget percentages. However, in video 6, our method has no advantage over uniform sampling when the budget is small, but as the budget increases, the advantage of our method becomes clear.

3.5.4 Accuracy of Expensive Features

The accuracy of the expensive feature can cause variations in performance of DPA, since our decisions are based on the assumption that the expensive feature is very accurate. The prediction error rates of the expensive feature in the six videos are .0147, .0417, .0391, .0582, .0828, and .2339 respectively. Note that the error rate is highest in the sixth video, the video on which our method has no clear advantage over uniform sampling when given a small budget.

Table 3.2: Concavity of RB curves for the face detection task.

Video	**video 4**	**video 5**	**video 6**
Number of subsections	2720	2840	2800
concave (%)	80.22	66.16	68.61
concave or $\approx$ concave (%)	98.16	98.45	99.07
Rest median	0.0970	0.1014	0.1014

Radovilsky et al. [75] shows that inaccurate observations lead to bounded loss on the subset selection version of VoIDP. To investigate how the accuracy of the expensive features affects our method, which is based on a conditional plan setting of VoIDP, we compare performance of different methods using synthetic expensive features with different accuracies for each video. Based on ground truth, we generate expensive features by randomly choosing a set of frames to make its value incorrectly reflect the actual states. We varied the size of the set to be 0%, 10%, 20%, 30%, and 40% of the total number of frames, and a smaller set is always the proper subset of a larger set. To measure the performance of a method, we use the mean of average precision for budget going from 5% to 25%. Fig. 3.8 shows the performance of different methods as the accuracy of the expensive features decreases. We observe that an accuracy over 85% in general is needed to give DPA an advantage over other methods.

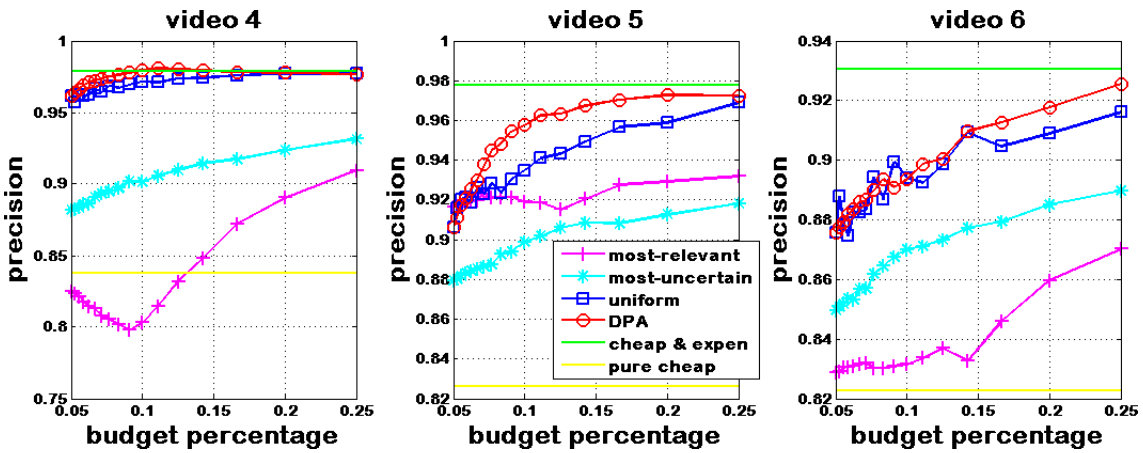


Figure 3.7: 11-point average precision values for the face detection task.

3.5.5 Usefulness of Cheap Features and Feature Dependency Modeling

Ideally, combining both features in DPA to determine the expensive feature sampling locations should give better results than purely using expensive features. To show this, we replace the cheap feature with a constant synthetic feature. Under this setting, the sampling locations and inference only depends on the expensive features. In addition, to show the importance of feature dependency modeling, we also compare to the result of using both types of features but without modeling the dependency between cheap features. That is, we remove the link between consecutive cheap features in Fig. 3.3(a), and the model becomes a standard HMM. We also experiment with varying frame rates, hoping to capture feature dependency at different levels.

Fig. 3.9 shows the performance when the frame rate is 30, 6, 3, 2, and 1. Under 3 frames per second, which is the frame rate in the experiments above, we observe that combining both features and modeling dependency has a clear advantage over using only the expensive features. Without dependency modeling, however, inference performance

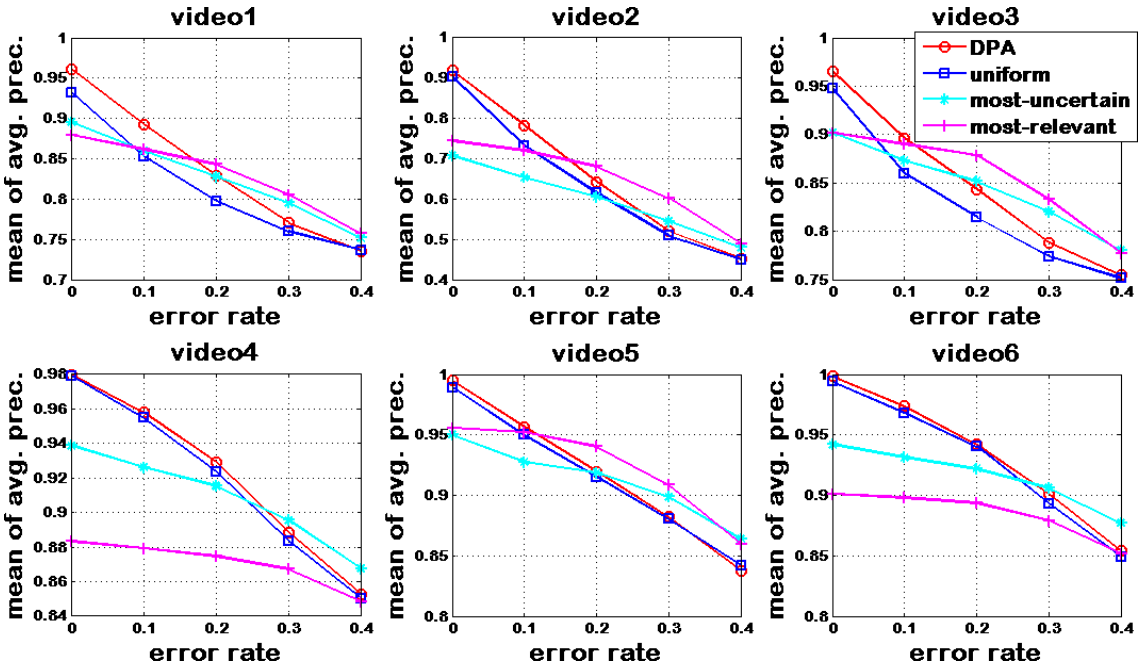


Figure 3.8: Mean of average precision as the accuracy of the expensive features decreases.

can sometimes be worse. For other frame rates, modeling feature dependency still has a clear advantage than that with no modeling. But we do observe that combining both features has no advantage when the frame rate is 30 or 6 frames per second. In particular, using purely expensive features has better performance for the first three videos for background subtraction at 30 frames per second. However, when the frame rate is 2 or 1 frame per second, the advantage of combining both features becomes clear again. In particular, even using the model without feature dependency is better than that using just expensive features when the frame rate is 1 frame per second. This is likely due to the issue of how well our model fits the data. When the frame rate becomes higher and higher, the dependency between errors made by cheap features is stronger and stronger, and our model cannot capture this well enough. As a result, the cheap feature is overconfident in some locations, and suppresses the correct decisions by the expensive features. Using a

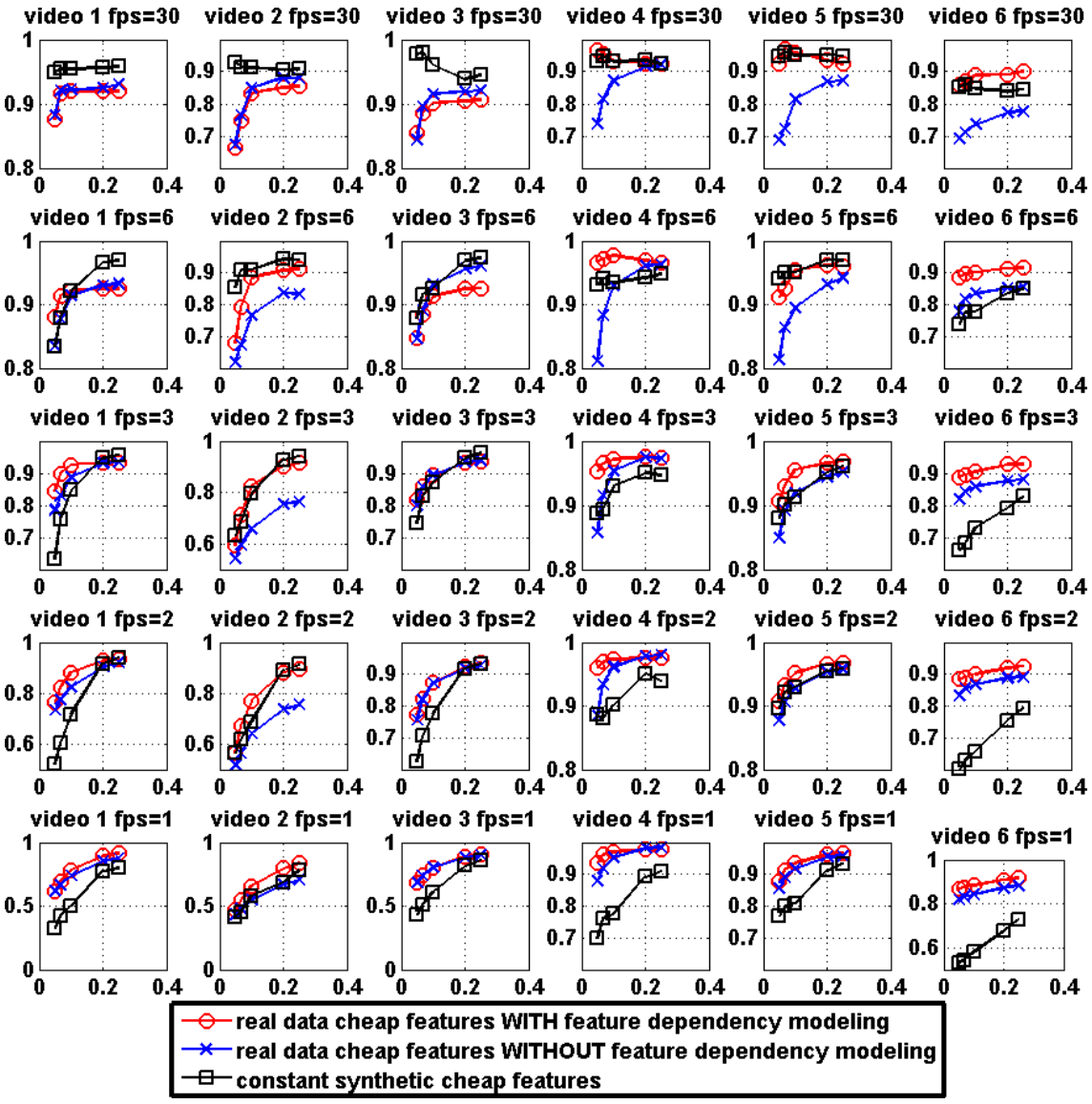


Figure 3.9: Comparison of using constant synthetic cheap features, and cheap features from real data with and without feature dependency modeling, using a frame rate of 30, 6, 3, 2, and 1 frames per second. As in Fig. 3.5, the x axis is budget and the y axis is average precision.

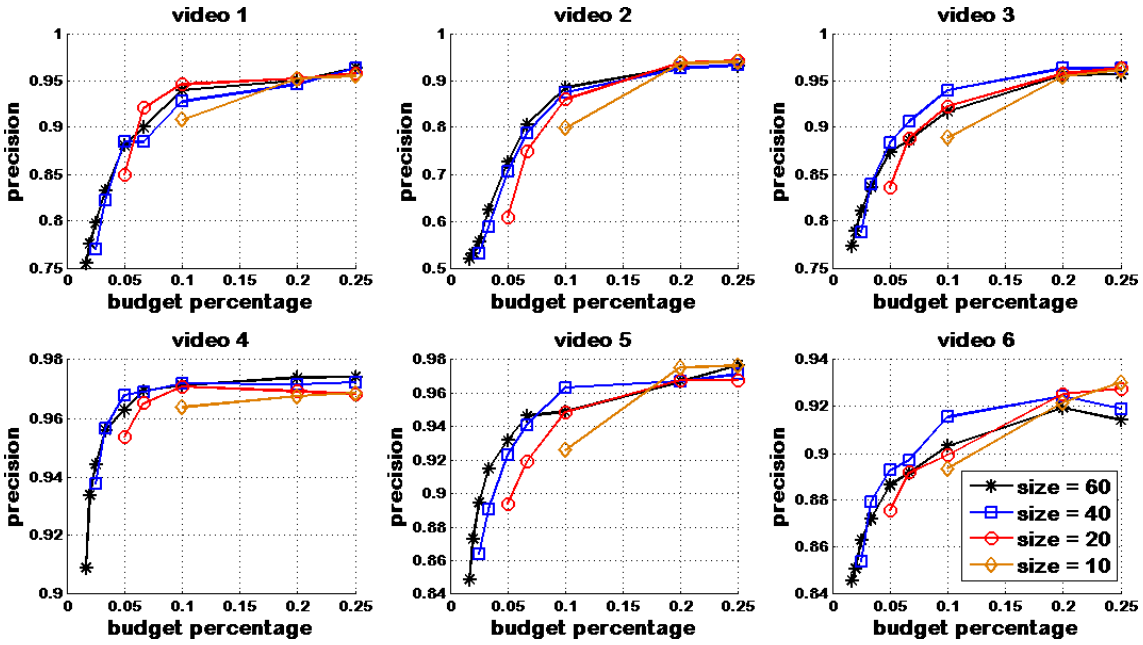


Figure 3.10: 11-point average precision values as size of subsections varies in DPA.

better model that captures this dependency adequately is one possible solution, and we discuss this more in Section 3.5.8. Real world videos, such as surveillance videos, need to run for a long time in general. A small frame rate is more suitable for storage and power issues [58]. At the same time, when cheap features are not helpful, DPA will still provide significant benefits by using information from expensive features to control processing.

3.5.6 Size of Subsections and Running Time

The size of subsections determines B', the budget used for uniform sampling. With larger subsections, more budget can be used within the subsections. However, a smaller subsection size does have advantages in terms of running time. Fig. 3.10 shows performance of DPA as the subsection size decreases. In general, we observe that a larger subsection size has better performance at the same budget level. This is because more

budget can be allocated by VoIDP-SCP with a larger subsection size, and the allocation is affected less by the uniform sampling. However, at a high budget level, performance of different subsection sizes start to converge. In addition, the performance at size 60 has no advantage over that of size 40 in many cases. This may indicate a saturation of performance as the size increases. The left side of Fig. 3.11 shows an example of the running time of VoIDP-SCP as the size of the subsections increases. The plot shows that we cannot afford to run VoIDP-SCP on the whole testing sequence in our experiment, which has length over 1000. The right side shows the running time of DPA with increasing subsection size averaged over all testing sequences from all videos. Each testing sequence is about one fourth of a 30-minute video at 3 frames per second. A large subsection size, such as 40 or 60, requires a fair amount of running time. A small size, such as 10, runs very fast, but suffers from disadvantages discussed above. In our experiments, we choose 20 as the subsection size, which has reasonable performance and fast running time.

