APPROVAL

Name: Zhi Feng Huang
Degree: Master of Science
Title of Thesis: Latent Boosting for Action Recognition

Examinining Committee: Dr. Eugenia Ternovska
Chair

Dr. Greg Mori, Senior Supervisor

Dr. Ze-Nian Li, Supervisor

Dr. Anoop Sarkar, SFU Examiner

Date Approved: ________________________________
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the “Institutional Repository” link of the SFU Library website (www.lib.sfu.ca) at http://summit/sfu.ca and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author’s written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2011
Abstract

In this thesis, we present work towards addressing a grand challenge of computer vision, human action recognition and detection. In particular, we focus on the problem of recognizing and detecting the actions of a person from a video sequence.

To recognize human actions in a video, a typical approach involves first detecting and tracking people, followed by classification. However, accurate tracking is challenging, and the state-of-art tracking methods are not reliable. Since accurate tracking is not a direct end-goal of action recognition, we consider tracking as a latent variable and train a model focused on action recognition. We propose a novel learning algorithm for training models with latent variables in a boosting framework. Moreover, we show that the algorithm can be used to train an action recognition model in which the tracking trajectory of a person is a latent variable. This new model outperforms baselines on a variety of datasets.

Keywords: computer vision; machine learning; boosting; human action recognition; human action detection
Acknowledgments

First, I would like to thank my supervisor Dr. Greg Mori, for his support and guidance in both my undergraduate and graduate careers. Greg directed me into the fascinating realm of computer vision, and has spent great efforts in helping me through many obstacles during my thesis. The knowledge and skills I learned from him will continue to help me in my future life. Thanks to my thesis committee members, Dr. Ze-Lian Li and Dr. Anoop Sarkar for their insightful comments on this thesis. Thanks to Dr. Eugenia Ternovska for chairing my thesis defense.

Weilong Yang and Dr. Yang Wang are my main collaborators in research. I am deeply grateful for their generous help when I first started computer vision. I also would like to thank all members in SFU Vision and Media Lab, especially Tian Lan and Guang-Tong Zhou. They helped me a lot on this thesis.

Finally, thanks to my family. It is only through their dedication and sacrifices that I could have come this far.
Contents

Approval ii
Partial Copyright License iii
Abstract iv
Acknowledgments v
Contents vi
List of Tables viii
List of Figures ix

1 Introduction 1
  1.1 Human Action Recognition and Detection ....................... 2
  1.2 Machine Learning and Boosting .................................. 5
  1.3 Contribution ...................................................... 7
  1.4 Outline ............................................................. 8

2 Related Work 10
  2.1 Learning Algorithms ............................................... 10
      2.1.1 Max-Margin Approach ...................................... 10
      2.1.2 Conditional Random Field Approach ....................... 13
      2.1.3 Boosting ..................................................... 15
  2.2 Human Action Recognition ......................................... 17
      2.2.1 Motion Approach ............................................. 17
List of Tables

4.1 Comparison of classification accuracy with similar baselines on the Weizmann dataset. ................................................. 37
# List of Figures

1.1 The nine actions from the Weizmann dataset [3] ............................... 3

1.2 Typical tracklets from the TRECVID dataset. The first row consists two 5-frame tracklets of running people. The second row consists two 5-frame tracklets of not-running people. ........................................ 4

2.1 Two multiple instance learning SVMs. Negative data points are denoted by “-” symbols, and positive bag data points by numbers that encode the bag membership. Circles are the support vectors. (a) sketches the mi-SVM solution while (b) shows the MI-SVM solution. This image is from [2] ........................................ 11

2.2 Constructing the motion descriptor. (a) Original image, (b) Optical flow, (c) Separating the $x$ and $y$ components of optical flow vectors, (d) Half-wave rectification of each component to produce 4 separate channels, (e) Final blurry motion channels. This image is from [7] ........................................ 18

4.1 Illustration of the LatentBoost model. Each circle corresponds to a variable, and each square corresponds to a factor in the model. $x_1$ to $x_5$ are the variables representing frame 1 to frame 5 in a tracklet, and $l_1$ to $l_5$ are the latent variables representing the offset of each frame in a tracklet ........................................ 32

4.2 Per-frame and per-video classification accuracies on the Weizmann dataset with different stride configurations and 5-frame tracklet on LatentBoost ........................................ 34

4.3 Typical tracklets from the Weizmann dataset. The four actions are jacking, running, jumping, and waving ........................................ 35

4.4 Per-frame and per-video classification accuracies on the Weizmann dataset with and without pairwise potentials and $5 \times 5$ strides, 5-frame tracklet on LatentBoost ........................................ 36
4.5 Per-frame and per-video classification accuracies on the Weizmann dataset with different lengths of tracklets and 5×5 strides configuration on LatentBoost. 37

4.6 The representation of the positive optical flow features from the final LatentBoost classifier for the Weizmann dataset. The red arrows are from the first 4 directional motion channels, and the blue points are from the motion magnitude channel. (a) bend, (b) jack, (c) jump, (d) pjump, (e) run, (f) side, (g) walk, (h) wave1, (i) wave2. 38

4.7 Comparison of GradientBoost with our LatentBoost on the TRECVID dataset. (a) The DET curves of the two methods. (b) The minimum DCR scores of the two methods. (c) Examples of running detection in the TRECVID dataset. Note that the DET curve is the lower the better and the minimum DCR score is also the lower the better. Please zoom in for a clear view of the plot. 39

4.8 Comparison of LatentBoost with and without pairwise terms on TRECVID dataset. 41

4.9 Typical tracklets from the TRECVID dataset. The first row consists two 5-frame tracklets of running people. The second row consists two 5-frame tracklets of not-running people. 42

4.10 Running children in the TRECVID dataset. 43

4.11 Examples of running detections. 44

4.12 Hard examples in the running detection. 44
Chapter 1

Introduction

There are thousands of surveillance cameras in a city nowadays. They are used to protect us in train stations, airports, and buildings. However, it is heavy labor to monitor all of them 24 hours a day and 7 days a week. To make full use of the surveillance cameras and reduce operation costs, we need an automatic system that can spot suspect or interesting events in the surveillance videos. We would like to have a report from the system consisting of the details of the events, such as the temporal information of when the events occur in the video and the spatial information of where the events occur in the frames. Building such a system requires using many computer vision technologies, such as object detection, tracking, and human action recognition and detection. For example, consider detecting a running person in a surveillance video. In a typical approach, we first need to find all the people in the video and track them, followed by classification to see if the people are running or not.

In the computer vision literature, object detection, tracking, and human action recognition have been studied as separate problems. However, these problems are closely interleaved. The solution of one problem can typically provide useful information for solving another problem. For example, there are approaches of tracking that use different detection techniques. Similarly, human action recognition may rely on the performance of tracking. In this thesis, we limit ourselves to the problem of human action recognition and detection.
CHAPTER 1. INTRODUCTION

1.1 Human Action Recognition and Detection

Human action recognition and detection is an active research field in computer vision. The problem usually involves videos or still images. Since videos have more information compared to still images, such as temporal information and motion information, the approaches to handle videos are typically different from the approaches for still images. In this thesis, we limit ourselves to videos only. The goal of human action recognition is to classify a video into one of a set of pre-defined categories according to the action performed by the person in the video. The input of the problem is a video that consists of a person who is doing one of pre-defined actions, and the output should be the label of the action, which is also the label of the video.

The goal of human action detection is more challenging. For detection, we localize the spatial-temporal positions of the target action in a video, rather than getting a single class label for the entire video. The videos for detection may or may not contain a person who is doing the target action. Also, the videos typically consist of many people, and the people can do various actions. The output for detection should be a list of occurrences of the target action, and each occurrence could consist of the temporal information about when the action occurs in the video and the spatial information about where the action occurs in the frames. The temporal information can be the range of the frames that contain the action, and the spatial information is commonly the bounding box of the person who is doing the action in each frame. Real-world applications (i.e. surveillance monitoring) are usually interested in detection. We believe that solving human action recognition is essential to solving human action detection. If we want to detect an action, we first must be able to recognize the action reliably.

**Human Action** In this thesis, we use the term *human action* for simple and atomic human movements, such as walking, running, and jumping, as shown in Figure 1.1. The term human activity is also used in computer vision literature. We refer *human activity* as a series of atomic human actions. For example, an attack in a basketball game is an activity. It consists of several atomic human actions, such as running, dribbling, and shooting. Furthermore, an *event* consists of a series of human activities. For example, a basketball game is an event. It involves attacking, defending, and substituting. Note that the definitions are not unique. In the literature, the terms are used interchangeably. In this thesis, we focus on human action as simple and atomic human movements.
CHAPTER 1. INTRODUCTION

Figure 1.1: The nine actions from the Weizmann dataset [3].

**Figure-Centric Representation** Given a video, we want to focus on the person who is doing the action. In this thesis, we choose a *figure-centric* representation to present the input video to our proposed framework. Instead of directly feeding the original video into our framework, we first apply a standard human detection and tracking algorithm to track and stabilize the figures in the video. The tracker processes each frame to track each detected person in the frame within the next $k$ consecutive frames, where $k$ is a parameter. The resulting video clips have the human figures in the centre of the frames, and each video clip has the length of $k$ frames. Each video clip is essentially a partial trajectory of a person in the video. We use the term *tracklet* [42] to refer a figure-centric video clip generated by the tracker. Figure 1.2 shows four 5-frame tracklets from the TRECVID dataset [33]. As illustrated in Figure 1.2, the tracker will suffer from jitter, especially when people are
Figure 1.2: Typical tracklets from the TRECVID dataset. The first row consists two 5-frame tracklets of running people. The second row consists two 5-frame tracklets of not-running people.

performing varied actions. Although we admit that accurate tracking is challenging, the use of figure-centric representations would guarantee that we are recognizing the movements of the human, rather than the translations caused by a moving camera. Since accurate tracking is not a direct end-goal of action recognition, it is natural to consider precise tracking as unobserved information and train a model focused on action recognition. This is the main motivation for us to develop a learning algorithm that allows for training of models which have unobserved variables that are inferred from other observed variables in the models. The details of the proposed algorithm are discussed in Chapter 3.

**Human Action Recognition** In this thesis, we consider the task of human action recognition as a classification problem. Given an input video, we first extract tracklets from the video. From the tracklets, we can obtain some motion features and colour features. We use the discriminative information in the features to classify the tracklet to be one of the pre-define action labels, such as running and walking. Our approach does not consider any context information or pose information.

**Challenges** We would like to emphasize the difficulties in human action recognition and detection. One of the challenges in the problem comes from the (1) intra-class variations. Different people may perform the same action in different ways. Let us consider the running action as an example. People can run with different speed, styles, and directions. The other challenges include, but are not limited to, (2) appearance variation, (3) viewpoint variation, (4) illumination variation, (5) cluttered background and environment, and (6) low-resolution videos. For example, people can wear different clothes; cameras can be placed in different
locations; lighting can vary from time to time; background can be cluttered and moving people can be occluded; cameras can only record low-frame-rate and low-resolution videos. In order to be applied to many real-world dataset such as videos from surveillance cameras, a good action recognition and detection algorithm must be able to handle most of the challenges well.