3.5.7 Concavity

Our algorithm assumes that the RB curve is concave. Empirically, we find that this assumption holds well in our two, real-life domains. We have also performed experiments with synthetic data to get a sense of when this assumption might fail. We generate data from a Markov process as shown in Figure 3.3(b). Our experiments indicate that the key factor in determining the concavity of the RB curve is the probability of a state change from one node to the next. When states persist, RB curves tend to be concave. Figure 3.12 shows the variation in concavity with the probability that a state persists from one time

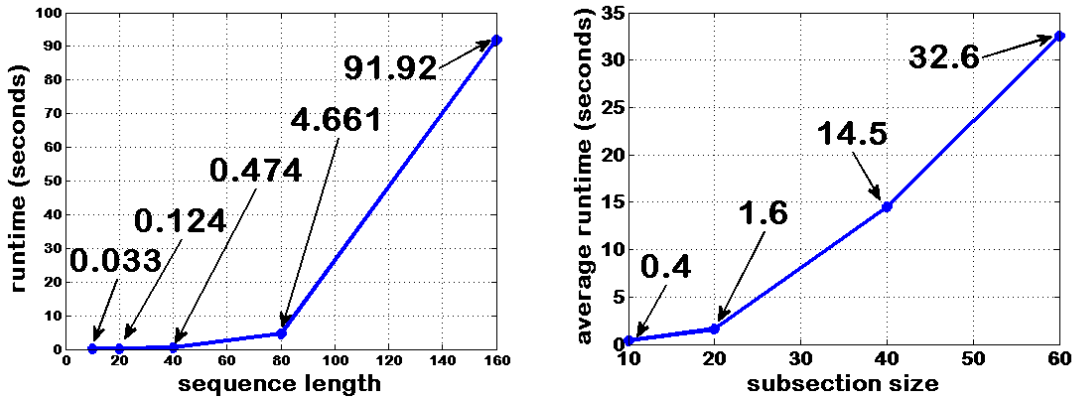


Figure 3.11: Left: running time of VoIDP-SCP at a budget of 25% of the sequence length. Right: running time of DPA at a budget of 25% averaged over all testing sequences from all videos. We use the multi-dimensional array to store the memory table in the dynamic programming. This allows the fastest table lookup speed. These measures are taken using a server with two 2.66GHz quad-core Xeon processors with 48GB of memory)

step to the next. Each curve pools results from a large number of simulations in which other parameters of the model vary. We also explored other state transition cases, such as when one state tends to persist while the other does not. However, their results are not as good as the case that both states tend to persist. We note that in our tasks, and in many other video analysis tasks, node states will persist; once an interesting event begins in a video it will tend to last for at least a few seconds. Therefore, we expect our results to be applicable in many settings.

3.5.8 Discussion

Our experiments demonstrate that DPA can make two potential contributions to video processing. First, we have shown that it is possible to use a Markov model to

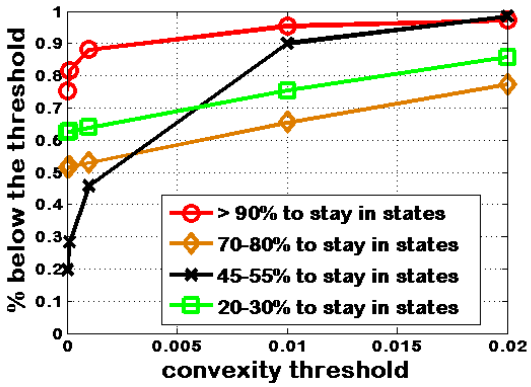


Figure 3.12: Percentage of subsections from all sampled sequences whose nonconcavity measure is not greater than a varying threshold. Each sampled sequence has a length of 2000, producing 100 subsections with a subsection size of 20. The threshold values are 0, 0.0001, 0.001, 0.01 and 0.02. The legend indicates probability for each state to stay in its current value. All other parameters of the model are sparsely sampled to cover their value ranges. Going from top to down in the legend, the number of sampled sequences used to generate the curve is 1728, 1750, 1956, and 1960 respectively.

integrate cheap and expensive features to improve system accuracy. Second, we have shown that by sparsely applying expensive features, our algorithm can use the results of inference to direct processing to portions of the video where further processing is most beneficial.

At the same time, we note that these potential benefits are dependent on some important assumptions. First, inaccurate expensive features will affect the performance of DPA. The advantage of DPA will decreases as the accuracy of the expensive features decreases. The results in Section 3.5.4 indicate that an 85% accuracy is in general needed to allow DPA to outperform other baseline methods. In many complex video analysis prob-

lems in unconstrained environments this accuracy is not yet achievable. For this reason we feel that DPA will be most relevant in two situations. First, in many controlled, or partially controlled environments, vision algorithms can achieve high accuracy. Second, in some applications a human analyst may serve as an expensive feature. It will be an interesting problem for future research to explore the use of DPA in integrating algorithmic output with human analysis.

It might also be of interest to modify VoIDP-SCP so that rather than considering all possible state assignments for each potential split point, it considers all possible expensive feature assignments. Such a method requires modification of formulas 3.4 and 3.5 for dynamic programming such that they are based on values of features rather than values of states. Since the feature space is in general much larger than the state space, this will be more computationally intensive. A similar idea has been exploited in the work by Radovilsky et al. [75]. However, they consider a subset selection setting of the problem which determines all sampling positions before samples are made, while we consider a conditional plan setting in which the next sampling position is conditioned on the sampled observations. These two problems have different recursive formulas for dynamic programming [48], and their approach may not be directly applicable to our problem.

A second issue that deserves further exploration is the modeling of dependency between features. Section 3.5.5 shows that this can improve inference. However, at high frame rates (eg., 30 fps) our model is still not able to properly capture these dependencies. It will be interesting to consider more sophisticated models, such as Conditional Random Fields (CRFs) [89]. CRFs can capture arbitrary dependencies among input observation variables, by conditioning on all inputs.

Finally, we have demonstrated our approach on relatively simple sample video that we have collected. It would be of interest to define tasks for which expensive features can achieve high accuracy in real-world surveillance datasets, such as i-LIDS MCTTR dataset [33]. On these challenging datasets, it might, for example, be of interest, to consider problems in which state-of-the-art algorithms act as a cheap feature and a human analyst serves as an expensive feature.

3.6 Summary

Our main goal has been to design inference algorithms that can be used to direct video processing. This allows us to replace simplistic methods such as reducing the frame rate with principled decisions that carry theoretical performance guarantees. We believe that this is a quite general framework that can be applied to many video processing tasks and may be extended in the future to more complex graphical structures.

To this end, we have made two more detailed contributions. First, we propose a graphical model that maps onto a video frame sequence and allows us to combine features from expensive and cheap algorithms to do inference. We show that in practical situations, there is much to be gained by this combination. Second, we have shown how to build on an existing algorithm that was designed for short chains to create an algorithm that runs efficiently on long video sequences. Specifically, we show that by applying an expensive algorithm in some extra locations, we can determine future sensing locations efficiently. Experiments with two concrete video processing tasks, low-level background subtraction and the higher level task of face detection, show that these can be mapped onto

our framework. The effectiveness of DPA's inference algorithm in these tasks illustrates the potential of our approach for general video processing.

Chapter 4

Learning SVMs with Latent Variables Using Structured Norms

Parts of the work from this chapter appear in the Advances in Neural Information Processing Systems (NIPS) Workshop on Optimization for Machine Learning in December 2011, [14].

4.1 Introduction

In the previous two chapters, we impose a budget on the resources we want to allocate. We also consider another idea for saving computational resources and maintaining program performance in the context of latent variable models without explicitly specifying the budget. We consider learning the latent state space complexity to simplify the model and reduce inference time. We start with describing the motivation of using the latent variable models. In the situation such that some variables relevant to the problem are not annotated in the datasets, latent variable models provide an ideal abstraction. For example, consider the task of training a person detector. Standard benchmarks only provide bounding box annotations indicating the presence of people. However, people tend to be highly articulated objects and in order to detect a person, it is often essential to reason about the pose of the person in terms of configuration of parts: *i.e*. location of head, torso, limbs – all quantities not labelled in the dataset.

This information is, therefore, more natural to model as latent variables. It allows

for modelling of interaction between the observed data (e.g. image features) and latent or hidden variables not observed in the training data (e.g. location of body parts). These hidden variables may help explain correlations in the features, provide a low-dimensional embedding of the input, or help with prediction. For example, in a deformable-part model [32] for person detection, these part locations are latent variables that help model articulations of the human body and help localize the person. In handwritten digit recognition, deformations of digit images, such as rotation, can be modelled as latent variables to greatly improve recognition accuracy [50, 110]. In document retrieval, the total ranking order of all documents related to a query can be modeled as a latent variable to help produce a higher number of relevant documents in the top $\boldsymbol{k}$ returned results.

Training latent variable models, however, is notoriously problematic, since it typically involves a difficult non-convex optimization problem. Common algorithms for solving these problems, Expectation-Maximization (EM) [21] and the Concave-Convex Procedure (CCCP) [32, 110, 112], are known to be highly sensitive to initialization and prone to getting stuck in a poor local optimum. Standard techniques for mitigating the poor behaviour of these algorithms include multiple restarts with random initializations, smoothing the objective function and annealing. Recently, Bengio *et al*. [7] and Kumar *et al*. [50] have presented a curriculum learning scheme that trains latent variable models in an easy-to-difficult manner, by initially pruning away difficult examples in the dataset.

Our goal at a high-level is to study the modelling-optimization tradeoff in designing latent variable models for computer vision problems. From a modelling perspective, we would like to design models with ever more complex latent variables, e.g. capture location of parts, their scale, orientation, appearance. However, from an optimization perspective,

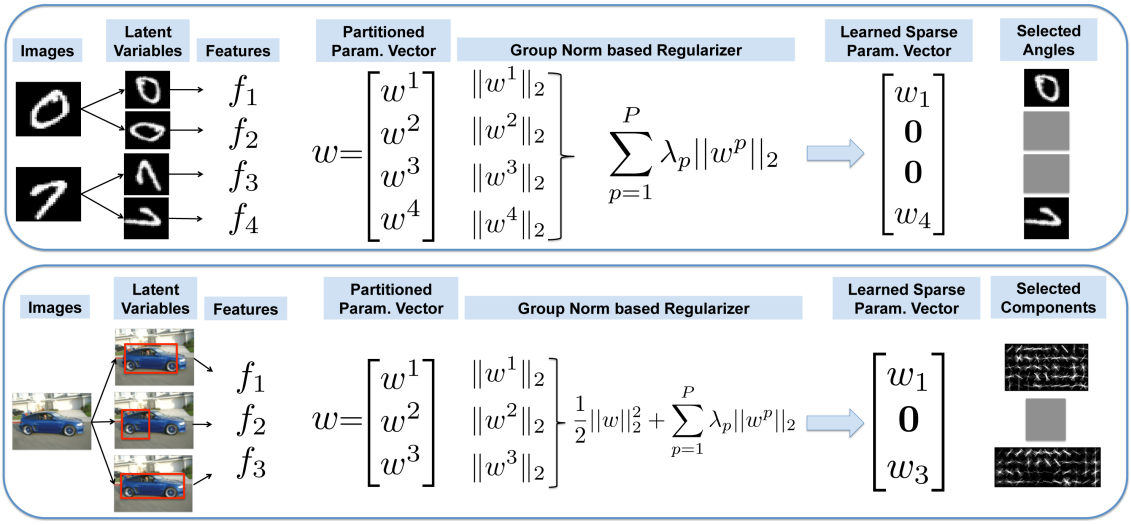


Figure 4.1: Overview of our approach for the digit recognition (top) and object detection experiment (bottom). In digit recognition, the latent state space is the rotation angles. For object detection, the latent state space is the component label in the mixture of deformable part models. The model parameter vector is partitioned into groups corresponding to different states. Parameters for non-informative states become zero in the final model under such regularizers, allowing us to select meaningful states for prediction.

complex models are more difficult to train than simpler ones, more prone to getting stuck in a bad local minimum, ultimately resulting in poor generalization (typically even worse than simpler models). In most existing models, the complexity of the latent variable space is typically left as a free design choice that is hand-tuned. Thus, the question we seek to answer is: Is there a principled way to learn the complexity of the latent space in a latent variable model?

In this chapter, we propose the use of structured sparsity inducing norms like ℓ_1-ℓ_2 to estimate the parameters of a latent-variable model, thereby regularizing the complexity

of the latent space. Structured sparsity inducing norms are a generalization of the ℓ_1 norm and regularize solutions to be sparse in a structured way. Specifically, the group ℓ_1-ℓ_2 norm behaves like an ℓ_1 norm at a group level and encourages groups of variables to be sparse.

Note that this is a subtle yet important difference between our goal and the typical scenario in which sparsity inducing norms are used. Traditional approaches are interested in variable selection, *i.e*. which latent variables should be included in the model. We are interested in state design, *i.e*. which latent variables states should we model for optimum performance. Our motivation is the observation that often most latent variables have a few key informative states (*e.g*. is the head on top of the body or not) while most of the remaining states may be pruned without any loss in performance (*e.g*. the head being besides or below or around the body are all equally bad, so only one state may be kept). Traditional approaches would compare the cumulative (or mean) informativeness of different variables, while we aim to directly prune the non-informative states. The key challenge is in identifying these informative states from data, rather than making hard design choices before looking at the data.

Our approach for solving this problem utilizes some of the same classical ideas as variable selection. We divide the latent variable state space into different groups, among which the group norm is induced. Since the group norm encourages group-sparsity, this allows simultaneous parameter estimation as well as state selection. Conceptually, this is an elegant solution since it gives the designer of a latent model tremendous flexibility in including plenty of latent variables without being concerned about the optimization issues – the group norm will automatically prune out latent variable states that are not helpful

for prediction, while still utilizing all latent variables that have some informative states. Our approach is in a sense orthogonal to that of Bengio *et al*. [7] and Kumar *et al*. [50], in that they prune out difficult training examples to make the non-convex optimization easier, while we prune out difficult (or irrelevant) latent states.

We apply our approach to SVMs with latent variables, for both the binary and structured output case. We perform two sets of experiments: handwritten digit recognition on MNIST and object detection on the PASCAL VOC 2007 dataset [28]. Our first set of experiments show that our approach is indeed able to prune the complexity of latent space, resulting in a model that allows significantly faster inference at test time without a drop in accuracy over a complete (non-sparse) model. Our second set of experiments show that our approach is able to learn a better model by adapting the complexity of the latent variable space to the category being trained.

4.2 Prior Work

Most relevant to our work are algorithms for discovering latent structure in latent variable models and other applications of structured-sparsity-inducing norms. These are both broad goals and cover a vast amount of literature. We mention the works most directly relevant to our approach.

Latent variable models have been used to model observations in both generative and discriminative settings. In the generative setting, the goal is to explain the data with a low-dimensional latent structure. Mixture models like Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) have a long history in applications such as speech recognition [74]. More recently, a number of discriminative latent models such as Hid-

den Conditional Random Field [101], Latent SVMs [32] and Latent Structural SVMs (LSSVMs) [110] have been proposed. These models have demonstrated success in a number of applications. They differ from generative models in the sense that the ultimate goal is prediction not explanation of the data.

In both kinds of models, the parameter learning problem is non-convex and solved with techniques like Expectation-Maximization (EM) [21] and Concave-Convex Procedure (CCCP) [32, 110, 112] respectively. Kumar et al. [50] introduce self-paced learning for latent variable models and demonstrate its effectiveness for learning LSSVMs. Note that for all the models above, the latent variables and their state space are predefined and fixed for specific applications. Our approach, on the other hand, aims for parameter estimation as well as discovery of meaningful latent variable states.

Related to this goal of discovery is the work of Chandrasekaran et al. [13], which attempt to identify the graphical model structure assuming that latent and observed variables are jointly Gaussian. Our work is different in that we are interested in prediction via a sparse latent model and not identification of such a model. Moreover, we make no Gaussian assumptions, which may be infeasible for applications. Salzmann et al. [81] propose an approach to encourage a shared-private factorization of latent space to be nonredundant and simultaneously discover the dimensionality of the latent space. Such state dimensionality learning is based on trace norm while we use group norm to achieve this goal. Jia et al. [43] use the group norm to learn a latent space with shared-private factorization with a limited number of latent dimensions. However, they follow a dictionary learning approach. Regularized by the group norm, the latent space is learned by minimizing the norm of the difference between an observation matrix and a product

between a dictionary and a latent embedding matrix. Our approach, however, is based on risk minimization using slack variables and the group norm. The former is related to continous observations and based on regression models, and the later is related to discrete observations and is based on discriminative models.