1.2 Machine Learning and Boosting

Machine learning consists of the techniques that enable a computer program to learn. Mitchell [27] has a widely quoted definition about learning: A computer program is said to learn from experience \( E \) with respect to some class of tasks \( T \) and performance measure \( P \), if its performance at tasks in \( T \), as measured by \( P \), improves with experience \( E \). Computer vision is an active application of machine learning. Since the data in computer vision problems is getting much more complex, hardcoded solutions are not sufficient to handle the data. Instead, people utilize machine learning techniques to capture the discriminative patterns from the data and generalize from the data. The discriminative patterns can then be used to produce reasonable output in new data. For example, hardcoding what a running action looks like is not feasible if we have hundreds of videos and many variations of running samples. However, there are some common patterns in the running action, such as the movements of the body, average speed of running, etc. We can use a machine learning algorithm to learn those patterns from a set of training videos. The learned patterns can then be used to classify new videos as running or not-running. To state human action recognition in Mitchell’s definition:

Human Action Recognition

- Task \( T \): recognizing and classifying pre-defined actions in videos
- Performance measure \( P \): percent of videos correctly classified
- Training experience \( E \): a set of videos with corresponding action labels

Supervised Learning In machine learning, there are several kinds of algorithms, such as supervised learning, unsupervised learning, and reinforcement learning. In this thesis, we focus on supervised learning. The goal of a supervised learning algorithm is to produce
an inferred function from a set of supervised data. The supervised data is a set of training examples, which are pairs consisting of an input object and an output value (or supervised label). The inferred function produced is called a classifier if the output is discrete or a regression function if the output is continuous. The inferred function should predict the correct output value for any valid input object. The training algorithm learns the inferred function to optimize a loss function which measures the accuracy of the inferred function on the training data. The value of the loss function is minimized on the training data giving the inferred function.

**Boosting** There are several common learning algorithms used in computer vision, and boosting is one of them. Boosting is a set of supervised learning algorithms. The underlying idea of boosting is to combine simple hypotheses to form an ensemble hypothesis such that the performance of the ensemble hypothesis is improved compared to the performance of the simple hypotheses, i.e. “boosted”. The simple hypothesis is called weak learner in boosting literature, and they can be arbitrary classifiers or regression functions. The ensemble hypothesis is call a strong learner, and it is expected to have much higher performance compared to the weak learners. There are successful stories in computer vision which use boosting algorithms, such as the Viola-Jones face detector [36]. The learning algorithm we propose in Chapter 3 is also a variant of boosting algorithm.

**Latent Variable** In statistics, there are two kinds of variables, observable variables and latent variables. Observable variables are the ones that can be directly measured in the model. Latent variables are variables that are not directly observed but are inferred from observed variables. For example, consider a face recognition model. We can have observable variables for measuring the pixel information of a given image. We also can have a latent variable for the location of a face in the image. We do not know the location of a face in the image, but we can obtain such location by performing an inference through the model with the observable pixel information. Moreover, instead of being the location of a face, a latent variable can be a complex structure such as a graph, a so called structured latent variable. For example, a structured latent variable can be a short trajectory of a person in a video. We can use structured latent variables to capture and model much more information that we cannot directly obtain from the data. For example, we can model a face as a part based model. We divide a face into several parts, and each part has a latent variable representing the location of the part in the face. There are constraints between some parts to model the distances between those parts. In an iterative learning process, we can refine the values
of those latent variables and constraints by inference. By having such a structured latent variable in our model, we can locate both the face and the parts of the face.

1.3 Contribution

The contribution of this thesis is twofold. First, we develop a novel learning algorithm for training models with latent variables in a boosting framework. We call it LatentBoost. The algorithm allows for training of structured latent variable models with boosting. For most of the boosting algorithms, each training example consists of an input feature and an output label. However, LatentBoost allows each training example to have an extra latent structure, which can be an arbitrary undirected graph. The values of the nodes in the latent structure for each training example are unobserved before training time, and they are learned and refined during training time. By considering the latent structures in training examples, we believe that we can discover more useful information in training time and improve the performance of the final strong learner. The popular latent SVM [9] framework allows for training of models with structured latent variables in a max-margin framework. LatentBoost provides an analogous capability for boosting algorithms. Moreover, it generalizes from the multiple instance boosting (MILBoost) [37]. MILBoost can only handle a single latent variable, such as the location of a face in an image. In our LatentBoost, the latent variable can have much more complex structure, such as the space-time locations of a person in a track.

There are existing machine learning frameworks that can handle structured latent variable models, such as latent SVM [9] and hidden Conditional Random Field (hCRF) [29]. However, LatentBoost has its own advantages. First, compared to SVMs, there is no $C$ parameter turning in boost. This can save a lot of training time. Second, LatentBoost does not require to normalize different features. All the features are encapsulated inside weak learners. It is important to normalize different features to the same range in SVMs. Otherwise, the features with large range will dominate. However, the common difficulty is that it is not trivial how we should normalize the features. The last but not the least advantage of LatentBoost is its feature selection ability. Each weak learner depends on one or few features. When LatentBoost is selecting weak learners, it is also selecting features. LatentBoost can select the critical features instead of concatenating all the features in a vector like what SVMs do.
The formulation of LatentBoost is similar to hCRF. LatentBoost and hCRF are two different approaches for solving the same problem. In hCRF, we define the scoring function with a set of parameters $\theta$. We learn the parameters $\theta$ by using a conjugated-gradient method to optimize a loss function. In LatentBoost, we are not learning the parameters of a scoring function. Since the scoring function is defined as a linear combination of weak learners, we learn the entire scoring function in LatentBoost instead. In each iteration, LatentBoost greedy picks the best weak learner and adds it to the final scoring function. In hCRF, we have to specify the features in the scoring function. The hCRF learning algorithm will learn the weights for the features. The algorithm assigns a large weight to the important features and assigns small weights to other features. The advantage of LatentBoost is that we do not need to specify which features we want. The algorithm will pick the features for us.

Our second contribution is to apply LatentBoost in human action recognition and detection. We propose a model in which the trajectory of a person is a latent variable. Each training example in our model is a figure-centric tracklet from a standard tracker. Since the tracker is not completely reliable, we allow each frame of a tracklet to move around in a fixed range with respect to its initial position from the tracker. This will provide some robustness against imperfections of the tracker. We use a chain structure to model the frames in a tracklet. Each node in the chain structure models the offset of each frame in the tracklet. There are unary motion features for action classification and pairwise appearance features for enforcing appearance consistency over adjacent frames.

The work presented in this thesis has been published in [16]. LatentBoost is developed in collaboration with Weilong Yang, Yang Wang, and Greg Mori. The action recognition model is proposed and developed by myself, as well as the experimental work.

1.4 Outline

The rest of this thesis is organized as follows:

Chapter 2 provides an overview of previous work in boosting and human action recognition that is most relevant to this thesis.

Chapter 3 focuses on LatentBoost, the novel algorithm we propose to allow for training of structured latent variable models with boosting.

Chapter 4 presents our action recognition model in LatentBoost framework. Different
features and implementation details are discussed. We show that our model outperforms baselines on a variety of datasets.

Chapter 5 concludes this thesis.
Chapter 2

Related Work

In this chapter, we give a general overview of previous work on learning algorithms (Section 2.1) and human action recognition (Section 2.2).

2.1 Learning Algorithms

Computer vision has been deeply influenced by machine learning techniques in recent years. There are many successful stories in computer vision associated with some techniques in the machine learning literature. For example, the Viola-Jones face detector [36] uses a boosting algorithm called AdaBoost, and Felzenszwalb et al. object detector [9] is trained by Latent SVM. In this review, we focus on discriminative learning methods that are highly related to our work.

2.1.1 Max-Margin Approach

Support vector machine (SVM) is a supervised learning method that constructs a hyperplane that has the largest distance to the nearest training data points of any class. The distance between the hyperplane to the nearest training data points is called the margin. In general, the larger the margin the lower the generalization error of the classifier. Generalization error measures the average error of a classifier over the entire possible set of valid data. SVM is a favourable classifier due to its max-margin property. SVM has been deeply studied in the machine learning community, and many machine learning textbooks have an excellent introduction about it, e.g. [32]. In supervised classification problems, we usually have a set
Figure 2.1: Two multiple instance learning SVMs. Negative data points are denoted by “-” symbols, and positive bag data points by numbers that encode the bag membership. Circles are the support vectors. (a) sketches the mi-SVM solution while (b) shows the MI-SVM solution. This image is from [2].

of training examples where each example consists of a data point and a class label. Multiple instance learning (MIL) [6] generalizes from this setting in which class labels are associated with sets of data points, or so-called bags, instead of individual data points. In MIL, a bag receives a particular label, if at least one of the data points in the bag possesses the label. In a binary classification problem, a bag is positive if at least one of data points in the bag is positive; otherwise, a bag is negative if every data point in the bag is negative. Andrews et al. [2] modify and extend SVM to deal with MIL problems in two different approaches. The first one, called mi-SVM, explicitly treats the class labels of individual data points as unobserved latent integer variables, subject to constraints defined by the bag labels. The second approach, called MI-SVM, generalizes the notion of a margin to bags and aims at maximizing the bag margin directly. The difference between the two approaches of max-margin problems is illustrated in Figure 2.1. For the data-point-centred mi-SVM, the margin of every data point in a positive bag matters. On the other hand, in the bag-centred MI-SVM, only one data point per positive bag matters, since it will determine the margin of the bag. The MI-SVM can be used to train models with a single plain latent variable. One bag contains all possible assignments of the latent variable for one training example.
Later, Felzenszwalb et al. [9] propose Latent SVM, a reformulation of MI-SVM in terms of latent variables. The main advantage of Latent SVM compared with MI-SVM is that Latent SVM can handle a structured latent variable. Latent SVM leads to a non-convex optimization problem. However, Latent SVM is considered as \textit{semi-convex}. Consider a set of training examples $D = (\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)$, where $y_i \in \{-1, +1\}$ for $i = 1, \ldots, n$. In a linear SVM, we train the parameter of the SVM, $\beta$, from the set $D$ by minimizing the objective function:

$$L(\beta) = \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \quad (2.1)$$

where $\max(0, 1 - y_i f_\beta(x_i))$ is the standard hinge loss and $f_\beta(x) = \beta \cdot \Phi(x)$ is linear in $\beta$ for scoring $x$. The hinge loss in a linear SVM is convex in $\beta$ for each training example because it is always the maximum of two convex functions. It follows that the objective function of a linear SVM is convex in $\beta$. However, the scoring function in Latent SVM is not linear in $\beta$:

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z) \quad (2.2)$$

where $z$ is a latent value from a set of possible latent values, $Z(x)$, for an example $x$. It follows that the hinge loss, $\max(0, 1 - y_i f_\beta(x_i))$, is convex in $\beta$ when $y_i = -1$ but the hinge loss is not convex in $\beta$ when $y_i = 1$ (i.e. $(1 - y_i f_\beta(x_i))$ is concave). This property of the loss function is called \textit{semi-convexity}. To solve the optimization problem in Latent SVM, coordinate descent approach is used. It is an iterative process. In each iteration, the first step is to fix the latent values for each positive example by inference. In this case, $f_\beta(x_i)$ is linear in $\beta$ for a positive example and the loss due to each positive example is also convex in $\beta$. The second step can then solve a convex optimization problem with unfixed latent values for negative examples. Latent SVM provides a framework to train discriminative models with structured latent variables. However, Latent SVM still has the main disadvantage of SVMs, which is the need to turn the parameter $C$ in the objective function. The parameter $C$ controls the relative weight of the regularization term in the objective function. The performance of a SVM is very sensitive to the $C$ value. It may take a long time on cross validation to find the best $C$ value. The second main disadvantage of SVMs is the need to normalize different features. It is important to normalize different features to the same range. Otherwise, the features with large range will dominate. However, the common
difficulty is that it is not trivial how we should normalize the features.