There is a fairly mature body of work on ℓ_1 regularization for sparse regression models [18, 26, 90]. Sparse coding with ℓ_1 regularization has been successfully used to solve many problems in compressed sensing [25] and signal processing [61]. Yuan and Lin [111] introduce group norm regularization to allow parameter estimation as well as selection of certain groups of variables. Bengio et al. [6] the apply group norm to build a word dictionary in bag-of-words document representations widely used in text, image, and video processing. Bach [1] proposes general sparsity inducing structured norms. A survey by Bach et al. [2] describes a list of applications of the group norm, including group Lasso [92, 111], multitask learning [54, 67, 73], and multiple kernel learning [3].

The rest of this paper is organized as follows: Section 4.3 and Section 4.4 revisit SVMs with latent variables and describe our proposed group norm modification. Section 4.5 describes how parameter learning can be performed in this model. Finally, Section 4.6 describes the two sets of experiments.

4.3 SVMs with Latent Variables

We begin by first giving an overview of the Latent Structural SVM model and then specializing it to the latent SVMs used for binary classification experiments.

Notation. For any positive integer n, let $[n]$ be shorthand for the set $\{1, 2, \ldots, n\}$. We

denote training data as $\mathcal{D} = \{(x_i, y_i) \mid i \in S \equiv [n]\}$, where $x_i \in \mathcal{X}$ is the (input) observed feature-vector and $y_i \in \mathcal{Y}$ is the (possibly structured) output label for the i^{th} sample. In addition, let $h_i \in \mathcal{H}$ denote the latent variable for the i^{th} sample. For example, in digit recognition, x_i is the original digit image, $y_i \in \{0, 1, \ldots, 9\}$ is the true digit label and h_i is the (deformation) rotation angle that must be corrected for before extracting features. We use $\boldsymbol{v}^T$ to denote transpose of $\boldsymbol{v}$, which can either be a vector or a matrix.

Latent Structured SVMs (LSSVMs). The linear prediction rule of LSSVMs is of the following form:

$$f_{\boldsymbol{w}}(x) = \max_{(y,h)\in\mathcal{Y}\times\mathcal{H}} \boldsymbol{w} \cdot \phi(x, y, h), \tag{4.1}$$

where $\phi(x, y, h)$ is the joint feature vector that encodes the relationship between the input, hidden and output variables, and $\boldsymbol{w}$ is the model parameter vector. In digit recognition, this joint feature vector is the vector representation of the image x rotated by the angle corresponding to h. Let

$$\{\hat{y}_i(\boldsymbol{w}), \hat{h}_i(\boldsymbol{w})\} \triangleq \operatorname*{argmax}_{(y,h)\in\mathcal{Y}\times\mathcal{H}} \boldsymbol{w} \cdot \phi(x_i, y, h) \tag{4.2}$$

be the predicted output and latent variables for data-point i, written as a function of the parameter vector $\boldsymbol{w}$. A user-specified risk function $\Delta(y_i, \hat{y}_i(\boldsymbol{w}))$ measures the loss incurred for predicting $\hat{y}_i(\boldsymbol{w})$ for the i^{th} sample, when the ground-truth label is y_i. Note that the risk function may additionally depend on the predicted latent variables, *i.e.* have the form $\Delta(y_i, \hat{y}_i(\boldsymbol{w}), \hat{h}_i(\boldsymbol{w}))$. The parameter vector $\boldsymbol{w}$ is learned by minimizing the (regularized) risk of the prediction on the training dataset $\mathcal{D}$. Unfortunately, this is a difficult optimization problem. Yu and Joachims [110] proposed minimizing an upper-bound on

the risk and formulated the following optimization problem:

$$\min_{\boldsymbol{w}} \quad \Omega(\boldsymbol{w}) + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \tag{4.3a}$$

$$s.t. \quad \max_{h_i \in \mathcal{H}} \boldsymbol{w} \cdot \Big(\phi(x_i, y_i, h_i) - \phi(x_i, \hat{y}_i, \hat{h}_i)\Big) \geq \Delta(y_i, \hat{y}_i, \hat{h}_i) - \xi_i, \tag{4.3b}$$

$$\xi_i \geq 0 \tag{4.3c}$$

$$\forall (\hat{y}_i, \hat{h}_i) \in \mathcal{Y} \times \mathcal{H},\ i \in S.$$

where, the regularization term is $\Omega(\boldsymbol{w}) = \Omega_{\ell_2}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$. Intuitively, we can see that constraint (4.3b) tries to ensure that for each training instance i, the ground-truth and its best latent variable prediction have a higher score than all other labels and latent variable assignment pairs by a loss function $\Delta(y_i, \hat{y}_i, \hat{y}_i)$, subject to a positive slack variable, ξ_i, which allows this rule to be violated to some degree. $\Delta(y_i, \hat{y}_i, \hat{y}_i)$ is the score margin. High-risk configurations are forced to have a larger margin between them and the ground-truth. It can be shown that ξ_i is an upper bound on the risk, *i.e.* $\xi_i \geq \Delta(y_i, \hat{y}_i(\boldsymbol{w}), \hat{h}_i(\boldsymbol{w}))$. We refer the reader to [110] for more details about this formulation.

Latent SVMs (LSVMs). The fairly general formulation of LSSVMs includes a number of interesting models as special cases. We describe one such instantiation, the deformable-parts based Latent SVM model of Felzenszwalb *et al.* [32], which we use for one set of our experiments. In this model, x_i are the HOG descriptors [20] computed at a particular sliding window location and scale in the image; $y_i \in \{+1, -1\}$ indicates presence or absence of a particular category in the window and h_i indicates the mixture type of the deformable template and location and scale of root and part filters. The scoring function

in this case can be reduced to the following form:

$$f_{\boldsymbol{w}}(x) = \max_{h \in \mathcal{H}} \boldsymbol{w} \cdot \phi(x, h), \tag{4.4}$$

where the joint feature vector $\phi(x, h)$ now does not depends on the label y. The risk function is a zero-one loss function, and the learning problem looks more like a binary SVM:

$$\min_{\boldsymbol{w}} \quad \Omega(\boldsymbol{w}) + C \sum_{i=1}^{n} \xi_i, \tag{4.5a}$$

$$s.t. \quad y_i f_{\boldsymbol{w}}(x_i) \geq 1 - \xi_i, \tag{4.5b}$$

$$\xi_i \geq 0 \tag{4.5c}$$

$$\forall i \in S.$$

where $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$. Different from those in LSSVMs, constraints (4.5b) try to ensure that positive and negative training instances lie on different sides of the separation hyperplane subject to positive slack variables so that this rule can be violated to some degree. The next section describes our proposed group norm modification to the LSSVM and LSVM models, and Section 4.5 describes how parameter learning can be performed in the presence of this modification.

4.4 Inducing Group Norm for State Learning

Let $\mathcal{H} = \{1, \cdots, P\}$ be the set of states which the latent variables can take. Such a set of states, for example, can be the set of all possible rotation angles in digit recognition, or the set of object components in a part based object detection model. Recall that our goal is to regularize the complexity of the latent space and learn which hidden states are

really relevant for the prediction problem. To this end, we consider using the ℓ_1-ℓ_2 norm in the regularizer $\Omega(\boldsymbol{w})$ in problem (4.3) and (4.5) to learn meaningful latent variable states.

We start by describing a modification to the linear prediction rule in problem (4.3) that makes it easier to encode the group structure of latent states. Specifically, instead of learning a single weight vector $\boldsymbol{w}$, we now learn P weight vectors $\boldsymbol{w}^{\hbar}$, one corresponding to each of the latent states. Let the parameter vector for the p^{th} group be denoted by $\boldsymbol{w}^p = [w_1^p, \cdots, w_{n_p}^p]$, where n_p is the length of this vector. The modified linear prediction rule is given by

$$f_{\boldsymbol{w}}(x) = \max_{y \in \mathcal{Y}, \hbar \in [P]} \boldsymbol{w}^{\hbar} \cdot \phi(x, y, \hbar). \tag{4.6}$$

We note that with appropriate zero-padding of the features, this model is equivalent to the original linear model. To see that, let $\boldsymbol{w} = [\boldsymbol{w}^1, \cdots, \boldsymbol{w}^P]$ be the concatenation of weight vectors from each group. We can also define new features:

$$\tilde{\phi}(x, y, \hbar) = [\mathbf{0}_{n_1}, \mathbf{0}_{n_2}, \ldots, \phi(x, y, \hbar), \ldots, \mathbf{0}_{n_P}]^T. \tag{4.7}$$

The above equation zeros pads the joint feature vectors such that only the weight vectors and features for the same group interact in the dot product: $\boldsymbol{w} \cdot \tilde{\phi}(x, y, \hbar) = \sum_{p \neq \hbar} \mathbf{0}_{n_p} \cdot \phi(x, y, p) + \boldsymbol{w}^{\hbar} \cdot \phi(x, y, \hbar)$. Similarly, we can partition the parameter vector $\boldsymbol{w}$ and build a joint feature vector $\phi(x, h)$ according to this partition in problem (4.5).

The key reason for working with this representation is that parameters for each state are now represented separately and thus group ℓ_1 regularization is possible over the state space. We directly apply the group norm in $\Omega(\boldsymbol{w})$ to perform this regularization in

problem (4.3).

$$\Omega_{\mathcal{G}}(\boldsymbol{w}) = \sum_{p=1}^{P} \lambda_p \left\| \boldsymbol{w}^p \right\|_q , \tag{4.8}$$

for any $q \in [1, \infty)$, where $\lambda_p \geq 0$ is the regularization weight for group p. This norm is usually referred as ℓ_1-ℓ_q norm, and in practice, popular choices for q are $\{2, \infty\}$ [2]. In our work, we only consider $q = 2$ and the regularizer is thus given as follows:

$$\Omega(\boldsymbol{w}) = \sum_{p=1}^{P} \lambda_p \left\| \boldsymbol{w}^p \right\|_2 . \tag{4.9}$$

Within each group, the ℓ_2 norm is used, which does not promote sparsity. At the group level, this norm behaves like the ℓ_1 norm and thus induces group sparsity, *i.e.* the parameters of some groups are encouraged to be set completely to zero. Uninformative states will thus have sparse learned parameters. This gives us a way to select the most useful states for prediction and shrink the state space size.

In problem (4.5), we use a different strategy to apply the group norm. Similar to the idea of Elastic Nets [114], we can use the group norm in combination with the ℓ_2 norm:

$$\Omega(\boldsymbol{w}) = \frac{1}{2} \left\| \boldsymbol{w} \right\|_2^2 + \sum_{p=1}^{P} \lambda_p \| \boldsymbol{w}^p \|_2 . \tag{4.10}$$

Such a regularizer has the effect of both the original regularizer and the group norm. When $\lambda_p = 0$ for all p, the regularizer is reduced to the original form (ℓ_2-norm). Group level sparsity can be induced when λ_p is sufficiently large. Note that both approaches are not feasible when the latent space is structured (trees, etc) and thus exponentially large. We will come back to discuss this issue in Section 5.2.2. The next section gives a detailed description of our algorithm for solving these problems.

4.5 Coordinate Descent

From an optimization perspective, both problems (4.3) and (4.5) can be viewed as minimizing a sum of convex and concave functions. Such problems are studied in the context of difference of convex programming and lend themselves to the concave-convex procedure (CCCP) [110, 112] and a similar coordinate descent approach described in Felzenszwalb [32]. We first describe the algorithm to solve problem (4.3). We rewrite the problem as:

$$\min_{\boldsymbol{w}} L(\boldsymbol{w}) \doteq \min_{\boldsymbol{w}} \left[\Omega(\boldsymbol{w}) + \frac{C}{n} \sum_{i=1}^{n} \max\left\{0, f_i(\boldsymbol{w}) - g_i(\boldsymbol{w})\right\} \right] \tag{4.11a}$$

$$\text{where} \quad f_i(\boldsymbol{w}) = \max_{(\hat{y}_i, \hat{h}_i) \in \mathcal{Y} \times \mathcal{H}} \left[\boldsymbol{w} \cdot \phi(x_i, \hat{y}_i, \hat{h}_i) + \Delta(y_i, \hat{y}_i, \hat{h}_i) \right], \tag{4.11b}$$

$$g_i(\boldsymbol{w}) = \max_{h_i \in H} \boldsymbol{w} \cdot \phi(x_i, y_i, h_i). \tag{4.11c}$$

We can now see that $L(\boldsymbol{w})$ is a difference of two functions $f_i(\boldsymbol{w})$ and $g_i(\boldsymbol{w})$ that are both convex, since they are point-wise maximums of convex (linear) functions. In order to minimize (4.11a), we follow the approach of Felzenszwalb et al. [32] and minimize the following upper bound on $L(\boldsymbol{w})$:

$$\min_{\boldsymbol{w}, \{h_i\}} L_h(\boldsymbol{w}, \{h_i\}) \doteq \min_{\boldsymbol{w}, \{h_i\}} \left[\Omega(\boldsymbol{w}) + \frac{C}{n} \sum_{i=1}^{n} \max\left\{0, f_i(\boldsymbol{w}) - g_i(\boldsymbol{w}, h_i)\right\} \right], \tag{4.12}$$

$$\text{where} \quad g_i(\boldsymbol{w}, h_i) = \boldsymbol{w} \cdot \phi(x_i, y_i, h_i). \tag{4.13}$$

Intuitively, by replacing $g_i(\boldsymbol{w})$ by $g_i(\boldsymbol{w}, h_i)$ implies that we enforce the margin not with respect to the best latent assignment for the ground-truth, rather only the current latent assignment of h_i. $L_h(\boldsymbol{w}, \{h_i\})$ is thus the objective function with latent variables specified for the training data. Fixing the latent variables makes $L_h(\boldsymbol{w}, \{h_i\})$ convex in $\boldsymbol{w}$.