### 2.1.2 Conditional Random Field Approach

Conditional Random Field (CRF) [20] is a discriminative probabilistic framework that models a conditional probability distribution over labels and data points, rather than a joint distribution over both labels and data points. Let $X$ and $Y$ be random variables respectively ranging over data points and their corresponding labels. In order to define a joint distribution $\Pr(X, Y)$, generative models must enumerate all possible data points and assume the features in the data points are independent from other features. This independence assumption is not desirable for many real-world applications. Instead, we can use conditional models that define a conditional probability $\Pr(Y|X)$ over labels given a data point. The conditional model is then used to label a data point $x$ by selecting the label $y^*$ that maximizes the conditional probability (i.e. $y^* = \arg \max_{y \in Y} \Pr(y|x)$).

Quattoni et al. [29] propose hidden Conditional Random Field (hCRF) which extends the CRF framework to incorporate hidden variables and combine class CRFs into a unified framework for part based object recognition. Given a set of images and labels $(x_i, y_i)$ for $i = 1, \ldots, n$. Each image $x$ is a vector of patches $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$. For any image $x$ there is also a vector of part variables $h = \{h_1, h_2, \ldots, h_m\}$ where $h_j \in H$. The part variables are not observed on training data, and will therefore form a set of hidden variables in the model. Each $h_j$ is considered as a label of $x_j$. The hCRF defines a conditional probabilistic model:

$$
\Pr(y, h|x, \theta) = \frac{\exp(\Psi(y, h, x, \theta))}{\sum_{y', h} \exp(\Phi(y', h), x, \theta)}
$$

where $\theta$ are the parameters of the model, and $\Phi(y, h, x, \theta)$ is a potential function parameterized by $\theta$. The potential function defines an undirected graph, with the hidden variables corresponding to vertices in the graph. It follows that:

$$
\Pr(y|x, \theta) = \sum_h \Pr(y, h|x, \theta) = \frac{\sum_h \exp(\Psi(y, h, x, \theta))}{\sum_{y', h} \exp(\Phi(y', h), x, \theta)}
$$

Given a new test image $x$ and parameter values $\theta^*$ induced from the training data, we can take the label for the image to be $\arg \max_{y \in Y} \Pr(y|x, \theta^*)$. The model parameters $\theta$ are learned by maximizing the conditional log-likelihood on the training images:
\[ \theta^* = \arg \max \mathcal{L}(\theta) = \arg \max \sum_i \log \Pr(y_i | x_i, \theta) - \frac{1}{2\sigma^2}||\theta||^2 \]  

(2.5)

where the first term is the log-likelihood of the data and the second term is to regularize the parameter values. Different from CRF [20], the objective function \( \mathcal{L}(\theta) \) of hCRF is not concave, due to the hidden variables \( h \). However, we still can use gradient ascent to find \( \theta \) that is locally optimal.

Wang and Mori [39] propose a similar model for action recognition. Instead of maximizing the conditional likelihood in hCRFs, they also propose to learn the parameters \( \theta \) by maximizing the margin, so called Max-Margin hCRF (MMhCRF). In MMhCRF, an \((x, y)\) pair is scored by a function of the form:

\[ f_\theta(x, y) = \max_h \theta^T \Phi(x, h, y) \]  

(2.6)

where \( \theta \) are the model parameters and \( h \) is a vector of hidden variables. The hidden variables form a undirected graphical model. We use \( y^* = \arg \max_{y \in Y} f(x, y) \) to classify a new example \( x \). To learn \( \theta \), we have the following optimization problem:

\[
\min_{\theta, \xi} \frac{1}{2}||\theta||^2 + C \sum_{i=1}^{n} \xi^{(i)} \\
\text{s.t. } f_\theta(x^{(i)}, y) - f_\theta(x^{(i)}, y^{(i)}) \leq \xi^{(i)} - 1, \forall i, \forall y \neq y^{(i)} \\
\xi^{(i)} \geq 0, \forall i
\]  

(2.7)

It turns out that, similar to Latent SVMs [9], MMhCRFs have the property of semi-convexity. A local optimum of (2.7) can be computed by using a coordinate descent algorithm similar to Latent SVMs: 1) holding \( \theta \) and \( \xi \) fixed, optimize the hidden variables \( h^{(i)}_{y^{(i)}} \) for every \((x^{(i)}, y^{(i)})\) pair; 2) holding \( h^{(i)}_{y^{(i)}} \) fixed, optimize both \( \theta \) and \( \xi \).

The learning algorithms for MMhCRFs and hCRFs involve solving two different types of inference problems - maximizing over \( h \), versus summing over \( h \). It is shown that MMhCRFs is favourable compared to hCRFs in [39]. However, MMhCRFs still have the limitations from SVMs, namely turning the parameter \( C \) and normalizing different features.
2.1.3 Boosting

Boosting is an iterative machine learning technique for combining multiple base classifiers to produce a form of committee whose performance can be significantly better than that of any of the base classifiers. The base classifiers also are known as weak learners, and the boosted committee is called the strong learner. The most widely used form of boosting algorithm is called AdaBoost [11] by Freund and Schapire. Given a two-class classification problem with \( N \) data points, each data point is given an associated weighting parameter \( w_n \), which is initially set \( 1/N \) for all data points. At each stage of the algorithm, AdaBoost trains a new classifier using a data set in which the weighting coefficients are adjusted according to the performance of the previously trained classifier so as to give greater weight to the misclassified data points. The idea is to focus on the misclassified data points in the next round. When the desired number of weak classifiers have been trained, they are combined to form a strong learner using coefficients that give different weight to different weak learners.

The Viola-Jones face detector [36] uses a variant of AdaBoost to select features and train a strong classifier. Each weak learner is a classifier depending on a single feature. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak learner \( h(\mathbf{x}, f, p, \theta) \) thus consists of a feature \( f \), a threshold \( \theta \) and a polarity \( p \) indicating the direction of the inequality:

\[
    h(\mathbf{x}, f, p, \theta) = \begin{cases} 
        1 & \text{if } p f(\mathbf{x}) < p \theta \\
        -1 & \text{otherwise} 
    \end{cases} 
\]  

which is also called a decision stump in the machine learning literature. Boosting with decision stumps has a good property which is that it does not require to normalize different features. Each decision stump depends on one feature, and the range of a feature in one decision stump will not affect the performance of other decision stumps or the final strong classifier. Ali et al. [1] propose to use AdaBoost to select pose-indexed features and pose estimators for object detection. A pose-indexed feature is a feature depended on a pose and an image. Instead of labelling poses in the training data and searching poses in testing, they consider the poses in the pose-indexed features are latent variables. They combine a set of pose estimators and a set of pose-indexed features to form the pool of features for AdaBoost. In each iteration of AdaBoost, a pose estimator and a pose-indexed feature are selected. The
pose from the pose estimator is then used to compute the value of the pose-indexed feature. This approach overcomes the training of pose-dedicated weak learners as well as the labelling and computational overheads of searching pose space in testing. This approach also shows the flexibility of AdaBoost, which can combine various features, classifiers, and estimators.

Freund and Schapire [12] show that AdaBoost minimizes an exponential loss function:

\[ E = \sum_{n=1}^{N} \exp\{ -y_nf_m(x_n) \} \] (2.9)

where \( f_m(x) \) is a classifier from AdaBoost and \( y_n \in \{-1, +1\} \) are the labels for the training data \( x_n \). Lebanon and Lafferty [22] show that the only difference between minimizing the exponential loss used by AdaBoost and maximum likelihood for exponential models is that the latter requires the model to be normalized to form a conditional probability distribution over labels. They show that AdaBoost and logistic regression can be reduced to the same optimization problem with the same loss function and same feature constraints.

Friedman [13] proposes a framework, called GradientBoost, which views boosting as a gradient descent process. GradientBoost generalizes boosting methods by allowing optimization of an arbitrary differentiable loss function. GradientBoost will be discussed in detail in the next chapter as it inspires our framework. Mason et al. [25] also propose a similar framework called AnyBoost, which considers boosting as a gradient descent in function space. Later, Viola et al. [37] propose to use AnyBoost and two Multiple Instance Learning (MIL) cost functions to formulate MILBoost. Given an example that is indexed with two indices: \( i \), which indexes the bag, and \( j \), which indexes the example within a bag. The score of the example is \( s_{i,j} = C(x_{i,j}) = \sum_t \lambda_tC_t(x_{i,j}) \), a weighted sum of weak classifiers. The probability that an example is positive is given by a logistic function:

\[ p_{i,j} = \frac{1}{1 + \exp(-s_{i,j})} \] (2.10)

Then the probability that the bag is positive is a “noisy OR” \( p_i = 1 - \prod_{j \in i}(1 - p_{i,j}) \). Under this model the likelihood assigned to a set of training bags is:

\[ L(C) = \prod_{i} p_{i}^{y_i}(1 - p_i)^{(1 - y_i)} \] (2.11)

where \( y_i \in \{0, 1\} \) is the label of bag \( i \). Following the AnyBoost approach, we can compute the weight of each example, which is the derivative of the cost function with respect to a
In the $t^{th}$ iteration of boosting, a weak learner $c^t$ that maximizes $\sum_{i,j} c^t(x_{i,j})w_{i,j}$ is selected. The weight for the weak learner is then determined by a line search to maximize $\log L(C + \lambda_t c_t)$. Another variant of MILBoost is to use the Integrated Segmentation and Recognition (ISR) cost function. Given $X_{i,j} = \exp(s_{i,j})$ and $S_i = \sum_{j \in i} X_{i,j}$. The probability that the bag is positive is defined by ISR cost: $p_i = \frac{S_i}{1 + S_i}$. By following the AnyBoost approach, we can again compute the weight of each example:

$$\frac{\partial \log L(C)}{\partial s_{i,j}} = w_{i,j} = (y_i - p_i) \frac{X_{i,j}}{\sum_{j \in i} X_{i,j}}$$ (2.13)

MILBoost provides a boosting framework to train a model with a latent variable. For instance, the paper applies MILBoost to Viola-Jones face detection, which considers the location of a face in an image as the latent variable. A positive bag contains images of the same face in different locations. It is equivalent to emulating the latent variable space in each bag. However, the limitation of MILBoost is that the latent variable is a single variable. If the latent variable is a structured variable, each bag may have exponential number of examples. Since each example requires a weight, it is infeasible in computation to handle exponential number of examples. In contrast, our proposed algorithm in the next chapter provides a boosting framework that allows for training of structured latent variable models.