Moreover, $L(\boldsymbol{w}) \leq L_h(\boldsymbol{w}, \{h_i\})$. In a manner similar to Felzenszwalb et al. [32], we follow an alternating coordinate descent and subgradient descent scheme. At iteration t, we first fix $\boldsymbol{w}^t$ and optimize $L_h(\boldsymbol{w}^t, \{h_i\})$ w.r.t. $\{h_i\}$. This is equivalent to computing

$$h_i^{t+1} = \underset{h_i \in \mathcal{H}}{\operatorname{argmax}} \ \boldsymbol{w}^t \cdot \phi(x_i, y_i, h_i), \forall i \in S \tag{4.14}$$

This step is fairly straightforward and involves assigning the latent variables to their optimal states given the current setting of $\boldsymbol{w}^t$. Next, we fix $\{h_i^{t+1}\}$ and optimize w.r.t. $\boldsymbol{w}$. This is done via subgradient descent. The subgradient $\nabla L_h(\boldsymbol{w}, \{h_i^{t+1}\})$ is given by:

$$\nabla L_h(\boldsymbol{w}, \{h_i^{t+1}\}) = \nabla \Omega(\boldsymbol{w}) + \frac{C}{n} \sum_{i=1}^{n} m_i(\boldsymbol{w}, h_i^{t+1}), \tag{4.15}$$

where

$$\nabla \Omega(\boldsymbol{w}) = \Bigg[\underbrace{\frac{\lambda_1 w_1^1}{||\boldsymbol{w}^1||_2}, \cdots, \frac{\lambda_1 w_{n_1}^1}{||\boldsymbol{w}^1||_2}}_{\text{Group 1}}, \cdots, \underbrace{\frac{\lambda_P w_1^P}{||\boldsymbol{w}^P||_2}, \cdots, \frac{\lambda_P w_{n_P}^P}{||\boldsymbol{w}^P||_2}}_{\text{Group } P} \Bigg]^T, \tag{4.16}$$

and

$$m_i(\boldsymbol{w}, h_i^{t+1}) = \begin{cases} 0 & \text{if } f_i(\boldsymbol{w}) - g_i(\boldsymbol{w}, h_i^*) \leq 0 \\ \phi(x_i, \hat{y}_i^*, \hat{h}_i^*) - \phi(x_i, y_i, h_i^{t+1}) & \text{otherwise} \end{cases} \tag{4.17}$$

where

$$(\hat{y}_i^*, \hat{h}_i^*) = \underset{(\hat{y}_i, \hat{h}_i) \in \mathcal{Y} \times \mathcal{H}}{\operatorname{argmax}} \ f_i(\boldsymbol{w}) = \underset{(\hat{y}_i, \hat{h}_i) \in \mathcal{Y} \times \mathcal{H}}{\operatorname{argmax}} \ \Big[\boldsymbol{w} \cdot \phi(x_i, \hat{y}_i, \hat{h}_i) + \Delta(y_i, \hat{y}_i, \hat{h}_i) \Big]. \tag{4.18}$$

These two steps, *i.e.* fixing $\boldsymbol{w}^t$ and optimizing $L_h(\boldsymbol{w}^t, \{h_i\})$ w.r.t. $\{h_i\}$, and fixing $\{h_i^{t+1}\}$ to optimize w.r.t. $\boldsymbol{w}$, keep iterating until the objective L_h converges. It can be shown that the algorithm always converges to a local minimum or a saddle point [112]. Algorithm 3 describes the entire algorithm. Following [47], we choose the learning rate at iteration

t to be $\alpha_t = \frac{1}{\eta_t+1}$, where η_t is the number of times the objective value $L_h(\boldsymbol{w}, \{h_i^{t+1}\})$ has increased from one iteration to the next. This learning rate performed well in the handwritten digit recognition experiments.

Problem (4.5) can be solved using a similar coordinate descent method and we use similar notations to illustrate this approach. The original objective (4.5) can be written as

$$\min_{\boldsymbol{w}} L(\boldsymbol{w}) = \min_{\boldsymbol{w}} F(\boldsymbol{w}) + G(\boldsymbol{w}), \tag{4.19a}$$

$$\text{where} \qquad F(\boldsymbol{w}) = \Omega(\boldsymbol{w}) + C \sum_{i \in S_-} \max\left\{0, 1 + \max_{h_i \in \mathcal{H}} \boldsymbol{w} \cdot \phi(x_i, h_i)\right\}, \tag{4.19b}$$

$$G(\boldsymbol{w}) = C \sum_{i \in S_+} \max\left\{0, 1 - \max_{h_i \in \mathcal{H}} \boldsymbol{w} \cdot \phi(x_i, h_i)\right\}. \tag{4.19c}$$

where S_+ is the set of positive training instances and S_- is the set of negative training instances. The objective $L(\boldsymbol{w})$ is not convex because $F(\boldsymbol{w})$ is convex and $G(\boldsymbol{w})$ is concave. However, if the latent values for all positive training instances are fixed, the resulting objective is convex and can be minimized using gradient descent. Therefore, we consider the latent configuration of all positive training instances as variables and minimize the following objective

$$\min_{\boldsymbol{w}, \{h_i\}} L_h(\boldsymbol{w}, \{h_i\}) = \min_{\boldsymbol{w}} F(\boldsymbol{w}) + G(\boldsymbol{w}, \{h_i\}), \tag{4.20a}$$

$$\text{where} \qquad F(\boldsymbol{w}) = \Omega(\boldsymbol{w}) + C \sum_{i \in S_-} \max\left\{0, 1 + \max_{h_i \in \mathcal{H}} \boldsymbol{w} \cdot \phi(x_i, h_i)\right\}, \tag{4.20b}$$

$$G(\boldsymbol{w}, \{h_i\}) = C \sum_{i \in S_+} \max\left\{0, 1 - \boldsymbol{w} \cdot \phi(x_i, h_i)\right\}. \tag{4.20c}$$

Similarly, it can be shown that $L_h(\boldsymbol{w}, \{h_i\}) \geq L(\boldsymbol{w})$ and we can minimize it using the following iterative method. At iteration t, it first fixs the current parameter w^t and minimizes

w.r.t to h_i for all $i \in S_+$. This is equivalent to computing

$$h_i^{t+1} = \underset{h_i}{\operatorname{argmax}}\, \boldsymbol{w}^t \cdot \phi(x_i, h_i), \qquad \forall i \in S_+. \tag{4.21}$$

Then, it minimize w.r.t. $\boldsymbol{w}$. This is equivalent to computing

$$w_i^{t+1} = \underset{w}{\operatorname{argmin}}\, L_h(\boldsymbol{w}, h_i^{t+1}). \tag{4.22}$$

The objective, $L_h(\boldsymbol{w}, h_i^{t+1})$, is convex and can be minimized using stochastic gradient descent. It can be shown that each iteration always improves or maintains the value of the learning objective and it will converge to a local minimum. Because a large amount of negative examples are used in training, Felzenszwalb et al. [32] designs a data-mining procedure that finds a small number of "hard negative" instances and applies it to the iterative algorithm. It can be shown that by iteratively finding "hard negative" examples and using them for training, it will converge to the exact solution defined by the original large negative training set. Therefore, the minimization in (4.22) runs multiple times and hard negative examples are detected before each run. We refer readers to [32] for details.

4.6 Experiment

We perform two sets of experiments: handwritten digit recognition on MNIST and object detection on the PASCAL VOC 2007 dataset [28]. Our first set of experiments show that our approach is indeed able to prune the complexity of latent space, resulting in a model that allows significantly faster inference at test time without a drop in accuracy over a complete (non-sparse) model. Our second set of experiments show that our approach is able to learn a better model by adapting the complexity of the latent variable space to the category being trained.

Algorithm 3 Coordinate and subgradient descent algorithm

Input: $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, $\boldsymbol{w}_0$, learning rate α_0, ϵ

$t \leftarrow 0$

repeat

  for $i = 1$ to n **do** {#Optimize Over $\{h_i\}$}

    $g_i^* \leftarrow \max_{h_i \in \mathcal{H}} \boldsymbol{w}^t \cdot \phi(x_i, y_i, h_i)$, and obtain maximizer h_i^t

  end for

  $t_w \leftarrow 0$, $\boldsymbol{w}^{t_w} \leftarrow \boldsymbol{w}^t$

  repeat {#Optimize Over $\boldsymbol{w}$ with Subgradient Descent}

    for $i = 1$ to n **do** {#Can pick a random element here if Stochastic}

      $f_i \leftarrow \max_{(\hat{y}_i, \hat{h}_i) \in \mathcal{Y} \times \mathcal{H}} \boldsymbol{w}^{t_w} \cdot \phi(x_i, \hat{y}_i, \hat{h}_i) + \Delta(y_i, \hat{y}_i, \hat{h}_i)$, and

        obtain maximizer $(\hat{y}_i^*, \hat{h}_i^*)$

      $m_i \leftarrow 0$

      if $f_i - g_i^* > 0$ **then**

        $m_i \leftarrow \phi(x_i, \hat{y}_i^*, \hat{h}_i^*) - \phi(x_i, y_i, h_i^t)$

      end if

    end for

    $\nabla L_h \leftarrow \nabla \Omega(\boldsymbol{w}^{t_w}) + \frac{C}{n} \sum_{i=1}^n m_i$

    $\boldsymbol{w}^{t_w+1} \leftarrow \boldsymbol{w}^{t_w} - \alpha_t \nabla L_h$

    $t_w \leftarrow t_w + 1$

  until $|L_h(\boldsymbol{w}^{t_w+1}) - L(\boldsymbol{w}^{t_w})| < \epsilon$

  $t \leftarrow t + 1$, $\boldsymbol{w}^t \leftarrow \boldsymbol{w}^{t_w}$

until $|(L(\boldsymbol{w}^t) - L(\boldsymbol{w}^{t-1}))/L(\boldsymbol{w}^t)| < \epsilon$

4.6.1 Handwritten Digit Recognition

We now demonstrate the efficacy of our approach in the context of handwritten digit recognition. We follow closely the experimental setup of Kumar et al. [50], who proposed a LSSVM approach for this problem. Each digit is represented as a vector x of grayscale values at pixels. The goal is to predict the label of the digit, $y \in \mathcal{Y} = \{0, 1, \cdots, 9\}$. It is well known that the accuracy can be greatly improved by explicitly modeling the deformations present in each image. Kumar et al. [50] model rotations as a hidden variable taking values in a set of 11 angles uniformly distributed from -60° to 60°. We show that using our approach only a few rotations are needed to achieve the recognition accuracy of using the full set of angles.

The joint feature vector is

$$\phi(x, y, h) = [\mathbf{0}_{y(m+1)} \; ; \; \theta_h(x) \; 1 \; ; \; \mathbf{0}_{(9-y)(m+1)}]^T, \tag{4.23}$$

where $\theta_h(x)$ is the image rotated by the angle specified by h, and then strung into a vector. To adapt our approach to this framework, we let the angle $h \in \mathcal{H} = \{h_0, h_1, \cdots, h_{10}\}$, where $\mathcal{H}$ is the set of 11 angles the digit can rotate, and induce a group norm over the parameters corresponding to each angle.

We choose the MNIST dataset [51]. This dataset contains hand written digit images for digits 0 to 9. We perform binary classification on four difficult digit pairs (1-7, 2-7, 3-8, 8-9). The training data for each digit contains about 6000 images and the testing data contains approximately 1000 images. We compute exactly the same features as in Kumar et al. [50]. We use PCA to project each image to a 10 dimensional feature vector. We vary the number of angles chosen for each digit from 1 to 11. For each angle budget, we

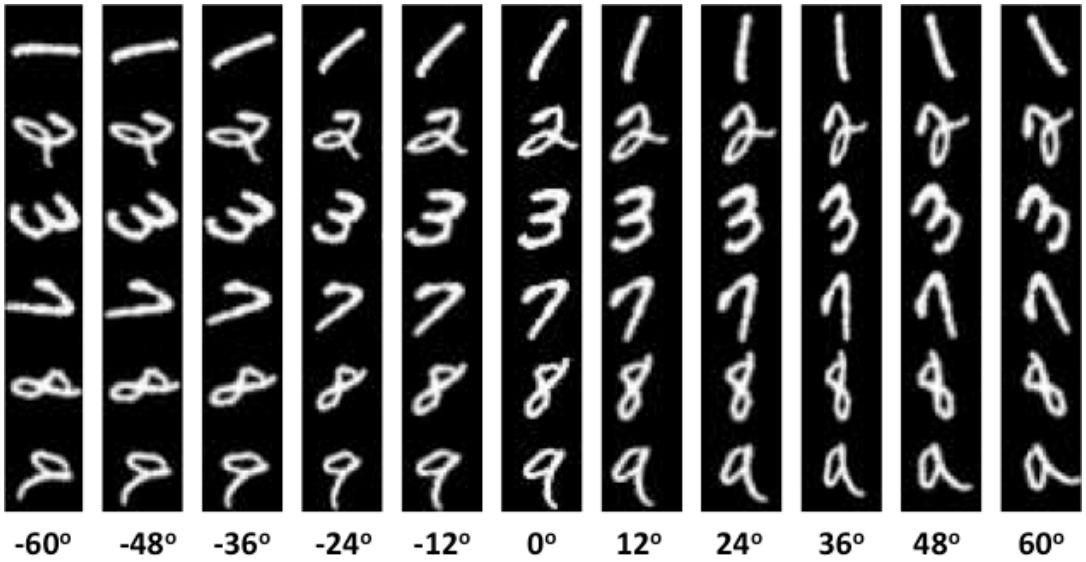


Figure 4.2: Examples of each digit and their rotations. The original images are in column corresponding to 0°(no rotation). All other columns display the images under different rotating degrees, which are uniformly sampled from [-60°, 60°]. The images are rotated counterclockwise.

select angles for a digit based on the magnitude of the ℓ_2-norm of the parameter vector corresponding to that angle. Angles with higher magnitude will be chosen first. As a baseline, we compare our approach to uniform angle selection based on the approach by Kumar et al. [50]. Given an angle budget B, we uniformly sample B angles from interval [-60°, 60°]. Figure 4.2 shows the images rotated with sampled angles when the budget is 11, where the column with degree 0 shows the original images. We use $\lambda_p = 1$ for each group in our experiment. We tried different values of C, and the prediction accuracies were fairly similar. We set $C = 1$.

Figure 4.3 shows the ℓ_2-norms of the parameter vectors for different angles in the 4 digit-pair experiments. Figure 4.4 shows how the prediction accuracy and feature computation time varies as angle budget increases. The feature computation time, which is proportional to the final prediction time, includes rotation time and PCA projection time.

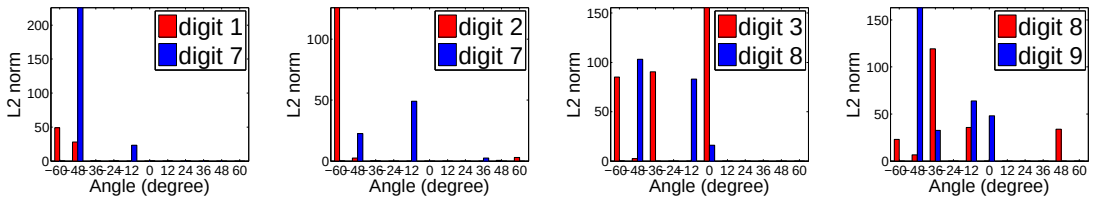


Figure 4.3: ℓ_2 norm of the parameter vectors for different angles over the 4 digit pairs.

We note a few key observations. First, in our approach, the ℓ_2-norms of the weight vector for many angles completely zero out, and only a subset of angles actually remain to contribute to the final prediction. In the end, the trained model essentially selects 5, 8, 7, and 9 angles in total for digit pairs 1-7, 2-7, 3-8, and 8-9 respectively. This is a significant reduction from the hidden space of 22 angles per digit pair using the original model in Kumar et al. [50]. Second, our approach gives very similar prediction accuracy compared to the original approach with a full set of angles. Third, due to the sparse solution of our model, using a maximum of 4 angles for each digit, we can achieve prediction accuracy similar to that using the full set of angles. Fourth, running time for feature computation increases linearly with the number of angles chosen because the time to rotate an image and perform PCA for each angle is about the same. Thus, as shown in Table 4.1, evaluation at test-time with our sparse method using maximumly 4 anlges per digit is 2.5-3 times faster than the non-sparse model without any or significant loss in accuracy.

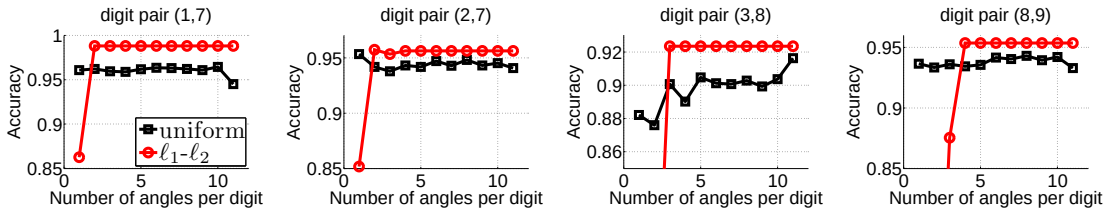


Figure 4.4: Comparison of prediction accuracy vs. angle budget (top) and test-time vs. angle budget (bottom) of our approach and uniform selection. We can see that our approach outperforms uniform selection and is able to quickly achieve accuracy comparable to the complete model (using all angles).

The Digit Pair	(1,7)	(2,7)	(3,8)	(8,9)
Our approach (4 ang. / digit)	6.5	7.0	5.4	6.0
Kumar *et al.* [50] (11 ang. / digit)	22.3	24.1	23.7	23.1

Table 4.1: Running time comparison of our approach using maximumly 4 angles per digit and original approach by Kumar *et al.* [50] using 11 angles per digit. The time is measured in seconds. The running time is based on the feature computation time of all testing digits which includes rotation time and PCA projection time. We implement the programs in MATLAB R2011a and the experiment is performed using a machine with 64-bit 8-Core Intel i7 machine with 12GB RAM.

4.6.2 Object Detection with Discriminatively Trained Deformable Part Models

We also apply our approach to discriminatively trained deformable part models [32]. In this framework, an n-component mixture of symmetric deformable part models represents each object class. Each component is a star-structured part-based model consisting of a root filter, part filters, and part displacement parameters. The score of one compo-

nent at a particular location and scale in the image is defined as the total scores from responses of the root and part filters minus the deformation cost of placing part filters in the image. The component, which gives the highest score defines the final score of the mixture model at the particular location and scale. This model is scanned across different locations and scales in the image, followed by some post-processing steps, such as bounding box prediction, non-maximal suppression, and context rescoring, to detect the actual object instances.

Each component is bilaterally symmetric and each component is grouped into a left-right symmetric pair. Thus an n-component mixture really has $2n$-members. During detection each image is matched to the component in both left and right orientation. This will allow the detector to better handle the object classes with clear left-right pose distinction, such as bicycle and car. The training data for each component is created by separating the positive training instances into different clusters and breaking down these clusters to separate left and right face examples using HoG features. The filters for the two members of a symmetric component are a flipped version of each other, so we only need to train one of them.

The mixture model can be formulated as a latent support vector machine (SVM), which the parameters of the root filter, part filters, and deformation parameters of all components are strung together to form the final parameter vector. The parameter learning proceeds in four stages. The first stage learns the parameters of root filter for each component separately. The second stage adds the flipped counter part and trains the symmetric root filters of each component independently. These two stages provide good initialization for mixture model training in subsequent stages. The third stage concatenates

the parameters of all components and learns a mixture model of root filters. The fourth stage adds the part and displacement parameters for each component and learns the final mixture model. Stages three and four use the coordinate descent approach described in Section 4.5.

We perform the experiment based on release 4 of the system [31], and use the PASCAL VOC 2007 dataset following $comp3$ protocol [28]. This dataset contains 9,963 images with annotations from 20 different object categories. In the object detection challenge, the objects are annotated with tight rectangular bounding boxes. The image set is divided into two main subsets: training/validation data (trainval) and test data (test), with the trainval data further divided into suggested training (train) and validation (val) data. The trainval data consists of 5,011 images with 12,608 annotated objects. The test data has 4,952 images with 12,032 annotated objects. Figure 4.5 shows an image from each object category and their bounding box annotation.

We apply our group norm approach to select and learn a subset of components to form the final mixture model. The motivation for doing this is our observation that as the number of components increases (from n=1 to n=6), we observe that the model trained by the original system generally overfits the training data as the number of components increases. We trained on the training and validation data, and performed testing on the test data. Figure 4.6 shows how the average precision varies with the increasing number of components, for each object category. As the number of components increases, the training accuracies always increase. However, the testing accuracies vary across different categories. For example, the test-accuracy for the "bottle" category keeps decreasing. The performance for "cat" and "horse" categories peaks at two components and then de-

creases. The performance for the "dog" category remains relatively the same when more than one component is used. Overall, as the number of components increases (from 1 to 6), the performance of the training dataset increases consistently, while the performance of the testing dataset tends to saturate or even decrease. These plots also empirically verify our hypothesis that a single setting of number of components is a suboptimal choice.

We apply the group norm to stage three of the learning process so that we can select a subset of components to do learning in stage four. Instead of using a fixed n, we consider using all of the learned components for $n = 1, 2, 3$ from stage 1 and 2 for training in stage 3 and 4. This results in a six-component mixture model in which we induce sparsity.

We set $\lambda_p = 0.12$ in equation (4.10) for all groups. We made this choice by tuning λ_p on the validation set. We vary λ_p and test on validation data (using models trained on training data alone). Table 4.2 shows the complete results for each category and the mean average precision and the average number of non-sparse components as a function of λ_p. We can see that $\lambda_p = 0.12$ gives the highest mean average precision and produces a reasonable number of non-sparse components on average. Note that in these experiment, for sufficiently high values of λ_p, sometimes all components of a model may be "sparsed out" for some object categories. For such categories, we clamp λ_p so that at least one component survives. We do not observe any such issue for experiments trained with $\lambda_p = 0.12$ on train+validation data.

Finally, bounding box prediction and context rescoring is used to further improve the detection performance. We refer the reader to [32] and [35] for details.

Figure 4.7 and 4.8 show examples of the mixture of root model before and after the training regularized by group norm. Some components whose filters are very similar to

root+part	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv	mean
$\lambda_p = 0.08$	27.8	52.2	9.8	14.0	22.7	43.9	48.9	10.9	16.9	14.7	13.3	4.1	45.3	44.0	32.3	3.9	10.2	12.5	40.2	34.4	25.1
num. comp.	3	6	5	4	3	6	4	5	6	6	5	5	6	6	6	3	1	5	6	4	4.8
$\lambda_p = 0.10$	28.0	50.4	9.6	14.0	23.8	40.3	47.0	7.3	18.0	15.3	15.3	4.6	49.0	43.1	30.8	6.7	10.2	17.3	38.9	34.2	25.2
num. comp.	4	6	3	4	2	2	4	4	6	3	4	5	5	6	6	3	1	1	6	2	3.9
$\lambda_p = 0.12$	25.2	49.3	6.8	14.0	24.0	42.3	48.1	8.8	17.6	14.3	17.6	4.5	51.5	43.7	31.7	9.9	10.2	20.5	44.5	32.4	25.8
num. comp.	2	3	2	4	1	2	4	1	6	5	2	5	4	4	6	1	1	1	5	2	3.1
$\lambda_p = 0.14$	26.6	45.5	9.9	0.7	27.3	35.4	48.5	13.0	19.1	13.5	16.8	11.2	49.3	38.7	28.5	9.4	10.2	14.9	24.0	35.3	23.9
num. comp.	1	3	2	1	1	1	4	1	6	2	1	5	5	2	6	2	1	1	1	2	2.4
$\lambda_p = 0.16$	22.8	51.8	9.4	0.7	26.3	27.8	47.4	10.5	17.4	16.3	18.6	11.0	45.3	36.7	31.4	9.8	10.2	14.9	24.2	33.0	23.3
num. comp.	1	5	1	1	1	1	4	1	2	2	1	4	3	2	5	1	1	1	1	2	2.0

Table 4.2: Results on VOC2007 validation data (using models trained on training data alone). Only root+part model accuracies are shown (no bounding box prediction or context rescoring is performed). We can see that $\lambda_p = 0.12$ achieves the highest mean average precision and with 3.1 non-sparse components on average (out of 6).

those in other components are removed by group norm. For example, for bus category, the filters of the bottom three components resemble those in the top three components. Their filters are trained to be sparse and are removed from the training of part filters and displacement parameters. We observe similar situations for the car category. We compare our approach to the original system in [31] with varying number of components when $n = 1, 2, 3, 4, 5,$ and 6. We also compare to the case when the group norm is not used in our approach, *i.e.*, λ_p is set to be 0 in (4.10). Table 4.3 summarizes the results. The mean accuracies for root+part initially increase from $n = 1$ to $n = 2$, but then stagnate. Our approach, on the other hand, is able to pick out a fairly non-uniform sparsity pattern (and thus the mixture size) across the categories, performing the best in 7 out of 20 categories with a mean average precision of 32.5. This is better than all other settings in which n

is fixed. In addition, when the group norm is not used in our case, the mean average precision is only 31.7 and we perform better in 14 out of 20 categories. On average, 3.6 out of 6 components remains in the final models. This is significantly lower than the original 6 components for each category. As is standard in this task, we also ran the bounding box prediction and context rescoring steps, the results for which are reported for the sake of completeness. Figure 4.9 shows some example detections produced by our approach.

4.7 Summary

We address the problem of estimating the parameters of a latent variable model as well as discovering meaningful states for the latent variables. This allows us to control the model complexity and potentially speed up inference time and learn a more reliable model. We address this problem in the context of SVMs with latent variables and use an ℓ_1-ℓ_2 norm regularization. Our experiments on handwritten digit recognition show that our approach is able to effectively reduce the size of latent variable state space and thus reduce the inference time with no loss of accuracy compared to using the full latent state space. Our experiment on object detection shows that we are able to control the number of components used in the mixture model and learn a better object detector.

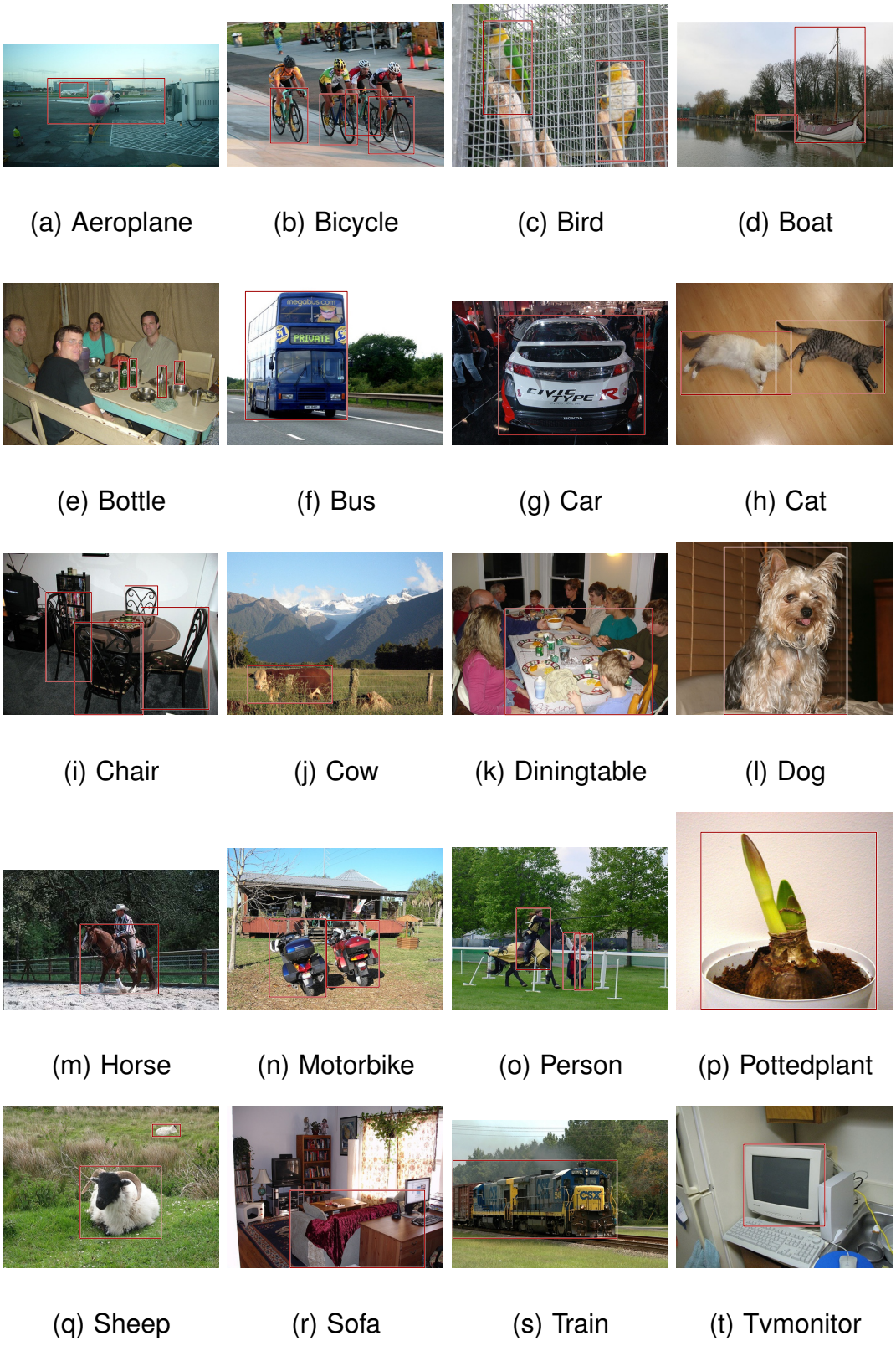


Figure 4.5: Example images from each category in PASCAL VOC 2007 dataset. The red bounding boxes are the annotated ground truth bounding boxes.

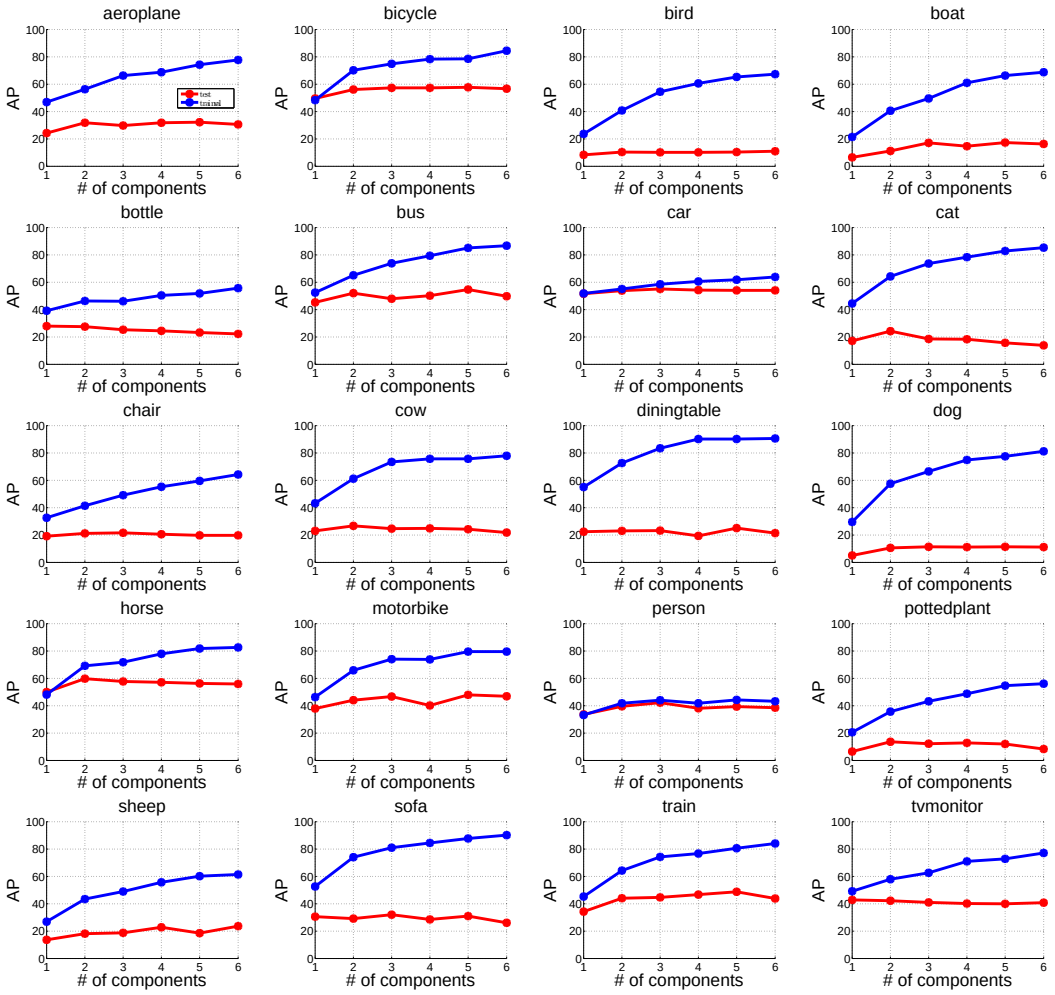


Figure 4.6: Behaviors of the Felzenszwalb *et al*. detector [31] as a function of the number of components for each object category on VOC2007 dataset. The models are trained on the training+validation data and tested in the test data. Only root+part accuracies are shown (no bounding box prediction or context rescoring is performed). We can see that as the number of components increases (from 1 to 6), the accuracies on train+val increase consistently for all categories, while those on test tends to saturate or even decrease for many categories. This suggests overfitting occurs as the number of components increases.