### 2.2 Human Action Recognition

Human action recognition is a very active area of research in computer vision. Weinland et al. [41], Turaga et al. [35], and Poppe [28] provide recent surveys. We review closely related work below.

#### 2.2.1 Motion Approach

A lot of work has been done in recognizing actions from video sequences. Much of this work is focused on analyzing patterns of motion. For example, Efros et al. [7] recognize the actions of small scale figures using features derived from blurred optical flow estimates.
Figure 2.2: Constructing the motion descriptor. (a) Original image, (b) Optical flow, (c) Separating the $x$ and $y$ components of optical flow vectors, (d) Half-wave rectification of each component to produce 4 separate channels, (e) Final blurry motion channels. This image is from [7].

Figure 2.2 shows how to construct the motion descriptors with an image. The idea is to separate the optical flow into channels that are sparse and non-negative. Little and Boyd [23] analyze the periodic structure of optical flow patterns for gait recognition. Jhuang et al. [17] describe a biologically plausible model containing alternating stages of spatio-temporal filter template matching and pooling operations. One of the stages is analyzed by a set of optical flow based features. Schindler and Van Gool [31] examine the issue of the length of video sequences needed to recognize actions. They extract optical flow at different scales, directions, and speeds as motion features. They also show that very short tracklets, as short as 7 frames, can be effective for action recognition. Our method of action recognition described in Chapter 4 also operates on tracklets, and we use optical flow based motion features in our experiments.

2.2.2 Probabilistic Models

Sminchisescu et al. [34] show that discriminative conditional random fields (CRFs) are better than Hidden Markov Models (HMMs) in action recognition. Generative approaches like HMMs have to make simplifying, often unrealistic assumptions on the conditional independence of observations given the motion class labels and cannot accommodate overlapping
features. In contrast, conditional models like CRFs seamlessly represent contextual dependencies, support efficient, exact inference using dynamic programming and their parameters can be trained using convex optimization. Probabilistic models with latent variables, such as the hidden conditional random fields (hCRFs) [29], have been explored for action recognition. Wang et al. [38] propose to use hCRFs for gesture recognition. They extend the original hCRFs [29], which captures spatial dependencies between hidden object parts, to model sequences where the underlying graphical model captures temporal dependencies across frames, and to incorporate long range dependencies. Wang and Mori [39] develop a similar model to ours, yet with max-margin and probabilistic learning criteria different from our boosting approach. In [39], they argue that the max-margin approach is better than the probabilistic learning approach. However, as we mention before, the max-margin approach has its limitations, namely turning parameter $C$ and normalizing different features, and our boosting approach can overcome these limitations.

### 2.2.3 Boosting Approach

Boosting algorithms are commonly used in action recognition. For example, Laptev and Pérez [21] learn a cascade of boosted action classifiers to detect actions in movies. Fathi and Mori [8] learn an efficient classifier on top of optical flow based features using AdaBoost. They first use AdaBoost to select a subset of so-called low-level features to form a pool of so-called mid-level features. Each low-level feature is a single optical flow based motion feature. The mid-level features are focused on local regions of the video sequence. Then they use AdaBoost again to select a subset of mid-level features to form the final strong classifier. Kim and Cipolia [18] also use AdaBoost to select a subset of pairwise canonical correlation features for action recognition. The common advantages of using boosting algorithm are efficient computation, flexible feature selection, and less parameter turning. Our proposed algorithm has the advantages of boosting, and moreover, it incorporates latent variables, in contrast to these pieces of work.
Chapter 3

Boosting with Latent Structures

In this work, we present LatentBoost, a novel learning algorithm for training models with latent variables in a boosting framework. This algorithm allows for training of structured latent variable models with boosting. The popular latent SVM [9] framework mentioned in previous chapter allows for training of models with structured latent variables in a max-margin framework. LatentBoost provides an analogous capability for boosting algorithms. LatentBoost is also different from MILBoost [37] which only supports a single latent variable. The effectiveness of this framework is highlighted by an application to human action recognition, which is described in details in the next chapter.

We first briefly review GradientBoost algorithm proposed by Friedman [13] in Section 3.1, which also serves as the baseline approach in our experiment. Then, we present our LatentBoost algorithm in Section 3.2.

3.1 GradientBoost

Let us consider a classification problem with \( K \) classes. We denote a class label \( y \) as a length \( K \) binary vector of all zeros with a single one for the corresponding class. For example, for the \( k \)-th class, the class label \( y \) is written as \( y = [y_1, y_2, ..., y_K] \), where \( y_k = 1 \) and \( y_j = 0 \) for all \( j \neq k \). In GradientBoost, the goal is to learn a set of scoring functions \( \{F_j(x)\}_{j=1}^K \), one for each class. The probability \( p_k(x) \) of an example \( x \) being class \( k \) can be defined as:

\[
p_k(x) = \frac{\exp(F_k(x))}{\sum_{j=1}^K \exp(F_j(x))}
\]  

(3.1)
CHAPTER 3. BOOSTING WITH LATENT STRUCTURES

Given a set of \( N \) training examples \( \{x^{(n)}, y^{(n)}\}_{n=1}^{N} \), GradientBoost learns the set of scoring functions \( \{F_{j}(x)\}_{j=1}^{K} \) by minimizing the negative log-loss of the training data:

\[
\text{loss}_{GB} = \sum_{n=1}^{N} \Psi(\{y^{(n)}_{j}, F_{j}(x^{(n)})\}_{j=1}^{K}) = -\sum_{n=1}^{N} \sum_{j=1}^{K} y^{(n)}_{j} \log p_{j}(x^{(n)}). \tag{3.2}
\]

Under the boosting framework, the scoring function \( F_{k}(x) \) for the \( k \)-th class is assumed to be a linear combination of so-called “weak learners”:

\[
F_{k}(x) = \sum_{m=1}^{M} \rho_{k,m} h_{k,m}(x) \tag{3.3}
\]

where \( h_{k,m} \) is the \( m \)-th weak learner for the \( k \)-th class, and \( \rho_{k,m} \) is the weight associated with this weak learner. Let \( F_{k,m}(x) \) be the scoring function for the \( k \)-th class after \( m \) iterations.

At the \( m \)-th iteration of the gradient boosting algorithm, for the \( k \)-th class, a new weak learner \( h_{k,m} \) and its weight \( \beta_{k,m} \) are obtained by solving the following optimization problem:

\[
(\beta_{k,m}, h_{k,m}) = \arg \min_{\beta,h} \sum_{n=1}^{N} \Psi(\{y^{(n)}_{k}, F_{k,m-1}(x^{(n)}) + \beta h(x^{(n)})\}). \tag{3.4}
\]

Then the scoring function for the \( k \)-th class will be updated as: \( F_{k,m}(x) = F_{k,m-1}(x) + \beta_{k,m} h_{k,m}(x) \). Let us define the functional gradient of the loss function with respect to the current estimated \( F_{k,m-1}(x) \) (\( k = 1, 2, ..., K \)) as:

\[
-g_{k,m}(x^{(n)}) = -\left[ \frac{\partial \Psi(\{y^{(n)}_{j}, F_{j}(x^{(n)})\}_{j=1}^{K})}{\partial F_{k}(x^{(n)})} \right]_{\{F_{j}(x) = F_{j,m-1}(x)\}_{j=1}^{K}} = y^{(n)}_{k} - p_{k}(x^{(n)}), \ n = 1, 2, ..., N \tag{3.5}
\]

The following are the steps to obtain Equation (3.5). First, we take the first derivative of the loss function in Equation (3.2):

\[
-g_{k,m}(x^{(n)}) = - \left[ \frac{\partial \Psi(\{y^{(n)}_{j}, F_{j}(x^{(n)})\}_{j=1}^{K})}{\partial F_{k}(x^{(n)})} \right] \tag{3.6}
\]

\[
= - \left\{ - \frac{\partial}{\partial F_{k}(x^{(n)})} \left[ \sum_{j=1}^{K} y^{(n)}_{j} \log p_{j}(x^{(n)}) \right] \right\} \tag{3.7}
\]

\[
= \frac{\partial}{\partial F_{k}(x^{(n)})} \left[ y^{(n)}_{k} \log p_{k}(x^{(n)}) \right] + \sum_{j \neq k} \frac{\partial}{\partial F_{k}(x^{(n)})} \left[ y^{(n)}_{j} \log p_{j}(x^{(n)}) \right] \tag{3.8}
\]
CHAPTER 3. BOOSTING WITH LATENT STRUCTURES

Given Equation (3.1), we can get the first derivative of the term with $y_k^{(n)}$ as:

$$\frac{\partial}{\partial F_k(x^n)} \left[ y_k^{(n)} \log p_k(x^{(n)}) \right]$$

(3.9)

$$= y_k^{(n)} \frac{1}{p_k(x^{(n)})} \frac{\partial}{\partial F_k(x^{(n)})} \left[ p_k(x^{(n)}) \right]$$

(3.10)

$$= y_k^{(n)} \frac{1}{p_k(x^{(n)})} \left[ \exp(F_k(x^{(n)})) \sum_{j=1}^K \exp(F_j(x^{(n)})) - \left[ \exp(F_k(x^{(n)})) \right]^2 \right]$$

(3.11)

$$= y_k^{(n)} \frac{\sum_{j=1}^K \exp(F_j(x^{(n)})) - \exp(F_k(x^{(n)}))}{\sum_{j=1}^K \exp(F_j(x^{(n)}))}$$

(3.12)

$$= y_k^{(n)} - y_k \frac{\exp(F_k(x^{(n)}))}{\sum_{j=1}^K \exp(F_j(x^{(n)}))}$$

(3.13)

$$= y_k^{(n)} - y_k p_k(x^{(n)})$$

(3.14)

Similarly, given Equation (3.1), we can get the first derivatives of the terms with $y_j^{(n)}$ as:

$$\frac{\partial}{\partial F_k(x^n)} \left[ y_j^{(n)} \log p_j(x^{(n)}) \right]$$

(3.15)

$$= y_j^{(n)} \frac{1}{p_j(x^{(n)})} \frac{\partial}{\partial F_k(x^{(n)})} \left[ p_j(x^{(n)}) \right]$$

(3.16)

$$= y_j^{(n)} \frac{1}{p_j(x^{(n)})} \left[ 0 \times \sum_{j=1}^K \exp(F_j(x^{(n)})) - \exp(F_k(x^{(n)})) \exp(F_j(x^{(n)})) \right]$$

(3.17)

$$= y_j^{(n)} - \frac{\exp(F_k(x^{(n)}))}{\sum_{j=1}^K \exp(F_j(x^{(n)}))}$$

(3.18)

$$= y_j^{(n)} - y_j^{(n)} p_k(x^{(n)})$$

(3.19)

Combining Equations (3.8), (3.14), and (3.19), we get

$$-g_{k,m}(x^{(n)}) = y_k^{(n)} - y_k^{(n)} p_k(x^{(n)}) + \sum_{j \neq k} -y_j^{(n)} p_k(x^{(n)})$$

(3.20)

$$= y_k^{(n)} - y_k^{(n)} p_k(x^{(n)}) + p_k(x^{(n)}) \sum_{j \neq k} y_j^{(n)}$$

(3.21)

$$= y_k^{(n)} - p_k(x^{(n)}) \sum_{j=1}^K y_j^{(n)}$$

(3.22)

$$= y_k^{(n)} - p_k(x^{(n)})$$

(3.23)
From Equation (3.22) to Equation (3.23), we use \( \sum_{j=1}^{K} y^{(n)}_j = 1 \), which comes from the fact that the class label \( y \) is a vector where \( y_k = 1 \) and \( y_j = 0 \) for all \( j \neq k \). As noted in [13], \( h(x) \) is chosen from a finite size pool of weak learners, it usually cannot have the exact form as the function \(-g(x)\). One alternative approach is to select the \( h(x) \) that is the most parallel in the \( N \)-dimensional data space with the negative gradient \( \{-g(x^{(n)})\}_1^N \) by the following least-squares minimization problem:

\[
h_{k,m} = \arg \min_h \sum_{n=1}^N [-g_{k,m}(x^{(n)}) - h(x^{(n)})]^2
\]

(3.24)

After \( h_{k,m} \) is chosen, its weight \( \rho_{k,m} \) can be found by line search. Algorithm 1 illustrates the \( K \)-class GradientBoost approach.