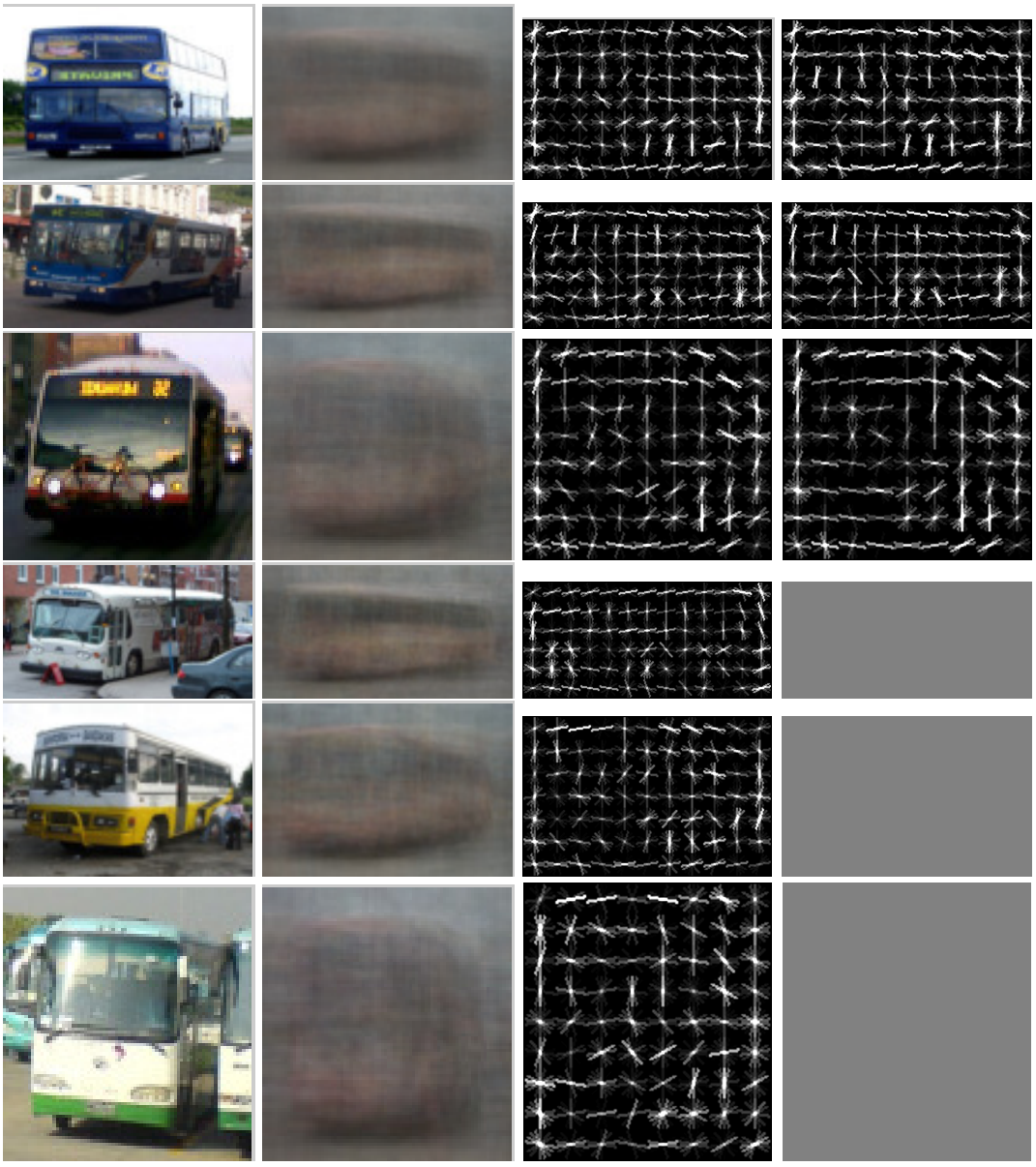


Figure 4.7: Mixture of root model before and after the training regularized by group norm for bus category. Each row shows the information for a component and its symmetric counterparts are not shown to save space. Starting from the left, the first column shows a positive example from the cluster which produces the component. The second column shows the average image of the cluster. The third column shows the root filter of the component before the training. The last column shows the root filter after training. A complete gray image indicates the filter is sparse and will be removed from training the part filters and displacement parameters.

Figure 4.8: Mixture of root model before and after the training regularized by group norm for car category. Each row shows the information for a component and its symmetric counterparts are not shown to save space. Starting from the left, the first column shows a positive example from the cluster which produces the component. The second column shows the average image of the cluster. The third column shows the root filter of the component before the training. The last column shows the root filter after training. A complete gray image indicates the filter is sparse and will be removed from training the part filters and displacement parameters.

root+part	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv	mean
n=1	24.3	49.5	8.2	6.5	27.8	45.2	51.5	17.0	19.2	22.9	22.2	5.1	49.8	37.9	33.6	6.5	13.6	30.4	34.2	42.7	27.4
n=2	31.7	56.0	10.3	11.2	27.5	52.0	53.8	**24.2**	21.1	26.6	22.9	10.6	**59.6**	44.1	39.5	13.6	18.1	29.1	44.0	42.0	31.9
n=3	29.6	57.3	10.1	17.1	25.2	47.8	**55.0**	18.4	21.6	24.7	23.3	11.2	57.6	46.5	**42.1**	12.2	18.6	31.9	44.5	40.9	31.8
n=4	31.8	57.2	10.1	14.7	24.3	50.0	54.1	18.2	20.4	24.8	19.3	11.0	57.0	40.2	38.1	12.8	22.8	28.4	46.6	40.0	31.1
n=5	32.2	**57.6**	10.4	**17.2**	23.2	**54.5**	54.0	15.6	19.6	24.2	25.1	11.3	56.2	**47.8**	39.3	12.0	18.5	30.9	**48.7**	39.8	31.9
n=6	30.5	56.7	**11.0**	16.2	22.1	49.7	54.1	13.9	19.6	21.7	21.4	11.2	55.7	46.8	38.5	8.3	**23.6**	26.0	43.9	40.8	30.6
ours ($\lambda_p = 0$)	**34.0**	54.3	6.2	13.9	24.4	49.3	54.5	22.8	20.1	25.6	26.9	9.1	56.8	45.1	39.4	13.3	18.2	**34.8**	43.4	42.3	31.7
ours ($\lambda_p = 0.12$)	30.8	56.3	9.4	15.5	**28.4**	52.4	54.5	19.8	**21.9**	**30.1**	**28.0**	**11.3**	57.1	45.7	38.6	**14.7**	15.4	31.8	44.5	**43.2**	**32.5**
comp. num.	4	6	1	2	2	3	4	5	2	4	3	5	4	4	6	4	1	2	6	3	3.6
bbox	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv	mean
n=1	25.0	50.2	8.3	6.5	28.3	45.6	54.5	17.2	19.6	22.8	23.1	5.2	50.4	37.9	33.8	6.8	14.9	31.1	35.0	**44.0**	28.0
n=2	31.0	58.5	10.2	10.7	27.4	52.6	56.2	**26.1**	21.5	26.6	22.3	10.8	**61.6**	45.3	40.0	13.5	17.9	29.9	45.4	42.6	32.5
n=3	28.9	59.5	10.0	15.2	25.5	49.6	**57.9**	19.3	**22.4**	25.2	23.3	11.1	56.8	46.6	**41.9**	12.2	17.8	**33.6**	45.1	41.6	32.2
n=4	32.1	**59.6**	10.2	15.3	24.6	51.9	57.5	18.5	20.2	25.1	17.3	11.0	57.4	43.0	36.6	12.5	22.5	28.0	46.6	41.7	31.6
n=5	31.0	58.5	10.4	**17.7**	23.5	**54.9**	57.6	17.3	19.5	22.6	24.7	11.1	57.6	**49.2**	39.8	11.6	18.4	32.3	**47.1**	40.8	32.3
n=6	29.6	56.1	**10.9**	15.3	21.8	50.5	57.1	15.3	20.2	19.8	21.0	11.4	55.7	45.5	38.5	10.3	**23.6**	25.4	42.6	41.6	30.6
ours ($\lambda_p = 0$)	**33.9**	54.7	5.6	13.5	23.0	47.3	55.9	23.2	20.6	23.9	26.1	9.1	57.7	44.4	39.8	13.4	10.3	31.7	43.5	41.7	31.0
ours ($\lambda_p = 0.12$)	33.6	57.6	9.4	15.5	**28.9**	51.7	55.3	20.2	22.1	**30.4**	**28.9**	**11.5**	58.1	46.4	38.8	**14.1**	16.2	32.3	45.6	43.8	**33.0**
context	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv	mean
n=1	27.9	52.0	11.2	9.8	**29.7**	47.7	56.3	23.6	20.8	25.0	26.4	13.3	53.4	42.2	35.5	9.0	15.7	34.4	38.3	**45.8**	30.9
n=2	33.2	60.5	12.5	10.8	28.6	52.7	58.0	**30.9**	22.9	28.7	26.3	13.2	**65.3**	47.6	42.6	15.6	19.9	33.1	49.3	44.2	34.8
n=3	31.1	**61.6**	11.9	17.3	27.1	49.0	**59.6**	22.9	23.0	26.7	24.5	12.9	60.2	49.5	**43.2**	13.5	18.9	**36.4**	49.2	43.0	34.1
n=4	35.2	59.4	12.4	16.7	25.4	51.9	59.1	20.2	21.3	26.3	18.8	12.7	60.3	46.2	38.3	14.5	21.4	30.6	**51.1**	43.1	33.2
n=5	32.6	58.4	**12.7**	17.9	25.1	**55.8**	58.9	17.5	19.9	24.7	21.4	13.2	61.0	**51.6**	42.4	13.5	19.4	32.8	49.8	41.5	33.5
n=6	30.8	57.1	12.1	15.7	21.6	49.1	57.7	19.1	20.6	22.3	18.8	13.3	58.3	47.1	41.4	12.0	**22.9**	28.0	43.0	42.2	31.7
ours ($\lambda_p = 0$)	35.5	54.9	8.1	15.4	23.0	44.3	57.3	26.3	22.1	25.1	28.3	12.0	60.7	45.9	42.2	15.3	10.4	35.3	45.6	42.7	32.5
ours ($\lambda_p = 0.12$)	**35.8**	59.9	10.3	**18.1**	29.3	53.4	56.3	25.3	**23.5**	**30.6**	**31.0**	**13.9**	60.5	48.9	41.1	**16.1**	17.3	35.3	49.5	45.7	**35.1**

Table 4.3: VOC2007. Testing on testing data. Evaluation is performed based on 1) root and part (root+part) filters, 2) bounding box (bbox) prediction, and 3) context rescoring. Cases with the highest average precision is in bold font. The last row in the root+part table shows the number of non-sparse components out of a total of 6 components being trained.

Figure 4.9: Examples top detections for different object categories using root-plus-part models based on our approach. Within each image, the red bounding box is the predicted location of the root filter and the blue bounding boxes are the predicted locations of the part filters.

Chapter 5

Conclusions and Future Work

In this thesis, we broadly examine the problem of visual computing under resource constraints. Specifically, we consider constraints of human resources and computational resources. Chapter 2 describes our approach to the first constrained setting with a frame retrieval problem in a camera network. We first map video frames in a camera network onto a graphical model. We then perform collective classification and active inference to produce more effective video analysis in camera networks. Chapter 3 describes the work to direct a more resource-consuming algorithm with a less resource-consuming algorithm. We design a new algorithm that uses this model to determine where in a video to apply the expensive algorithm. We modify an algorithm by Krause and Guestrin [48] so that it can be run efficiently over a Markov chain with thousands of nodes. Our final contribution is to experimentally demonstrate the value of this algorithm in two vision tasks: motion and face detection. Chapter 4 describes the work for latent variable state learning. We propose the use of structured sparsity inducing norms like ℓ_1-ℓ_2 to estimate the parameters of a latent-variable model, thereby regularizing the complexity of the latent space. We apply our approach to the Latent SVM model, for both the binary and structured output case. We demonstrate the effectiveness of our approach on two sets of experiments: handwritten digit recognition on MNIST and object detection on the PASCAL VOC 2007 dataset [28]. There are many questions and future work and we summarize them as follows.

5.1 New Reward Functions for Value of Information in Graphical Models

In Chapter 2, we considered using the value of information [48] to formulate the reward function to perform active inference. We propose a new type of reward function based on joint probabilities. This reward function is based on energy functions on graphical models, which occur in a variety of problems in early computer vision, such as stereo matching and image restoration.

5.1.1 Problem Formulation

We formulate the problem based on similar notations and ideas in the problem formulation of Krause and Guestrin [48]. We will assume that the state of the world is described by a collection of random variables $X_V = (X_1, \cdots, X_n)$, where V is the index set. For example, V could denote a set of pixel locations, and X_i model the disparity value for a pixel at location $i \in V$. Let x_i be a realization for X_i, and $x_v = (x_1, \cdots, x_n)$ be a realization for all random variables X_V. Let $\overline{X_V}$ be the set of all possible realizations for X_V. For a subset $A = \{i_1, \cdots, i_k\} \subseteq V$, we use the notation X_A to refer to the random vector $X_A = (X_{i_1}, \cdots, X_{i_k})$. We assume that the variables X_V are discrete. We take a Bayesian approach, and assume a prior probability distribution $P(X_V)$ over the outcomes of the variables. Suppose we select a subset of variables, X_A (for $A \subseteq V$), and observe $X_A = x_A$. For example, A is the set of locations where we observe pixel labels, or a set of video frames in which we detect anomaly. After observing a realization of these variables $X_A = x_A$, we can compute the posterior distribution over all variables $P(X_V | X_A = x_A)$.

Based on this posterior probability we obtain a reward $R(P(X_V|X_A = x_A))$. For example, this reward function could depend on the uncertainty (e.g. measured by entropy) of the distribution $P(X_V|X_A = x_A)$.

In general, when selecting observation, we will not know ahead of time what observations we will make. Instead, we only have a distribution over the possible observations. Hence, we will be interested in the expected reward, where we take the expectation over the possible observations. We can also consider maximum or minimum reward over the possible observations. We refer to them as maximum or minimum reward respectively.

When optimizing the selection of variables, we can consider different settings. In subset selection, our goal is to pick a subset $A^* \subseteq V$ of variables, maximizing

$$A^* = \arg\max_A R(X_V, X_A), \tag{5.1}$$

where

$$R(X_V, X_A) =$$
$$\begin{cases} R_{exp}(X_V, X_A) = \sum_{x_A} P(X_A = x_A) R(P(X_V|X_A = x_A)) & \text{if expected reward} \\ R_{max}(X_V, X_A) = \max_{x_A} P(X_A = x_A) R(P(X_V|X_A = x_A)) & \text{if maximun reward} \\ R_{min}(X_V, X_A) = \min_{x_A} P(X_A = x_A) R(P(X_V|X_A = x_A)) & \text{if minimun reward} \end{cases} \tag{5.2}$$

We impose some constraints on the set A we are allowed to pick (e.g., on the number of variables that can be selected, etc.) In the subset selection setting, we commit to the selection of the variables before we get to see their realization.