Algorithm 1 \( K \)-class GradientBoost

1: \( F_{k,0} = 0, \ k = 1, \ldots, K \)
2: for \( m = 1 \) to \( M \) do
3: \( p_k(x) = \exp(F_k(x))/\sum_{j=1}^{K} \exp(F_j(x)), \ k = 1, \ldots, K \)
4: for \( k = 1 \) to \( K \) do
5: \( -g_{k,m}(x^{(n)}) = y^{(n)}_k - p_k(x^{(n)}), \ n = 1, \ldots, N \)
6: \( h_{k,m} = \arg \min_h \sum_{n=1}^N [-g_{k,m}(x^{(n)}) - h(x^{(n)})]^2 \)
7: \( \rho_{k,m} = \arg \min_\rho \sum_{n=1}^N \Psi(y^{(n)}_k, F_{k,m-1}(x^{(n)}) + \rho h_{k,m}(x^{(n)})) \)
8: \( F_{k,m} = F_{k,m}(x) + \rho_{k,m} h_{k,m}(x) \)
9: end for
10: end for

3.2 LatentBoost

In discriminative latent structure learning frameworks (i.e. latent SVM [9] or HCRF [29]), each training sample is usually assumed to be associated with a set of latent variables. The latent variables can be structured in some way. For example, in Section 4.1, we consider person tracks as latent variables, which are constrained by a chain structure. We emphasize that our LatentBoost algorithm is not limited to this chain structure but can be easily generalized to other types of latent structures.

We assume that an example \((x, y)\) is associated with a set of latent variables \( \mathbf{L} = \{l_1, l_2, \ldots, l_T\} \), where each latent variable takes its value from a discrete set, i.e. \( l_t \in \mathcal{L}^1 \). We

\footnote{To simplify notation, we assume the same set \( \mathcal{L} \). But our formulation can be generalized so that each}
assume these latent variables are constrained by an undirected graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote vertices and edges in the graph $\mathcal{G}$, respectively. For a fixed $L$, the scoring function of the $(x, L)$ pair for the $k$-th class can be written as the sum of a set of unary and pairwise potential functions:

$$
F_k(x, L) = \sum_{t \in \mathcal{V}} H^t_k(x, l_t) + \sum_{(t,s) \in \mathcal{E}} H^{t,s}_k(x, l_t, l_s)
$$

(3.25)

Under the boosting framework, we define $H^t_k(x, l_t)$ and $H^{t,s}_k(x, l_t, l_s)$ as linear combinations of weak learners:

$$
H^t_k(x, l_t) = \sum_{m=1}^{M} \rho^t_{k,m} h^t_{k,m}(x, l_t)
$$

(3.26)

$$
H^{t,s}_k(x, l_t, l_s) = \sum_{m=1}^{M} \rho^{t,s}_{k,m} h^{t,s}_{k,m}(x, l_t, l_s)
$$

(3.27)

Similar to GradientBoost, we define the probability of an example $x$ being class $k$ as:

$$
\hat{p}_k(x) = \frac{\sum_L \exp(F_k(x, L))}{\sum_L \sum_{k=1}^{K} \exp(F_j(x, L))}
$$

(3.28)

The difference (comparing Equation (3.1) and Equation (3.28)) from GradientBoost is that now we need to sum over $L$ since they are latent variables. Similarly, we can define the loss function for LatentBoost as the negative log-likelihood of the training data:

$$
\text{loss}_{LB} = \sum_{n=1}^{N} \Psi(\{y^{(n)}_k, F_k(x^{(n)}, L)\}_{k=1}^{K}) = - \sum_{n=1}^{N} \sum_{j=1}^{K} y^{(n)}_j \log \hat{p}^{(n)}_j(x^{(n)})
$$

(3.29)

Similar to GradientBoost, we learn the weak learners (both unary and pairwise) and their associated weights in an iterative fashion. Let us first focus on the unary potential $H^t_k(x, l_t)$. At the $m$-th iteration, the gradient of the loss function $\text{loss}_{LB}$ with respect to current strong learners can be calculated as:

$$
-g^t_{k,m}(x^{(n)}, l_t) = - \left[ \frac{\partial \Psi(\{y^{(n)}_j, F_j(x^{(n)}, L)\}_{j=1}^{K})}{\partial H^t_k(x^{(n)}, l_t)} \right]_{\{F_j(x,L)=F_j,m-1(x,L)\}_{j=1}^{K}}
$$

$$
= \frac{y^{(n)}_k \sum_L \exp(F_{k,m-1}(x^{(n)}, L))}{\sum_L \exp(F_{k,m-1}(x^{(n)}, L))} - \frac{\sum_L \exp(F_{k,m-1}(x^{(n)}, L))}{\sum_{j=1}^{K} \sum_L \exp(F_{j,m-1}(x^{(n)}, L))}
$$

$$
= y^{(n)}_k \Pr(l_t|x^{(n)}) - \Pr(y^{(n)}_k = 1, l_t|x^{(n)})
$$

(3.30)

latent variable is associated with a different label set.
The notation $\sum_{L,l_t}$ means summing over all possible values of $L$ with $l_t$ fixed. Then, we can obtain the optimal weak learner $h^t_{k,m}(x,l_t)$ by solving the following least-squares minimization problem:

$$h^t_{k,m} = \arg \min_{h^t} \sum_{n=1}^{N} \sum_{l_t \in L} [-g^t_{k,m}(x^{(n)},l_t) - h^t(x^{(n)},l_t)]^2$$

(3.31)

The steps to get the gradient in Equation (3.30) are the following, which are similar to the steps in GradientBoost. First, we take the first derivative of the loss function in Equation (3.29) with respect to the unary potential $H^t_k(x,l_t)$ which is associated to the latent variable $l_t$.

$$-g^t_{k,m}(x^{(n)},l_t) = -\left[ \frac{\partial \Psi(\{y^{(n)}_j, F_j(x^{(n)}, L)\}_{j=1}^{K})}{\partial H^t_k(x^{(n)},l_t)} \right]$$

(3.32)

$$= -\left\{ \frac{\partial}{\partial H^t_k(x^{(n)},l_t)} \left[ -\sum_{j=1}^{K} y^{(n)}_j \log \hat{p}^{(n)}_j(x^{(n)}) \right] \right\}$$

(3.33)

$$= \frac{\partial}{\partial H^t_k(x^{(n)},l_t)} \left[ y^{(n)}_k \log \hat{p}^{(n)}_k(x^{(n)}) \right] + \sum_{j \neq k} \frac{\partial}{\partial H^t_k(x^{(n)},l_t)} \left[ y^{(n)}_j \log \hat{p}^{(n)}_j(x^{(n)}) \right]$$

(3.34)

To simplify the equations, let us have the following:

$$N(x) = \sum_{L} \sum_{l=1}^{L} \exp(F_l(x,L))$$

(3.35)

$$N_k(x) = \sum_{L} \exp(F_k(x,L))$$

(3.36)

$$N_j(x) = \sum_{L} \exp(F_j(x,L))$$

(3.37)

$$D^t_k(x,l_t) = \frac{\partial}{\partial H^t_k(x,l_t)} \left[ \sum_{L} \exp(F_L(x,L)) \right] = \frac{\partial}{\partial H^t_k(x,l_t)} [N(x)] = \sum_{L,l_t} \exp(F_L(x,L))$$

(3.38)

By knowing Equations (3.25), (3.28), and (3.26), we can get the first derivative of the term with $y^{(n)}_k$ in Equation (3.34):
\[
\frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ y_k^{(n)} \log \hat{p}_k^{(n)}(x^{(n)}) \right] = y_k^{(n)} \frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ \hat{p}_k^{(n)}(x^{(n)}) \right]
\]

(3.39)

\[
= y_k^{(n)} \frac{1}{\hat{p}_k^{(n)}(x^{(n)})} \frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ \hat{p}_k^{(n)}(x^{(n)}) \right] = y_k^{(n)} \frac{1}{\hat{p}_k^{(n)}(x^{(n)})} \frac{N(x^{(n)})D_k^{(n)}(x^{(n)}, l^t) - D_k^{(n)}(x^{(n)}, l^t)N_k(x^{(n)})}{[N(x^{(n)})]^2}
\]

(3.40)

\[
= y_k^{(n)} \left[ \frac{N(x^{(n)})D_k^{(n)}(x^{(n)}, l^t)}{N_k(x^{(n)})N(x^{(n)})} - \frac{N_k(x^{(n)})D_k^{(n)}(x^{(n)}, l^t)}{N_k(x^{(n)})N(x^{(n)})} \right]
\]

(3.41)

\[
= y_k^{(n)} \left[ \frac{D_k^{(n)}(x^{(n)}, l^t)}{N_k(x^{(n)})} - \frac{D_k^{(n)}(x^{(n)}, l^t)}{N(x^{(n)})} \right]
\]

(3.42)

Similarly, we can get the first derivative of the terms with \( y_j^{(n)} \) in Equation (3.34):

\[
\frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ y_j^{(n)} \log \hat{p}_j^{(n)}(x^{(n)}) \right] = y_j^{(n)} \frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ \hat{p}_j^{(n)}(x^{(n)}) \right]
\]

(3.43)

\[
= y_j^{(n)} \frac{1}{\hat{p}_j^{(n)}(x^{(n)})} \frac{\partial}{\partial H_k^{(n)}(x^{(n)}, l^t)} \left[ \hat{p}_j^{(n)}(x^{(n)}) \right] = y_j^{(n)} \frac{1}{\hat{p}_j^{(n)}(x^{(n)})} \frac{0 \times N(x^{(n)}) - N_j(x^{(n)})D_k^{(n)}(x^{(n)}, l^t)}{[N(x^{(n)})]^2}
\]

(3.44)

\[
= -y_j^{(n)} \frac{N_j(x^{(n)})}{N_j(x^{(n)})} \frac{D_k^{(n)}(x^{(n)}, l^t)}{[N(x^{(n)})]^2}
\]

(3.45)

\[
= -y_j^{(n)} \frac{D_k^{(n)}(x^{(n)}, l^t)}{N(x^{(n)})}
\]

(3.46)

Combining Equations (3.34), (3.43), and (3.48), we have:
\[ -g_{k,m}^{t,n}(x^{(n)}, l_t) \]
\[ = y_k^{(n)} \left[ \frac{D_k^t(x^{(n)}, l_t)}{N_k(x^{(n)})} - \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} \right] + \sum_{u \neq k} -y_j^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} \]  
(3.49)

\[ = y_k^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N_k(x^{(n)})} - y_k^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} - \sum_{u \neq k} y_j^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} \]  
(3.50)

\[ = y_k^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N_k(x^{(n)})} - \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} \sum_{l=1}^{K} y_j^{(n)} \]  
(3.51)

\[ = y_k^{(n)} \frac{D_k^t(x^{(n)}, l_t)}{N_k(x^{(n)})} - \frac{D_k^t(x^{(n)}, l_t)}{N(x^{(n)})} \sum_{l} \text{exp}(F_k(x^{(n)})) \]  
(3.52)

\[ = y_k^{(n)} \sum_{L:L} \frac{\text{exp}(F_k(x^{(n)}))}{\sum_{L} \text{exp}(F_k(x^{(n)}))} \]  
(3.53)

\[ = y_k^{(n)} \Pr(t, l| x^{(n)}) - \Pr(y_k^{(n)} = 1, l, t| x^{(n)}) \]  
(3.54)

The negative gradient for the pairwise term is as follows:

\[ -g_{k,m}^{t,s}(x^{(n)}, l_t, l_s) = - \left[ \frac{\partial \Psi(\{y_j^{(n)}, F_j(x^{(n)}, L)\}_{j=1}^{K})}{\partial H_k^{t,s}(x^{(n)}, l_t, l_s)} \right] \mid_{F_j(x,L)=F_{j,m-1}(x,L)}^{j=1} \]  
(3.55)

The following steps show how to derive Equation (3.56). Again, we take the first derivative of the loss function in Equation (3.29) with respect to the pairwise potential \( H_k^{t,s}(x, l_t, l_s) \) which is associated with the latent variables \( l_t \) and \( l_s \).

\[ -g_{k,m}^{t,s}(x^{(n)}, l_t, l_s) \]  
(3.56)

\[ = - \left[ \frac{\partial \Psi(\{y_j^{(n)}, F_j(x^{(n)}, L)\}_{j=1}^{K})}{\partial H_k^{t,s}(x^{(n)}, l_t, l_s)} \right] \]  
(3.57)

\[ = - \left\{ \frac{\partial}{\partial H_k^{t,s}(x^{(n)}, l_t, l_s)} \left[ -\sum_{j=1}^{K} y_j^{(n)} \log \hat{p}_j^{(n)}(x^{(n)}) \right] \right\} \]  
(3.58)

\[ = \frac{\partial}{\partial H_k^{t,s}(x^{(n)}, l_t, l_s)} \left[ y_k^{(n)} \log \hat{p}_k^{(n)}(x^{(n)}) \right] + \sum_{j \neq k} \frac{\partial}{\partial H_k^{t,s}(x^{(n)}, l_t, l_s)} \left[ y_j^{(n)} \log \hat{p}_j^{(n)}(x^{(n)}) \right] \]  
(3.59)

To simplify the equations, let us have the following:
CHAPTER 3. BOOSTING WITH LATENT STRUCTURES

28

\[ D_{k,t,s}^L(x, l_t, l_s) = \frac{\partial}{\partial H_{k,t} (x, l_t, l_s)} \left[ \sum_L \exp(F_k(x, L)) \right] = \frac{\partial}{\partial H_{k,t} (x, l_t, l_s)} [N(x)] = \sum_{L;l,t,l_s} \exp(F_k(x, L)) \] (3.61)

Giving Equations (3.25), (3.28), and (3.27), we can derive the first derivative of the term with \( y_k(n) \) in Equation (3.60):

\[
\frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ y_k^{(n)} \log \hat{p}_k^{(n)} (x^{(n)}) \right] = y_k^{(n)} \frac{1}{\hat{p}_k^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_k^{(n)} (x^{(n)}) \right] \] (3.62)

\[
= y_k^{(n)} \frac{1}{\hat{p}_k^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_k^{(n)} (x^{(n)}) \right] \] (3.63)

\[
= y_k^{(n)} \frac{1}{\hat{p}_k^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_k^{(n)} (x^{(n)}) \right] \] (3.64)

\[
= y_k^{(n)} \frac{N(x^{(n)}) D_k^{L,s} (x^{(n)}, l_t, l_s) - N_k(x^{(n)}) D_k^{L,s} (x^{(n)}, l_t, l_s)}{N(x^{(n)}) N_k(x^{(n)})} \] (3.65)

\[
= y_k^{(n)} \left[ \frac{D_k^{L,s} (x^{(n)}, l_t, l_s)}{N_k(x^{(n)})} - \frac{D_k^{L,s} (x^{(n)}, l_t, l_s)}{N(x^{(n)})} \right] \] (3.66)

Similarly, for the term with \( y_j^{(n)} \) in Equation (3.60), we have:

\[
\frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ y_j^{(n)} \log \hat{p}_j^{(n)} (x^{(n)}) \right] = y_j^{(n)} \frac{1}{\hat{p}_j^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_j^{(n)} (x^{(n)}) \right] \] (3.67)

\[
= y_j^{(n)} \frac{1}{\hat{p}_j^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_j^{(n)} (x^{(n)}) \right] \] (3.68)

\[
= y_j^{(n)} \frac{1}{\hat{p}_j^{(n)} (x^{(n)})} \frac{\partial}{\partial H_{k,t}^L (x^{(n)}, l_t, l_s)} \left[ \hat{p}_j^{(n)} (x^{(n)}) \right] \] (3.69)

\[
= y_j^{(n)} \frac{0 \times N(x^{(n)}) - N_j(x^{(n)}) D_k^{L,s} (x^{(n)}, l_t, l_s)}{N(x^{(n)})^2} \] (3.70)

\[
= y_j^{(n)} \frac{N_j(x^{(n)}) D_k^{L,s} (x^{(n)}, l_t, l_s)}{N_j(x^{(n)}) N(x^{(n)})} \] (3.71)

Combining Equations (3.60), (3.66), and (3.71), we have:
\[ -g^{l,s}_{k,m}(x^{(n)}, l_t, l_s) \]
\[ = y_k^{(n)} \left[ \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N_k(x^{(n)})} - \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} \right] + \sum_{j \neq k} y_j^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} \]
\[ = y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N_k(x^{(n)})} - y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} + \sum_{j \neq k} y_j^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} \]
\[ = y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N_k(x^{(n)})} - y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} + \sum_{j=1}^{K} y_j^{(n)} \]
\[ = y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N_k(x^{(n)})} - y_k^{(n)} \frac{D^{l,s}_k(x^{(n)}, l_t, l_s)}{N(x^{(n)})} \]
\[ = y_k^{(n)} \Pr(l_t, l_s|x^{(n)}) - \Pr(y_k^{(n)} = 1, l_t, l_s|x^{(n)}) \]

From the derivation above, we get the following:

\[ \Pr(l_t|x^{(n)}) = \frac{\sum_{L, l_t} \exp(F_k(x)^{(n)}, L)}{\sum_L \exp(F_k(x)^{(n)}, L)} \] (3.78)

\[ \Pr(l_t, l_s|x^{(n)}) = \frac{\sum_{L, l_t, l_s} \exp(F_k(x)^{(n)}, L)}{\sum_L \exp(F_k(x)^{(n)}, L)} \] (3.79)

\[ \Pr(y_k^{(n)} = 1, l_t|x^{(n)}) = \frac{\sum_{L, l_t} \exp(F_k(x)^{(n)}, L)}{\sum_L \sum_j^{K} \exp(F_j(x^{(n)}), L)} \] (3.80)

\[ \Pr(y_k^{(n)} = 1, l_t, l_s|x^{(n)}) = \frac{\sum_{L, l_t, l_s} \exp(F_k(x)^{(n)}, L)}{\sum_L \sum_j^{K} \exp(F_j(x^{(n)}), L)} \] (3.81)

The marginal distributions \( \Pr(l_t|x^{(n)}) \), \( \Pr(y_k^{(n)} = 1, l_t|x^{(n)}) \), \( \Pr(l_t, l_s|x^{(n)}) \) and \( \Pr(y_k^{(n)} = 1, l_t, l_s|x^{(n)}) \) above can be computed efficiently by using Belief Propagation (or be approximated by using Loopy Belief Propagation if the latent structure has cycles). If we have a latent tree structure, Belief Propagation can compute all the summations in the marginal distributions efficiently. If our latent structure has cycles, the information can flow many times around the structure by the message passing rules in Loopy Belief Propagation. For some models, the algorithm will converge, whereas for others it will not. Once the algorithm has converged, or once it has been stopped if convergence is not observed, approximated local marginals can still be computed.
After $h_{k,m}^t$ is chosen, the weight $\rho_{k,m}^t$ can be simply computed by a line search algorithm. The weight $\rho_{k,m}^{t,s}$ of $h_{k,m}^{t,s}$ can be obtained in the same way. Putting everything together, we have the LatentBoost algorithm illustrated in Algorithm 2.

**Algorithm 2 LatentBoost**

1: $F_{k,0} = 0, \ k = 1, \ldots, K$
2: for $m = 1$ to $M$ do
3: Compute $\Pr(l_t|x^{(n)})$, $\Pr(y_k^{(n)} = 1, l_t|x^{(n)})$, $\Pr(l_t,l_s|x^{(n)})$ and $\Pr(y_k^{(n)} = 1, l_t,l_s|x^{(n)})$ $\forall n, \forall t \in V, \forall (t,s) \in E$
4: for $k = 1$ to $K$ do
5: //update unary potentials
6: for $t \in V$ do
7: $h_{k,m}^t = \arg \min_{h^t} \sum_{n=1}^N \sum_{l_t} [-g_{k,m}^t(x_n, l_t) - h^t(x_n, l_t)]^2$
8: $\rho_{k,m}^t = \arg \min_{\rho^t} \sum_{n=1}^N \Psi(y_k^{(n)}, h_{k,m-1}^t(x^{(n)}, l_t) + \rho^t h_{k,m}^t(x^{(n)}, l_t))$
9: $F_{k,m}^t(x, l_t) = F_{k,m-1}^t(x, l_t) + \rho_{k,m}^t h_{k,m}^t(x, l_t)$
10: end for
11: //update pairwise potentials
12: for $(t,s) \in E$ do
13: $h_{k,m}^{t,s} = \arg \min_{h^{t,s}} \sum_{n=1}^N \sum_{l_t,l_s} [-g_{k,m}^{t,s}(x^{(n)}, l_t, l_s) - h^{t,s}(x^{(n)}, l_t, l_s)]^2$
14: $\rho_{k,m}^{t,s} = \arg \min_{\rho^{t,s}} \sum_{n=1}^N \Psi(y_k^{(n)}, F_{k,m-1}^{t,s}(x^{(n)}, l_t, l_s) + \rho^{t,s} h_{k,m}^{t,s}(x^{(n)}, l_t, l_s))$
15: $F_{k,m}^{t,s}(x, l_t, l_s) = F_{k,m-1}^{t,s}(x, l_t, l_s) + \rho_{k,m}^{t,s} h_{k,m}^{t,s}(x, l_t, l_s)$
16: end for
17: end for
18: end for
Chapter 4

Human Action Recognition With LatentBoost

We test LatentBoost on the task of human action recognition. In Section 4.1, we describe the LatentBoost model we implement. In Section 4.2, we describe the features we use and implementation details of LatentBoost. In Section 4.3, we show the results of our experiments.