Instead, we can also sequentially select one variable after the other, letting our choice depend on the observations made in the past. In this setting, we would like to find

a conditional plan π^* that maximizes

$$\pi^* = \arg\max_{\pi} R(X_V, \pi), \tag{5.3}$$

where

$$R(X_V, \pi) = \begin{cases} R_{exp}(X_V, \pi) = \sum_{x_V} P(x_V) R(P(X_V | X_{\pi(x_V)} = x_{\pi(x_V)})) & \text{if expected reward} \\ R_{max}(X_V, \pi) = \max_{x_V} P(x_V) R(P(X_V | X_{\pi(x_V)} = x_{\pi(x_V)})) & \text{if maximum reward} \\ R_{min}(X_V, \pi) = \min_{x_V} P(x_V) R(P(X_V | X_{\pi(x_V)} = x_{\pi(x_V)})) & \text{if minimum reward} \end{cases} \tag{5.4}$$

Hereby, π is a conditional plan that can select a different set of variables for each possible state of the world x_V. We use the notation $\pi(x_V) \subseteq V$ to refer to the subset of variables selected by the conditional plan π in state $X_V = x_V$.

Using the expected reward, Problems (5.1) and (5.3) are referred to by Krause and Guestrin [48] as the problems of optimizing value of information. With expected reward, Krause and Guestrin [48] show that optimizing the value of information is wildly intractable (NP^{PP}-complete) even for a probability distribution for which efficient inference is possible, including Naive Bayes models and discrete polytrees. Therefore, they [48] consider chain graphical models. They use a class of local reward functions R_j, which are defined on the marginal probability distribution of the variables X_i. This class has the computational advantage that local rewards can be evaluated using probabilistic inference techniques. Furthermore, it is decomposable along the chain conditioned on observations. The total reward will then be the sum of all local rewards. They design efficient dynamic programming algorithms in chain graphical models based on such local

rewards for the above two problems.

The local reward is built using marginal probabilities. However, in many early vision problems, such as stereo matching, MAP inference is considered. Optimization is based on energy minimization to maximize joint probability over all variables. We therefore consider total reward based on joint probability and hope to establish its relationship to the energy minimization problem. We next introduce notations and briefly describe MAP inference in undirected graphical models based on energy minimization.

5.1.2 MAP inference in Undirected Graphical Models

A graph consist of a set of vertices and associated edges, which is the set of unordered pairs of vertices. A graphical model is a collection of random variables indexed by the vertex of a graph. In our case, X_i corresponds to a vertex in the graph for all possible $i \in V$. For notational convenience, we index all vertices in the graph by V, and let $E = \{(i,j)|i,j \in V\}$ be the set of edges in the graph. We define nonnegative potential functions $\phi(X_i = x_i)$ for all $i \in V$ and $\psi(X_i = x_i, Xj = x_j)$ for all pairs of unordered vertices defined by the edge set. The joint probability for a realization x_V of X_V is computed as follows:

$$P(X_V = x_V) = \frac{W(x_V)}{Z} \tag{5.5}$$

where

$$W(x_V) = \prod_{i \in V} \phi(X_i = x_{V(i)}) \prod_{(i,j) \in E} \psi(X_i = x_{V(i)}, Xj = x_{V(j)}), \tag{5.6}$$

where $x_{v(i)}$ is the realization for X_i in x_v, and Z is referred to as the partition function, defined as

$$Z = \sum_{x_V} W(x_V). \tag{5.7}$$

Note that x_v is collection of all possible realizations for all random variables, and define a unique realization for each X_i. The MAP inference problem wants to find a x_v such that $P(x_v)$ is maximized.

We define $\phi(X_i = x_i) = exp(-\theta_i(x_i)))$ and $\psi(X_i = x_i, Xj = x_j) = exp(-\theta_{ij}(x_i, x_j))$, where $\theta_i(x_i)$ and $\theta_{ij}(x_i, x_j)$ are nonnegative values modeling the energy for both types of potential functions. Then,

$$W(x_V) = exp(-(\sum_{i \in V} \theta_i(x_{V(i)}) + \sum_{(i,j) \in E} \theta_{ij}(x_{V(i)}, x_{V(j)}))), \tag{5.8}$$

Therefore, the MAP inference problem becomes

$$x_V^* = \arg\min_{x_V} (\sum_{i \in V} \theta_i(x_{V(i)}) + \sum_{(i,j) \in E} \theta_{ij}(x_{V(i)}, x_{V(j)})) \tag{5.9}$$

This problem can be solved exactly for tree graphical models using belief propagation [70]. For more general graphs, exact inference is NP-hard. Various approximation algorithms, such as loopy belief propagation, graph cut, and tree-reweighted belief propagation, have been designed. We want to build relationships between these energy functionals and reward functions for optimizing the value of information problem. We consider a class of reward function such that $a \cdot R(f) = R(a \cdot f)$ for any functional f and any scalar a. Without being strict on the definition, we refer to this as the scale invariant property of the reward function. We next describe how the subset selection problem can be reformulated by this new class of reward function. For conditional plan, we are not able to work out a closed form formulation currently.

5.1.3 Subset Selection

We first consider the subset selection problem. We consider $R(P(X_V|X_A = x_A)) = \max(P(X_V|X_A = x_A))$. Since the reward function is scale invariant, we therefore have $P(X_A = x_A)R(P(X_V|X_A = x_A)) = R(P(X_A = x_A)P(X_V|X_A = x_A)) = R(P(X_V, X_A = x_A))$. In addition, since $A \subseteq V$, $P(X_V = x_V, X_A = x_A) = 0$ if $X_V = x_V$ conflict with $X_A = x_A$. We use $x_{x_{(V,A)}}$ to denote a valid realization of X_V with $X_A = x_A$ such that x_V does not conflict with x_A, and $X_{x_{(V,A)}}$ to denote the set of all such valid realizations.

Without providing rigorous proof, $R_{max}(X_V, X_A) = \max_{x_V}(P(X_V = x_V))$, and all choices of A share the same reward value. In addition, $R_{min}(X_V, X_A) \leq \max_{x_V}(P(X_V = x_V))$, and when A is the empty set, the equality holds. This means that the choice of A which maximizes $R_{min}(X_V, X_A)$ is the empty set. These two objective functions are not valid for optimization. Finally, $R_{exp}(X_V, X_A) \geq \max_{x_V}(P(X_V = x_V))$. Futhermore, $R_{exp}(X_V, X_{A_1}) > R_{exp}(X_V, X_{A_2})$, for $A_1, A_2 \subset V$ and $A_2 \subset A_1$. Denote the maximum cost for observation as B, which is also referred to as the budget for observation. This, then, means a larger B can always produces a maximum expected reward which is not smaller than that of a lower B. This can be achieved by adding additional variables to the subset which maximize expected reward of a smaller B if the budget permits. We therefore can use this objective function for optimization.

Based on equation (5.7), we know that the value of the partition function is inde-

pendent of the actual realization of X_V. We therefore have

$$P(X_A = x_A)R(P(X_V|X_A = x_A)) = R(P(x_{x(V,A)})) = R(\frac{W(X_{x(V,A)})}{Z}) \propto R(W(X_{x(V,A)})) \tag{5.10}$$

Objective function (5.2) for subset selection can thus be formulated as

$$R_{exp}(X_V, X_A) = \sum_{x_A} P(X_A = x_A)R(P(X_V|X_A = x_A)) \propto \sum_{x_A} R(W(X_{x_{(V,A)}})) \tag{5.11}$$

In addition, different choices of A share the same partition function value Z. Therefore, $A^* = \arg\max_A R_{exp}(X_V, X_A)$ is the same as $A^* = \arg\max_A \sum_{x_A} R(W(X_{x_{(V,A)}}))$. We can redefine

$$R_{exp}(X_V, X_A) = \sum_{x_A} R(W(X_{x_{(V,A)}})) \tag{5.12}$$

Finally, the subset selection problem becomes

$$A^* = \underset{A \subset V, \beta(A) \leq B}{\operatorname{argmax}} R_{exp}(X_V, X_A) \tag{5.13}$$

where, $\beta(A)$ is the cost of observing A and B is the total budget for the observation. The remaining questions are: 1) does this problem has an efficient solution over a chain graphical model, and 2) does this problem has an efficient solution over a more general graphical model? For either question, if the answer is positive, we need to design the actual algorithm that solves this problem. If not, we need to give a reason, and then possibly provide some approximation algorithm to the problem.

5.2 Future Work for Learning SVMs with Latent Variables Using Structured Norms

We mention three main directions of future work. Section 5.2.1 describes the future work for the object detection experiment using the idea of subcategories. Section 5.2.2 describes the possibility of learning the complexity of the structured state space. Finally, Section 5.2.3 describes the challenges of working on the dual forms of the learning problem of SVMs with latent variables.

5.2.1 Object Detection with Deformable Part Models Using Subcategories

In the object detection experiment in Section 4.6.2, each component corresponds to its own set of positive training examples. This requires a partition of the positive training examples. In Felzenszwalb et al. [32], they cluster the positive examples based on their aspect ratios and produce left-right flipped subclusters using appearance (HoG) features. Divvala et al. [24] notices that by only using appearance-based clustering, considerable improvement in performance is obtained. In addition, they observe that with the new components, the part deformations can be turned off, yet obtaining results that are almost on par with the original object detectors using deformable part models. Interestingly, with only root filters, as they increase the number of components, the mean average precision on VOC2007 dataset does tend to saturate and stabilizes around 15 components. Futhermore, different categories behave differently. For instance, the performance of boat and train stabilizes around 15 components, while the performance of tvmonitor peaks at 25

components and then decreases [23]. This is similar to the behavior of the origial detection system as the component number increases in Section 4.6.2 and suggests that each catergory has its own optimal number of components. Our group norm approach can be directly applied in this scenario.

5.2.2 Structured State Space

Chapter 4 described how to use group norm to learn the latent variable state space in latent SVMs. As described in Section 4.4, we consider that latent variables are independent and directly map to all the states into the joint feature vector. However, when the latent state space is not independent, it can be impractical to map the total number of possible states each into the joint feature vector. For example, Vedaldi and Zisserman [95] use structured output regression with latent variables to do object category detection. To handle partially occluded or truncated objects, they introduce a vector z of binary latent variables that encode the parts of the object that are visible. To do this, an image is decomposed into rectangular cells and each variable in z corresponds to one such cell. To learn the state space of z, we should not treat each variable in z as independent, because nearby cells are correlated. A state of z consists of the visibility value of each cell in the image. The number of possible states is 2^N, where N is the number of cells in the image. Consider a typical image of size 512×512 where the cell size is 16×16. The number of cells is 32. The state size of z is therefore $2^{32} = 4,294,967,296$. Each state has its own feature, and the size of the joint feature vector will be larger than 4 billion and the size will be similar for the model parameter vector. Therefore, in the case of such structured

state spaces, certain technique must be designed to make the size of the joint feature vector tractable, and this makes the parameter vector practical to train. For example, we can define some meaningful distance metric among states and cluster the states into different groups. We can treat these groups as new states and apply the old technique to learn them. A proper method to handle this issue is subject to future research.

5.2.3 The Dual Form of Learning SVMs with Latent Variables Regularized by Group Norms

Problem (4.3) and (4.5) in Chapter 4 are formulated as primal forms. For SVMs with ℓ_2 regularzed norm, it is well-known that we can work on the dual form of the problem and use kernel methods to train a nonlinear classifier [12]. We therefore consider the possibility of working on the dual form of our problems, hoping that the kernel method can help us build a better model. However, we encounter two challenges: 1) our training objective is not convex and it is not obvious that we can derive a dual problem which has exactly the same solutions as the primal problem, and 2) we use group norm as the regularizer and this creates another difficulty to derive dual problem in closed form.

To handle the first challenge, Yang et al. [108] propose an iterative algorithm for the dual problem of latent SVMs regularized by ℓ_2 norm. They notice that if the latent configuration for all positive examples are fixed (or observed), then the learning objective is convex. We can derive the kernelized dual form of this new learning objective that can be solved using standard dual solver in regular SVMs. When the latent configuration for positive examples are indeed observed, for example, we have the ground-truth bounding

boxes of the objects in an object detection task, we can directly use this method to estimate the parameters. When the latent configuration for positive examples are unknown, they consider finding a configuration so that the resultant nonlinear decision function from the dual form separates the two classes as widely as possible. In other words, they look for a set of latent configurations for positive examples that can maximize the SVM margin. This is equivalent to minimizing the dual objective w.r.t. the positive examples' latent configuration with the lagrangian multipliers in the function fixed. They thus introduce an iterative algorithm. In each iteration, they first use a co-ordinate ascent algorithm to minimize the dual objective function w.r.t. to the latent configuration of positive examples. They then maximize the dual function w.r.t. the lagrangian multipliers using the standard dual solver. Please refer to Yang et al. [108] for details of their approach.

Problem (4.5) differs from the problem in Yang et al. [108] in that we choose group norm as the regularizer. We thus mimic their approach and try to solve the dual form of our problem. In particular, we only consider ℓ_1-ℓ_2 norm as the regularizer. However, we are not able to derive a closed form dual objective function for both problem (4.3) and (4.5). We provide our derivation and analysis for problem (4.5). The issue in problem (4.3) is similar and we omit the description.

First, assume the latent configuration for all positive examples are fixed and denote such latent configuration as h_j^* for each $j \in S_+$, where S_+ is the set of positive examples. Let S_- be the set of negative examples. We rewrite problem (4.5) as follows:

$$\min_{\boldsymbol{w}} \quad \Omega(\boldsymbol{w}) + C \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \xi_{i,h} + C \sum_{j \in S_+} \xi_j, \tag{5.14}$$

$$s.t. \quad -\boldsymbol{w} \cdot \phi(x_i, h) \geq 1 - \xi_{i,h}, \xi_{i,h} \geq 0 \qquad \forall i \in S_-, \forall h \in \mathcal{H} \tag{5.15}$$

$$\boldsymbol{w} \cdot \phi(x_j, h_j^*) \geq 1 - \xi_j, \xi_j \geq 0 \qquad \forall j \in S_+ \tag{5.16}$$

For simplicity of derivation, we consider

$$\Omega(\boldsymbol{w}) = \frac{1}{2} \sum_{p=1}^{P} \lambda_p \left\| \boldsymbol{w}^p \right\|_2 . \tag{5.17}$$

This objective is convex and the corresponding Lagrangian of the problem is given by

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = & \frac{1}{2} \sum_{p=1}^{P} \lambda_p \left\| w^p \right\|_2 + C \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \xi_{i,h} + C \sum_{j \in S_+} \xi_j - \\ & \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \alpha_{i,h} \left[-\boldsymbol{w} \cdot \phi(x_i, h) + \xi_{i,h} - 1 \right] - \\ & \sum_{j \in S_+} \beta_j \left[\boldsymbol{w} \cdot \phi(x_j, h_j^*) + \xi_j - 1 \right] - \\ & \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \gamma_{i,h} \xi_{i,h} - \sum_{j \in S_+} \mu_j \xi_j \end{aligned} \tag{5.18}$$

where $\{\alpha_{i,h}\}, \{\beta_j\}, \{\gamma_{i,h}\}$, and $\{\mu_j\}$ are Lagrange multipliers. The corresponding set of KKT conditions are given by

$$\alpha_{i,h} \geq 0, \tag{5.19a}$$

$$-\boldsymbol{w} \cdot \phi(x_i, h) + \xi_{i,h} - 1 \geq 0, \tag{5.19b}$$

$$\alpha_{i,h} \left[-\boldsymbol{w} \cdot \phi(x_i, h) + \xi_{i,h} - 1 \right] = 0, \tag{5.19c}$$

$$\beta_j \geq 0, \tag{5.19d}$$

$$\boldsymbol{w} \cdot \phi(x_j, h_j^*) + \xi_j - 1 \geq 0, \tag{5.19e}$$

$$\beta_j \left[\boldsymbol{w} \cdot \phi(x_j, h_j^*) + \xi_j - 1 \right] = 0, \tag{5.19f}$$

$$\gamma_{i,h} \geq 0, \tag{5.19g}$$

$$\xi_{i,h} \geq 0, \tag{5.19h}$$

$$\gamma_{i,h}\xi_{i,h} = 0, \tag{5.19i}$$

$$\mu_j \geq 0, \tag{5.19j}$$

$$\xi_j \geq 0, \tag{5.19k}$$

$$\mu_j\xi_j = 0, \tag{5.19l}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0, \tag{5.19m}$$

$$\frac{\partial L}{\partial \xi_{i,h}} = 0, \tag{5.19n}$$

$$\frac{\partial L}{\partial \xi_j} = 0, \tag{5.19o}$$

$$\forall i \in S_-, h \in \mathcal{H}, \forall j \in S_+. \tag{5.19p}$$

Our goal is to express the objective L in terms of Lagrange multipliers, which we need to optimize out $\boldsymbol{w}$, $\{\xi_{i,h}\}$, and $\{\xi_j\}$. We know

$$\frac{\partial L}{\partial \xi_{i,h}} = 0 \Rightarrow C - \alpha_{i,h} - \gamma_{i,h} = 0 \tag{5.20}$$

$$\frac{\partial L}{\partial \xi_j} = 0 \Rightarrow C - \beta_j - \mu_j = 0 \tag{5.21}$$