4.1 Model

Our method for human action recognition operates on a “figure-centric” representation of the human figure extracted from an input video. The figure-centric representation is obtained by running a human detection/tracking algorithm over the input video. We use the term tracklet to denote short 5-frame long human trajectories returned by our tracker. Our method will operate on these tracklets, and classify them into one of $K$ actions.

A tracklet is denoted by 5 tuples $x_t = (I_t, u_t, v_t)$, $t = 1, \ldots, 5$ where $I_t$ is the image feature and $(u_t, v_t)$ are the position of person in the $t$-th frame of the tracklet. Since the tracker is not completely reliable, we allow each frame of a tracklet to move around in a fixed range with respect to its initial position from the tracker. This will provide some robustness against imperfections of the tracker. We denote the offset of the frame $t$ in a tracklet using a latent variable $l_t \in \mathcal{L}$. This results in five latent variables per tracklet. We define an offset as $l_t = (dx, dy)$ within a fixed range $W$, so that $\mathcal{L} = \{(dx, dy) | -W \leq dx, dy \leq W\}$. After
applying an offset in a latent variable $l_t$, we can refine the position of the person in frame $t$ of a tracklet in a video.

We assume these five latent variables to form a chain structure, as shown in Figure 4.1. For each frame of a tracklet, we will define unary features that can be used to classify the action of a person. As described below, in our implementation these will be based on optical flow values in the frame. These features will depend on the latent variable for a frame, which provides the offset to refine the position of the person.

In addition, we will define a set of pairwise features in our model that relate variables in adjacent frames of a tracklet. These features will be used to enforce appearance consistency, a tracking constraint, over the values of adjacent offset latent variables $l_t$ and $l_{t+1}$. These features will be based on colour similarity in our implementation. Since the model forms a chain structure, the marginal distributions in Equation 3.30 and 3.56 can be computed efficiently via belief propagation.

### 4.2 Features and Implementation Details

Our model is built upon the optical flow features in [7] and colour histogram features. The optical flow features are for the unary potentials in the model to describe motions, and they have been shown to perform reliably with noisy image sequences for action classification. The colour histogram features are used by the pairwise potentials. They impose tracking constraints between two consecutive frames.

**Optical flow features:** To compute the optical flow features, we use entire frames from the input video instead of just the stabilized human figure in the tracklets in order to
obtain absolute human figure motion. The Lucas and Kanade [24] algorithm is employed to compute the optical flow of each frame. Following [7], the optical flow vector field is then split into 4 scalar fields channels corresponding to different flow directions, and blurred. We add another channel corresponding to the motion magnitude which is obtained by computing the $L_2$ norm of an optical flow vector.

One optical flow feature $f_j(x_t)$ is defined as the motion in a channel $c$ at a location $(u, v)$. Let $U = \{f_j(x_t) : j = (c, u, v)\}$ be the pool of possible optical flow features. A weak learner $h^t$ used in a unary potential $H^t_k(\cdot)$ picks one of the features $f_j \in U$:

$$h^t_j(x_t, l_t) = \begin{cases} 1 & \text{if } p_j f_j(x_t + l_t) < p_j \theta_j \\ -1 & \text{otherwise} \end{cases} \quad (4.1)$$

where the operation + denotes applying the tracker offset in $l_t$ to the position of the person in $x_t$.

**Colour histogram features:** The pairwise features in our model are built from comparing colour histograms. We compare the colour histograms in sub-windows of the tracklets to enforce a tracking constraint between adjacent person locations.

One colour histogram feature $f_j(x_t, x_{t+1})$ is defined as the difference between colour histograms in rectangular sub-windows taken from adjacent frames. Let $P = \{f_j(x_t, x_{t+1})\}$ denote the pool of possible colour histogram features, with $j$ enumerating over a set of image sub-windows. In our experiments we use image sub-windows of fixed size (i.e. $15 \times 15$).

A weak learner $h^{t,t+1}_j$ on pairwise potential $H^{t,t+1}_k(\cdot)$ picks one of the features $f_j \in P$:

$$h^{t,t+1}_j(x_t, x_{t+1}, l_t, l_{t+1}) = \begin{cases} 1 & \text{if } p_j f_j(x_t + l_t, x_{t+1} + l_{t+1}) < p_j \theta_j \\ -1 & \text{otherwise} \end{cases} \quad (4.2)$$

again using the same + notation as above.

The weak learners in Equation (4.1) and (4.2) are decision stumps. Each weak learner consists of a feature $f_j$, a threshold $\theta_j$, and a parity $p_j$ indicating the direction of the inequality sign.

**Line search:** The optimal $\rho^t_k$ and $\rho^{t,s}_k$, which are the weights of $h^t_k$ and $h^{t,s}_k$ respectively, are computed by a line search algorithm. However, there is no closed form solution to the optimization problem. Our approach is to approximate them with a single Newton-Raphson step similar to the one in [13].
Figure 4.2: Per-frame and per-video classification accuracies on the Weizmann dataset with different stride configurations and 5-frame tracklet on LatentBoost.

4.3 Experiments

We present experimental results on two publicly available datasets: Weizmann human action dataset [3] and TRECVID surveillance event detection [33]. We show that LatentBoost outperforms a baseline using GradientBoost (Section 3.1) in these two datasets. Both LatentBoost and GradientBoost are trained with the same set of features including both optical flow features and colour histogram features. For GradientBoost, we fix the latent variable to be zeros (i.e. $(dx = 0, dy = 0)$), which corresponds to the case of no latent variables.

**Weizmann dataset**: We first present results on the Weizmann dataset, which is a standard benchmark dataset for action recognition. It consists of 83 videos showing nine different people, each performing nine different actions: running, walking, jumping jack, jumping forward on two legs, jumping in place on two legs, galloping sideways, waving two hands, waving one hand, and bending.

We track and stabilize the figures using the background subtraction masks that come with the dataset. We randomly choose videos of five subjects as the training set, and the videos of the remaining four subjects as the test set. We allow each frame in a tracklet to have an offset of at most 25 pixel locations centred around its initial position in the tracklet.

For our experiments, we report both the per-frame accuracy and per-video accuracy.
CHAPTER 4. HUMAN ACTION RECOGNITION WITH LATENTBOOST

Figure 4.3: Typical tracklets from the Weizmann dataset. The four actions are jacking, running, jumping, and waving.

Since we start a tracklet at every frame, we classify every tracklet as the per-frame classification. The per-frame accuracy is the percentage of tracklets that are correctly classified in the test set. We use majority voting to get the label for a video. The most voted label from the tracklets in a video is the label of the video. The per-video accuracy is the percentage of videos that are correctly classified in the test set.

To reduce the computation, we do not use all possible optical flow features and colour histogram features. We construct the pool of features by sampling with fixed strides. For example, for $5 \times 5$ strides, we pick every $5^{th}$ point in every $5^{th}$ row in each optical flow channel as our motion features. Similarly, we pick every $5^{th}$ pixel in every $5^{th}$ row in each frame as the top-left corner of the image sub-window to compute the colour histogram features. We have experimental results for $5 \times 5$, $7 \times 7$, $9 \times 9$, and $11 \times 11$ strides in Figure 4.2. From the plots, we can see that all four configurations have similar performance. Moreover, all four configurations have the same performance for the first 14 iterations. The $5 \times 5$ configuration achieves 100% per-video accuracy with the fewest iterations. However, it takes the longest time in training. On the other hand, the $11 \times 11$ configuration has the fastest training time and it also achieves 100% per-video accuracy with a few more iterations compared to the $5 \times 5$ configuration. These experiments show the feature selection ability of boosting algorithms. The feature sets in the $11 \times 11$ configurations are much smaller compared to the feature sets in the $5 \times 5$ configurations. LatentBoost can still pick the critical features in both configurations to achieve similar performance.

We test the effectiveness of the pairwise terms in our model. The pairwise terms in the model are considered as tracking constraints, which are used to enforce the appearance consistency between two consecutive frames. We modify the original LatentBoost in Section
3.2 to select the best weak learner among all unary potentials and pairwise potentials in each iteration, instead of selecting one weak learner for every potential in each iteration. With this setting, both the models with and without pairwise terms have the same number of weak learners in each iteration. We find that the pairwise features are not significant in this dataset. This might be because we can achieve reliable tracking in this dataset. Figure 4.3 shows some typical tracklets in the dataset. We can see that the tracking is very good and there is not much jitter in the tracklets. The model with pairwise terms only selects 5\% of its weak learners from the pairwise feature pool. Figure 4.4 shows the result on the setting with $5 \times 5$ strides and 5-frame tracklet.

Although we describe our model to have 5 frames in Section 4.1, we also try tracklets with 7 and 10 frames. We again select the best weak learner among all unary and pairwise in each iteration. With this setting, all three models have the same number of weak learners in each iteration. Figure 4.5 shows the experimental results. Note that the stride configuration is $5 \times 5$. The experimental results show that more frames in a tracklet do provide extra information but the impact is not significant. If a model has more frames, it takes longer to be trained.

Beside comparing with different settings, we compare LatentBoost with GradientBoost in Section 3.1 and the work in [39]. Both LatentBoost and GradientBoost use $5 \times 5$ strides.
CHAPTER 4. HUMAN ACTION RECOGNITION WITH LATENTBOOST

Figure 4.5: Per-frame and per-video classification accuracies on the Weizmann dataset with different lengths of tracklets and $5 \times 5$ strides configuration on LatentBoost.

<table>
<thead>
<tr>
<th></th>
<th>per-frame</th>
<th>per-video</th>
</tr>
</thead>
<tbody>
<tr>
<td>LatentBoost</td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>GradientBoost</td>
<td>0.93</td>
<td>0.97</td>
</tr>
<tr>
<td>MMHCRF [39]</td>
<td>0.93</td>
<td>1.0</td>
</tr>
<tr>
<td>HCRF [39]</td>
<td>0.90</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of classification accuracy with similar baselines on the Weizmann dataset.

configuration and 5-frame tracklets. The comparative results are shown in Table 4.3. LatentBoost achieves the typical near-perfect results, and slightly outperforms GradientBoost, max-margin hCRF (MMHCRF) and probabilistic hCRF in [39]. A visualization of the positive optical flow features in the final LatentBoost model is shown in Figure 4.6. These experiments provide evidence for the effectiveness of our LatentBoost-based action recognition model, albeit on a very simple dataset. Therefore, we focus next on the much more challenging TRECVID dataset.

**TRECVID dataset**: The Weizmann dataset is relatively simple and the performance on this dataset has already saturated. To further demonstrate our model, we have applied it on a much more challenging dataset from the TRECVID surveillance event detection challenge. The dataset consists of surveillance camera footage acquired at London Gatwick
airport. The dataset is large in scale. There are 5 fixed-view surveillance cameras, and each camera has 10 videos, each 2 hours long. The TRECVID challenge aims to locating the starting/ending frame during which one of the following 7 events occurs: PersonRuns, CellToEar, ObjectPut, PeopleMeet, PeopleSplitUp, Embrace, and Pointing.

To test our algorithm, we focus on the PersonRuns event and use the videos from camera 5. Most of the PersonRuns events from camera 5 are captured from a common, canonical side view. We follow the same evaluation criteria of TRECVID event detection challenge. Note that TRECVID is very challenging. The PersonRuns event is the only event for which state-of-the-art algorithms can achieve reasonable performance (i.e. better than random guessing). In TRECVID 2011 [26], the best minimum Detection Cost Rate (DCR) of PersonRuns is 0.83. Note that DCR = 0 is a perfect system, and DCR = 1.0 is equivalent to a system that never detects anything. The minimum DCRs for most of other actions are larger than 1.0.

Both Detection Cost Rate (DCR) scores and Detection-Error Tradeoff (DET) curves are the standard evaluation criteria of TRECVID challenge. DET curve plots a series of event-averaged missed detection probabilities and false alarm rates that are a function of a detection threshold. The threshold is applied to the our system’s detection scores meaning that the system observations with scores above the threshold are declared to be the set of detected observations. After the threshold is applied, the measurements are then computed separately for each event, then averaged to generate a DET line trace. DCR is a weighted linear combination of the system’s missed detection probability and false alarm rate. It is defined as the following formula:
Figure 4.7: Comparison of GradientBoost with our LatentBoost on the TRECVID dataset. (a) The DET curves of the two methods. (b) The minimum DCR scores of the two methods. (c) Examples of running detection in the TRECVID dataset. Note that the DET curve is the lower the better and the minimum DCR score is also the lower the better. Please zoom in for a clear view of the plot.
\[ DCR(S, E, \theta) = P_{Miss}(S, E, \theta) + \beta \ast R_{FA}(S, E, \theta) \] (4.3)

where \( S \) is our system, \( E \) is the event, \( \theta \) is the decision score threshold, and \( \beta \) is a constant pre-define weight. The measure’s derivation can be found in [26].

We randomly chose 5 videos of camera 5 as the training set, and the remaining 5 videos of camera 5 as the test set. To generate the positive training set, we first obtain the ground truth video clips from the 5 training videos. Each ground truth video clip contains at least one running subject. For each ground truth video clip, we run a tracker to generate all possible tracklets, and then we manually select the tracklets containing a running person (the ground truth provided in TRECVID dataset does not contain spatial localization). For the negative training set, we randomly select not-running tracklets from the rest of the training videos. The final training set consists of 800 running tracklets and 1800 not-running tracklets. Notice that a person who is far from the camera is much smaller than a person who is close to the camera in camera 5. We resize the frames of every tracklet to the same size (i.e. \( 29 \times 60 \) pixels), and use the resized tracklets to train both LatentBoost and GradientBoost. Similarly, we also need to normalize the magnitude of optical flow features across all tracklets to the same size.

For this dataset, we again allow each frame in a tracklet to have an offset of at most 25 pixel locations centred around its initial position given by the tracker. After training the classifiers, we apply them to the test videos. There are 10 hours of test videos in total. The event detection system we use is a standard tracking-and-classification detection system from the TRECVID Workshop 2010 [15]. It has three steps: a pre-processing step to detect and track humans, classification, and a post-processing step for non-maximum suppression.

We implement a naive tracker in our system. First, we use background subtraction to find all the moving regions in a frame. We then use a HOG [4] detector to find people in these moving regions. Giving a moving region that contains a person, we perform an exhaustive searching on each of these four directions: right, top, left, and bottom. The exhaustive searching is done in a frame-by-frame manner. We assume that the person either stays in the same region or moves no further than a certain distance (i.e. 10 pixels) in the next frame and that the person does not change his or her moving direction rapidly. The certain distance defines a fixed searching space in the next frame for a given region and a given direction. We then use the HOG detector to try to find the same person in the searching space. If the same person is found in the next frame, the next frame becomes the current
frame and the tracking is continued with the same procedure. The tracking is terminated when either a tracklet of 5 frames is constructed or the tracking person cannot be found by the system in the next frame. The tracklets that have fewer than 5 frames are discarded.

The results are shown in Figure 4.3 (a) and (b). Detection-Error Tradeoff (DET) curves and minimum Detection Cost Ratio (DCR), a summary statistic, are shown. These are the standard TRECVID evaluation criteria. Again, LatentBoost outperforms GradientBoost, this time on a much challenging and realistic dataset. Examples of running detection are shown in Figure 4.3 (c).

We also test the effectiveness of the pairwise terms in our model. The model with pairwise terms selects only the best weak learner among unary potentials and pairwise potentials in each iteration. By having such setting, both the model with and without pairwise terms will have the same number of weak learners in each iteration. The final models have 700 weak learners. Our experimental result shows that the pairwise terms have

Figure 4.8: Comparison of LatentBoost with and without pairwise terms on TRECVID dataset.
CHAPTER 4. HUMAN ACTION RECOGNITION WITH LATENTBOOST

Figure 4.9: Typical tracklets from the TRECVID dataset. The first row consists two 5-frame tracklets of running people. The second row consists two 5-frame tracklets of not-running people.

By far, the TRECVID dataset is considered the most challenging for action recognition and detection. Other than that the events have low frequencies, we notice other challenges. First, the event PersonRuns has the lowest frequency among all the events. It has about 7 instances per hour. The subjects of one third of all the running events are children. Our tracker is not trained with any children examples. Also, running children are harder to detect if just considering their motion magnitude. A running child may have the same motion magnitude as a walking adult. Figure 4.10 shows some examples of running children. We also see that children tend to have more variants of running poses. Beside categorizing the running instances into adult and child, we also can categorize them into different running paths. In the view of camera 5, there are two paths. One is parallel to the camera view, and the other one is about 30 degree up from the left bottom corner. Having two different paths in the same view increases the difficulty. The classifier has to recognize two different patterns as running if we train a binary classifier. Figure 4.11 shows some examples of the running detections in the two paths. Some other challenges in this dataset are more subtle.
Figure 4.10: Running children in the TRECVID dataset.

For example, the running subject in the left image of Figure 4.12 is so small. The whole running event only takes about 1 second. Short event with low resolution video makes it difficult to be detected. The image on the right side of Figure 4.12 shows a typical problem for detection task. The running subject is the child inside the blue bounding box. However, the child is always occluded during the whole event. Locating and tracking an occluded object could be a challenge.
Figure 4.11: Examples of running detections.

Figure 4.12: Hard examples in the running detection.
Chapter 5

Conclusion

In this thesis, we present LatentBoost, a novel learning algorithm for training models with latent variables in a boosting framework. The algorithm allows for training of structured latent variable models with boosting. The popular Latent SVM [9] framework allows for training of models with structured latent variables in a max-margin framework. LatentBoost provides an analogous capability for boosting algorithms. The effectiveness of this framework is demonstrated by an application to human action recognition. In the following, we briefly highlight the limitations of LatentBoost and the future research.

5.1 Limitations

There are some limitations in LatentBoost. First, LatentBoost uses gradient descent method, and it is not guaranteed to find the global optimum in a non-convex problem. The performance of the final classifier is very sensitive to the initialization. LatentBoost does not provide any good strategy for initialization beside setting everything to be zero. Inference would also be an issue to LatentBoost. If the latent structure is not a tree structure, theoretically LatentBoost can still perform inference with Loopy Belief Propagation (Loopy BP). However, Loopy BP is much slower compared to Belief Propagation and Loopy BP is not exact. Last, as the discussion in [39], the summation over all the possible latent variables in LatentBoost may cause problems. Recall that LatentBoost needs to compute the summation of all possible \( L \)'s in the following form:
The intuition behind the equation above is that a correct $L$ will have a larger value of probability $p(y, L|x)$, while an incorrect $L$ will have a smaller value. Hence correct $L$ will contribute more to $p(y|x)$. However, this is not necessarily the case. The latent variable $L$ has exponentially many possible configurations. But only a very small number of those configurations are correct ones. An incorrect $L$ only carries a small probability, but since there are exponentially many of them, the summation in the equation above can still be dominated by those incorrect $L$’s. This effect is exactly the opposite of what we have expected the model to behave. The effect is partly due to the high dimensionality of the latent variable $L$, and many counterintuitive properties of high dimensional spaces have also been observed in statistics and machine learning.

We understand the weakness of summation of all latent variables. Before formulating LatentBoost as the summation over all the possible latent variables, we also try a max-margin approach. A boosting algorithm produces a final classifier $f_\alpha = \sum_{m=1}^{M} \alpha_m h_m(x)$ where $h_m$ is the weak learner added in the $m^{th}$ iteration and $\alpha_m$ is the coefficient of $h_m$. The margin of a given example $(x^{(n)}, y^{(n)})$ is defined as $\rho(x^{(n)}, y^{(n)}) = y^{(n)} f_\alpha(x^{(n)})$. The margin of a set of examples is always the minimum over all examples. In a max-margin approach, we extends the margin to have latent variables. The margin of a given example $(x^{(n)}, y^{(n)})$ is $\rho(x^{(n)}, y^{(n)}) = y^{(n)} \max_L f_\alpha(x^{(n)}, L)$. Given the definition of the margin in the latent boosting setting, we can extend the linear programming problem in [40] as the following:

$$
\rho^*_t(\nu) = \max_{\alpha, \rho, \psi} \left( \rho - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n \right)
$$

s.t. $\rho - \psi_n \leq y^{(n)} \max_L f_\alpha(x^{(n)}, L), \forall n,$

$$
\sum_{m=1}^{M} \alpha_m = 1,
\alpha_m \geq 0,
\psi_n \geq 0.
$$

(5.2)

where $\psi_n$ are the slack variables. However, this linear programming problem is not feasible in general. Therefore, we do not continue with this approach.
5.2 Future Work

In the thesis, we have done a very limited small set of experiments. In the future, we would try other features such as space-time interest points like HOG3D [19]. We also should try our model on the whole TRECVID dataset.

LatentBoost is a general machine learning algorithm. We have only applied it to the task of action recognition and detection. We would apply LatentBoost to other computer vision problems such as object recognition and detection. To fully evaluate LatentBoost, we should do an extensive experimental comparison between LatentBoost and Latent SVM [9]. The part based model for objection detection in [9] is a very good application for LatentBoost to start with.
Bibliography