From equation (5.19m), we have the following formulas:

$$\boldsymbol{w} = Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) * R(\boldsymbol{w}) \tag{5.22}$$

where

$$Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{j \in S_+} \beta_j \phi(x_j, h_j^*) - \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \alpha_{i,h} \phi(x_i, h) \tag{5.23}$$

$$R(\boldsymbol{w}) = \left[\frac{||\boldsymbol{w}^1||_2}{\lambda_1}, \cdots, \frac{||\boldsymbol{w}^1||_2}{\lambda_1}, \cdots, \frac{||\boldsymbol{w}^P||_2}{\lambda_P}, \cdots, \frac{||\boldsymbol{w}^P||_2}{\lambda_P} \right]^T, \tag{5.24}$$

and $*$ is element-wise multiplication of two vectors. Let $R(\boldsymbol{w})^{*-1}$ be the element-wise inverse of $R(\boldsymbol{w})$. (When $||\boldsymbol{w}^p||_2 = 0$ for a group p, the inverse is not defined, and we let

the corresponding entry in $R(\boldsymbol{w})^{*-1}$ be 0.) Then we also have

$$R(\boldsymbol{w})^{*-1} * \boldsymbol{w} = Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.25}$$

$$\boldsymbol{w}^T \cdot \left[R(\boldsymbol{w})^{*-1} * \boldsymbol{w}\right] = \sum_{p=1}^{P} \lambda_p \left|\left|\boldsymbol{w}^p\right|\right|_2 \tag{5.26}$$

Based on equations (5.19a), (5.19d), (5.19g), (5.19j), (5.20), (5.21), (5.22), (5.25), and (5.26), we obtain the the dual Lagrangian for equation (5.18) in the form.

$$L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i \in S_-} \sum_{h \in \mathcal{H}} \alpha_{i,h} + \sum_{j \in S_+} \beta_j - \frac{1}{2} \left[Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) * R(\boldsymbol{w})\right]^T \cdot Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.27}$$

where $0 \leq \alpha_{i,h} \leq C$ and $0 \leq \beta_j \leq C$ for all $i \in S_-$, $h \in \mathcal{H}$, and $j \in S_+$.

Because of the existence of $R(\boldsymbol{w})$, it is not clear how to use kernel to express this dual function. In addition, it is not clear how to express $R(\boldsymbol{w})$ in terms of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Here we provide some discussion for $R(\boldsymbol{w})$.

To express $R(\boldsymbol{w})$ in terms of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we need to express $||\boldsymbol{w}^p||_2$ for each group p in terms of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Based on equation (5.22), for a specific group p, we have

$$\begin{aligned}
(w_1^p)^2 &= \frac{(Q_1^p)^2}{\lambda_p^2} ||\boldsymbol{w}^p||_2^2 = \frac{(Q_1^p)^2}{\lambda_p^2} \left[(w_1^p)^2 + \cdots + (w_{n_p}^p)^2\right], \\
&\vdots \\
(w_{n_p}^p)^2 &= \frac{(Q_{n_p}^p)^2}{\lambda_p^2} ||\boldsymbol{w}^p||_2^2 = \frac{(Q_{n_p}^p)^2}{\lambda_p^2} \left[(w_1^p)^2 + \cdots + (w_{n_p}^p)^2\right].
\end{aligned}$$

where w_n^p is the n^{th} entry of $\boldsymbol{w}$ in group p, Q_n^p is the corresponding n^{th} entry of $Q(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in group p, and n_p is the size of group p. If we let

$$\mathbf{a} = \left[(w_1^p)^2, \cdots, (w_{n_p}^p)^2\right]^T, \tag{5.28}$$

$$\mathbf{b} = \left[\frac{(Q_1^p)^2}{\lambda_p^2}, \cdots, \frac{(Q_{n_p}^p)^2}{\lambda_p^2}\right]^T, \tag{5.29}$$

$$M = \underbrace{[\mathbf{b}, \cdots, \mathbf{b}]}_{n_p \text{ columns}} \tag{5.30}$$

Then, we have $\mathbf{a} = M\mathbf{a}$. Then we know that $\mathbf{a}$, which is the element-wise square of the parameter vector of group p, is either a zero vector or an eigenvector of M corresponding to eigenvalue 1 if 1 is indeed an eigenvalue of M. In addition, $\sqrt{\mathbf{1}^T \cdot \mathbf{a}} = ||\boldsymbol{w}^p||_2$. This provides a way to build relationship between $||\boldsymbol{w}^p||_2$ and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ because M is expressed in terms of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Another interesting observation is that $\mathbf{1}^T \cdot \mathbf{a} = ||\boldsymbol{w}^p||_2^2 \mathbf{1}^T \cdot \mathbf{b} = \left(\mathbf{1}^T \cdot \mathbf{a}\right) \mathbf{1}^T \cdot \mathbf{b}$. This means that either $\mathbf{1}^T \cdot \mathbf{a} = ||\boldsymbol{w}^p||_2^2 = 0$ or $\mathbf{1}^T \cdot \mathbf{b} = 1$. The former case means that the parameters of the entire group is sparsed out and the latter case means that $\frac{(Q_1^p)^2}{\lambda_p^2} + \cdots + \frac{(Q_{n_p}^p)^2}{\lambda_p^2} = 1$ when the the group is not sparsed.

Bibliography

[1] F. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems*, 2010.

[2] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27:450–468, 2012.

[3] F. R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[4] S. Barotti, L. Lombardi, and P. Lombardi. Multi-module switching and fusion for robust video surveillance. In *IEEE International Conference on Image Analysis and Processing*, 2003.

[5] V. Bayer-Zubek. Learning diagnostic policies from examples by systematic search. In *Conference on Uncertainty in Artificial Intelligence*, 2004.

[6] S. Bengio, F. Pereira, Y. Singer, and D. Strelow. Group sparse coding. In *Advances in Neural Information Processing Systems*, 2009.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

[8] M. Bilgic and L. Getoor. Effective label acquisition for collective classification. In *International Conference on Knowledge Discovery and Data mining*, 2008.

[9] M. Bilgic and L. Getoor. Reflect and correct: A misclassification prediction approach to active inference. *ACM Transactions on Knowledge Discovery from Data*, 3:1–32, 2009.

[10] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *IEEE International Conference on Computer Vision*, 2001.

[11] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[13] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. *Annals of Statistics*, 40:1935–1967, 2012.

[14] D. Chen, D. Batra, W. T. Freeman, and M. K. Johnson. Group norm for learning latent structural svms. In *Advances in Neural Information Processing Systems Workshop on Optimization for Machine Learning*, 2011.

[15] D. Chen, A Bharusha, and H. Wactlar. People identification through ambient camera networks. In *International Conference on Data Engineering Worshop on Multimedia Ambient Intelligence, Media and Sensing*, 2007.

[16] D. Chen, M. Bilgic, L. Getoor, and D. Jacobs. Dynamic processing allocation in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:2174–2187, 2011.

[17] D. Chen, M. Bilgic, L. Getoor, D. Jacobs, L. Mihalkova, and T. Yeh. Active inference for retrieval in camera networks. In *IEEE Workshop on Person Oriented Vision*, 2011.

[18] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43:129–159, 2001.

[19] S. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *Visual Communications and Image Processing*, 2004.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[22] G. Denina, B. Bhanu, H. Nguyen, C. Ding, A. Kamal, C. Ravishankar, A. Roy-Chowdhury, A. Ivers, and B. Varda. Videoweb dataset for multi-camera activities and non-verbal communication. In B. Bhanu, C. Ravishankar, A. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, editors, *Distributed Video Sensor Networks*. Springer, 2010.

[23] S. K. Divvala. *Context and Subcategories for Sliding Window Object Recognition*. PhD thesis, Carnegie Mellon University, 2012.

[24] S. K. Divvala, A. A. Efros, and M. Hebert. How important are 'deformable parts' in the deformable parts model? In *Parts and Attributes Workshop, European Conference on Computer Vision*, 2012.

[25] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32:407–451, 2004.

[27] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision*, 2000.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. [Online: accessed 24-February-2013].

[29] R. Farrell, D. Doermann, and L. Davis. Learning higher-order transition models in medium-scale camera networks. In *International Conference on Computer Vision workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, 2007.

[30] P. F. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[31] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://www.cs.brown.edu/∼pff/latent-release4/. [Online: accessed 24-February-2013].

[32] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2010.

[33] J. Fiscus, J. Garofolo, T. Rose, and M. Michel. Avss multiple camera person tracking challenge evaluation overview. In *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009.

[34] T. Fletcher. Switching autoregressive hidden markov model. Technical report, Department of Computer Science, University College London, 2009.

[35] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. release4-notes.pdf. http://www.cs.brown.edu/∼pff/latent-release4/. [Online: accessed 24-February-2013].

[36] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European Conference on Computer Vision*, 2008.

[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[38] R. A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2:22 –26, 1966.

[39] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:696–706, 2002.

[40] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *IEEE International Conference on Computer Vision*, 2005.

[41] R. C. Jain and H. H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:206–213, 1979.

[42] O. Javed, K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *IEEE Computer Vision and Pattern Recognition*, 2005.

[43] Y. Jia, M. Salzmann, and T. Darrell. Factorized latent spaces with structured sparsity. In *Advances in Neural Information Processing Systems*, 2010.

[44] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46:81–96, 2002.

[45] J. Keller. Dcrp review: Canon powershot s5 is. http://www.dcresource.com/reviews/canon/powershot_s5-review, 2011. [Online; accessed 24-February-2013].

[46] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11:172–185, 2005.

[47] T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Conference on Empirical Methods in Natural Language Processing*, 2010.

[48] A. Krause and C. Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research*, 35:557–591, 2009.

[49] R. Krishna, K. McCusker, and N. E. O'Connor. Optimising resource allocation for background modeling using algorithm switching. In *ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[50] P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 2010.

[51] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[52] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE International Conference on Image Processing*, 2002.

[53] Z. Lin and L. Davis. Learning pairwise dissimilarity profiles for appearance recognition in visual surveillance. In *International Symposium on Visual Computing*, 2008.

[54] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *International Conference on Machine Learning*, 2009.

[55] B. P. L. Lo and S. A. Velastin. Automatic congestion detection system for underground platforms. In *IEEE International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001.

[56] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[57] C. C. Loy, T. Xiang, and S. Gong. Modelling activity global temporal dependencies using time delayed probabilistic graphical model. In *IEEE International Conference on Computer Vision*, 2009.

[58] C. C. Loy, T. Xiang, and S. Gong. Multi-camera activity correlation analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[59] Q. Lu and L. Getoor. Link based classification. In *International Conference on Machine Learning*, 2003.

[60] S. Lyall. London bombers visited earlier, apparently on practice run. http://www.nytimes.com/2005/09/21/international/europe/21london.html, 2005. [Online; accessed 13-February-2013].

[61] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning*, 2009.

[62] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[63] C. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[64] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. In *IEEE International Conference on Computer Vision*, 2005.

[65] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

[66] J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI Workshop on Learning Statististical Models from Relational Data*, 2000.

[67] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, 2010.

[68] N. Olivarez-Giles. Flickr reaches 6 billion photos uploaded. http://latimesblogs.latimes.com/technology/2011/08/flickr-reaches-6-billion-photos-uploaded.html, 2011. [Online; accessed 13-February-2013].

[69] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[70] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[71] M. Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[72] PRWeb. Brickstream launches video analytics managed services program. http://www.prweb.com/releases/2008/05/prweb955134.htm, 2008. [Online; accessed 24-February-2013].

[73] A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for ℓ_1-ℓ_∞ regularization. In *International Conference on Machine Learning*, 2009.

[74] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[75] Y. Radovilsky, G. Shattah, and S.E. Shimony. Efficient deterministic approximation algorithms for non-myopic value of information in graphical models. In *IEEE International Conference on Systems, Man and Cybernetics*, 2006.

[76] M. Rattigan, M. Maier, and D. Jensen. Exploiting network structure for active inference in collective classification. In *IEEE International Conference on Data Mining Workshop on Mining Graphs and Complex Structures*, 2007.

[77] J. Rittscher, J. Kato, S. Joga, and A. Blake. A probabilistic background model for tracking. In *European Conference on Computer Vision*, 2000.

[78] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM SIGGRAPH*, 23:309–314, 2004.

[79] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:23–38, 1998.

[80] R. Ryan and Agencies. July 7 bombers staged dummy run. http://www.guardian.co.uk/uk/2005/sep/20/july7.uksecurity, 2005. [Online: accessed 24-February-2013].

[81] M. Salzmann, C. Ek, R. Urtasun, and T. Darrell. Factorized orthogonal latent spaces. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[82] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[83] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29:93–106, 2008.

[84] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[85] P. Sinha. Object recognition via image invariants: A case study. *Investigative Ophthalmology and Visual Science*, 35:1735–1740, 1994.

[86] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision*, 2003.

[87] B. Song and A. Roy-Chowdhury. Stochastic adaptive tracking in a camera network. In *IEEE International Conference on Computer Vision*, 2007.

[88] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[89] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[90] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

[91] K. Tieu, G. Dalley, and W. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *IEEE International Conference on Computer Vision*, 2005.

[92] B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. *Technometrics*, 2005.

[93] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[94] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1582–1596, 2010.

[95] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occulsion. In *Advances in Neural Information Processing Systems*, 2009.

[96] S. Vijayanarasimhan and K. Grauman. Whats it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[97] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[98] S. Vijayanarasimhan and A. Kapoor. Visual recognition and detection under bounded computational resources. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[99] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57:137–154, 2002.

[100] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM SIGGRAPH*, 24:585–594, 2005.

[101] S. B. Wang, A. Quattoni, L.-P. Morency, and D. Demirdjian. Hidden conditional random fields for gesture recognition. In *IEEE conference on Computer Vision and Pattern Recognition*, 2006.

[102] X. Wang, G. Doretto, T. Sebastian, J. Rittscher, and P. Tu. Shape and appearance context modeling. In *IEEE International Conference on Computer Vision*, 2007.

[103] D. Weiss and B. Taskar. Structured prediction cascades. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[104] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[105] R. Xiao, L. Zhu, and H. Zhang. Boosting chain learning for object detection. In *IEEE International Conference on Computer Vision*, 2003.

[106] G. Yang and T. S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27:53–63, 1994.

[107] M. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:34–58, 2002.

[108] W. Yang, Y. Wang, A. Vahdat, and G. Mori. Kernel latent svm for visual recognition. In *Advances in Neural Information Processing Systems*, 2012.

[109] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Survey*, 38:13, 2006.

[110] C. J. Yu and T. Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning*, 2009.

[111] M. Yuan, M. Yuan, Y. Lin, and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[112] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

[113] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, 2004.

[114] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 67:301–320, 2005.